Artificial Intelligence Empowered Virtual Network Function Deployment and Service Function Chaining for Next-Generation Networks

by

Mahzabeen Emu

B.Sc. Ahsanullah University of Science and Technology, 2017

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE (SPECIALIZATION IN ARTIFICIAL INTELLIGENCE)

Artificial Intelligence Empowered Virtual Network Function Deployment and Service Function Chaining for Next-Generation Networks

by

Mahzabeen Emu

Supervisory Committee

Dr. Salimur Choudhury,
Supervisor
(*Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada*)

Dr. Yimin Yang,
Internal Examiner
(*Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada*)

Dr. Khaled Rabie,
External Examiner
(*Department of Engineering, Manchester Metropolitan University, United Kingdom*)

ABSTRACT

The entire Internet of Things (IoT) ecosystem is directing towards a high volume of diverse applications. From smart healthcare to smart cities, every ubiquitous digital sector provisions automation for an immersive experience. Augmented/Virtual reality, remote surgery, and autonomous driving expect high data rates and ultra-low latency. The Network Function Virtualization (NFV) based IoT infrastructure of decoupling software services from proprietary devices has been extremely popular due to cutting back significant deployment and maintenance expenditure in the telecommunication industry. Another substantially highlighted technological trend for delay-sensitive IoT applications has emerged as multi-access edge computing (MEC). MEC brings NFV to the network edge (in closer proximity to users) for faster computation.

Among the massive pool of IoT services in NFV context, the urgency for efficient edge service orchestration is constantly growing. The emerging challenges are addressed as collaborative optimization of resource utilities and ensuring Quality-of-Service (QoS) with prompt orchestration in dynamic, congested, and resource-hungry IoT networks. Traditional mathematical programming models are NP-hard, hence inappropriate for time-sensitive IoT environments. In this thesis, we promote the need to go beyond the realms and leverage artificial intelligence (AI) based decision-makers for “smart” service management. We offer different methods of integrating supervised and reinforcement learning techniques to support future-generation wireless network optimization problems. Due to the combinatorial explosion of some service orchestration problems, supervised learning is more superior to reinforcement learning performance-wise. Unfortunately, open access and standardized datasets for this research area are still in their infancy. Thus, we utilize the optimal results retrieved by Integer Linear Programming (ILP) for building labeled datasets to train supervised models (e.g., artificial neural networks, convolutional neural networks). Furthermore, we find that ensemble models are better than complex single networks for control layer intelligent service orchestration. Contrarily, we employ Deep Q-learning (DQL) for heavily constrained service function chaining optimization. We carefully address key performance indicators (e.g., optimality gap, service time, relocation and communication costs, resource utilization, scalability intelligence) to evaluate the viability of prospective orchestration schemes. We envision that AI-enabled network management can be regarded as a pioneering tread to scale down massive IoT resource fabrication costs, upgrade profit margin for providers, and sustain QoS mutually.

ACKNOWLEDGEMENTS

"Life is not easy for any of us. But what of that? We must have perseverance and above all confidence in ourselves. We must believe that we are gifted for something, and that this thing, at whatever cost, must be attained."

– Marie Curie

I would like to give thanks to the following funding sources, for their financial support to my research:

- **Lakehead University & Ontario Provincial Government for Ontario Graduate Scholarship**;
- **Lakehead University Faculty of Graduate Studies**;
- **Lakehead University Faculty of Science and Environmental Studies**;
- **Mesh AI and Mitacs**;
- **Dr. Salimur Choudhury**;
- **Toronto Vector Institute**.

I am deeply indebted to my supervisor Dr. Salimur Choudhury for his continuous feedback, support, motivation, and much more. I also thank him for providing me with the opportunity to grow as a research student in many ways with his expertise and the unique research environment he creates. My thanks and appreciation go to my former lab mate Peizhi Yan for his input and collaboration in Chapter 3. I am extremely grateful to Dr. Yimin Yang and Dr. Khaled Rabie for their constructive comments and valuable feedbacks on this thesis.

Last but not the least, I would like to express my gratitude towards my family - my parents, sister, husband, and in-laws for their unconditional support, encouragement, and love. In addition, I want to thank my aunts and brilliant cousins for their wise counsel, positive influence, and unlimited motivation.

PUBLICATIONS

Parts of this thesis have been published or accepted for publication:

- M. Emu and S. Choudhury, "IoT Ecosystem on Exploiting Dynamic VNF Orchestration and Service Chaining: AI to the Rescue?," in IEEE Internet of Things Magazine, vol. 3, no. 4, pp. 30-35, December 2020. (part of Chapter 2 and 6)
- M. Emu, P. Yan, and S. Choudhury, "Latency Aware VNF Deployment at Edge Devices for IoT Services: An Artificial Neural Network Based Approach," 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 2020, pp. 1-6. (part of Chapter 3)
- M. Emu and S. Choudhury, "Ensemble Deep Learning Aided VNF Deployment for IoT Services," 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2020, pp. 1-7. (part of Chapter 4)
- M. Emu and S. Choudhury, "Towards 6G Networks: Ensemble Deep Learning Empowered VNF Deployment for IoT Services," 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2021, pp. 1-4. (part of Chapter 4)
- M. Emu and S. Choudhury, "DSO: An Intelligent SFC Orchestrator for Time and Resource Intensive Ultra Dense IoT Networks," 2021 IEEE International Conference on Communications (ICC), Montreal, Canada, 2021. (Accepted, In press)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The worldwide deployment of upcoming 5G wireless networks is provisioned to lay a foundation of Artificial Intelligence (AI) based network services. Yet, the ultimate aim to furnish fully-intelligent networks and render a thoroughly immersive user experience can only be realized in Beyond 5G (B5G) future networks [1]. The prospective 6G telecommunication industry is anticipated to be driven by automated, self-configurable, and on-the-fly suitable operations to secure many-fold enhancement in overall network performance, especially service management [2]. The state-of-the-art visions for 6G are considered as a complex connected network with the ability to respond to the service calls rapidly by learning from concerned network states. The network states can be defined by the edge information (e.g., cache pattern), user-specifics (e.g., locations, battery-life), even air interface (e.g., radio propagation channel, radio-frequency), and so forth [3]. The unbelievably rapid growth of the Internet of Things (IoT) devices significantly contributes to the increasing complexity and size of the future communication networks. The active IoT market will reach approximately 24 billion of devices by 2030, while each person will nearly own 15 “connected things” [4]. Figure 1.1 illustrates the relation between each mobile generation system and IoT.

The revolution of 6G will transform the “connected things” into “connected intelligence” for ubiquitous wireless connectivity. The breakdown of present and upcoming IoT services are mainly categorized into four sections: connected living, productivity, smart health, and entertainment [5]. Remotely operated e-Home devices and office automation can boost the smart living environment and cause an extraordinary productivity leap. Furthermore, wearable IoT health devices bring a fresh outlook to improve well-being and fitness monitoring for healthcare units, professional ath-

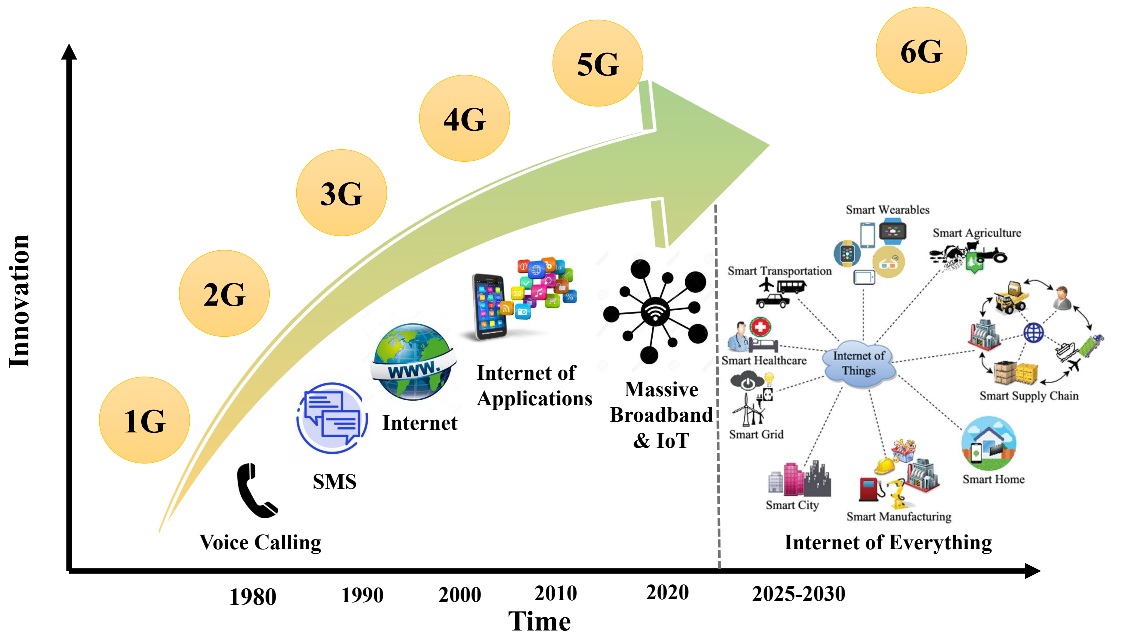


Figure 1.1: The evolution of mobile generation and IoT

letes, and millennials. On a crucial pitch, IoT in health sectors has revolutionized telemedicine, telesurgery, and surgical robots as well. A digital landscape shift towards immersive entertainment includes crystal-clear video or audio streaming, cloud gaming, user-specific services, and more. The full potential of the aforementioned IoT services can be unraveled only depending on the actualization of 6G communications. Figure 1.2 points out various aspects related to future 6G networks. AI is one of the most integral parts of 6G architecture as listed in the Figure 1.2. The AI embedding into next-generation networks can facilitate the widespread application of potential use cases by resolving different issues existing with traditional algorithms [3] [6] [7] [8]. Especially, AI algorithms have been gaining a lot of attention since most of today's devices own enough computational power. Moreover, every sector of research is following the trend of accommodating smart or intelligent solutions. AI methods subsume multidisciplinary approaches, such as optimization theory, machine learning (ML), deep learning (DL), meta-heuristics, and game theory [3]. Specifically, ML and DL are the most popular sub-fields that are broadly embraced in wireless networks [3]. DL is a more advanced category of AI techniques for enabling a machine to learn and perform intelligent tasks with better accuracy compared to ML without human intervention. Therefore, it is worth investigating the strengths and limitations of ML/DL in future wireless system researches likewise.

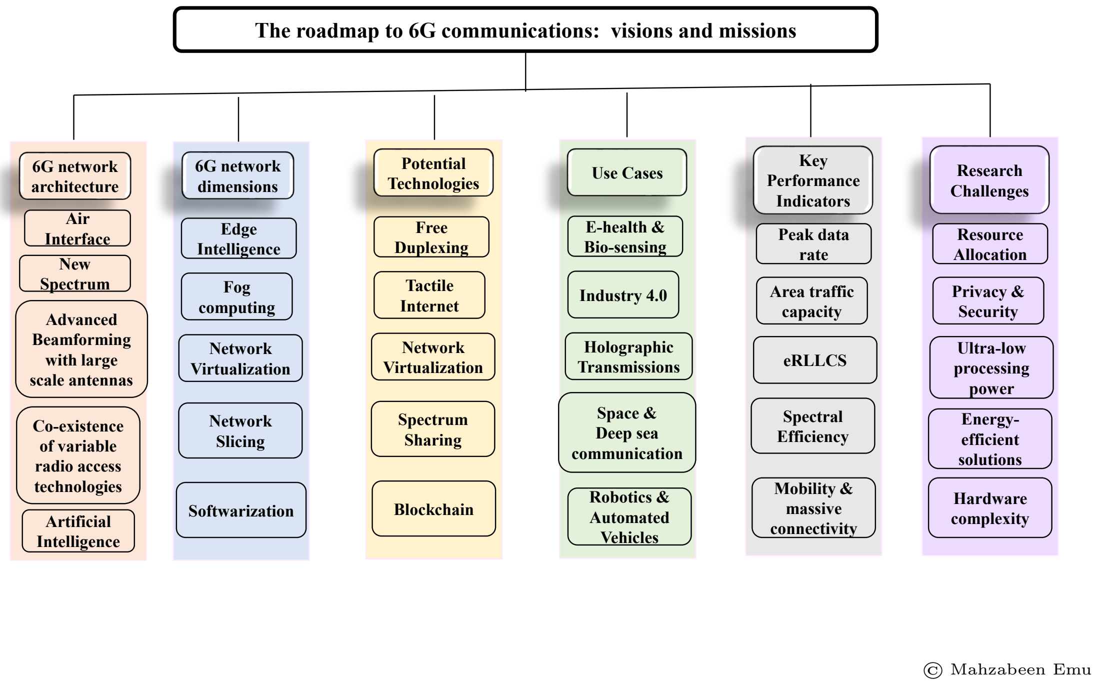


Figure 1.2: Different aspects of 6G communications

In the following, we outline some of the likely benefits of ML/DL aided network solutions:

- Identifying hidden and significant patterns of wireless networks;
- Improving key performance indicators (KPIs) compared to traditional approaches;
- Ensuring ultra-low response time with reasonable solution quality compared to state-of-the-art approaches;
- Resolving technical glitches associated with designing a complex mathematical model;
- Improving the flexibility, scalability, and adaptability of the algorithm.

With the integration of AI in future large-scale, multi-layered, high complex, dynamic, and heterogeneous networks, it is possible to secure diverse Quality-of-Service (QoS) requirements [9]. The smart agents learn to serve various level of services to different prioritized group of users (e.g., faster service for premium charged subscribers) without any form of functional constraints. Moreover, ML/DL algorithms can support low data rate, seamless connectivity, high throughput, and better resource utilization [9]. The reason being that ML/DL are able to assist optimized decision making and certain localization tasks based on network behaviour activity and traffic pattern. Thus, the zero-touch operation and control in future networks inhibit ultimate cognition to alleviate different issues (e.g., extremely high time complexity) with traditional mathematical programming models by initiating prompt response to service calls [3] [9].

As shown in Figure 1.3, the overall AI-enabled network functions in 6G can be categorized into four sections: intelligent sensing layer, data mining and analytics layer, intelligent control layer, and smart application layer [3]. In this thesis, we explore and demonstrate the potential applicability of ML/DL algorithms in control layer of next-generation IoT network management optimization problems.

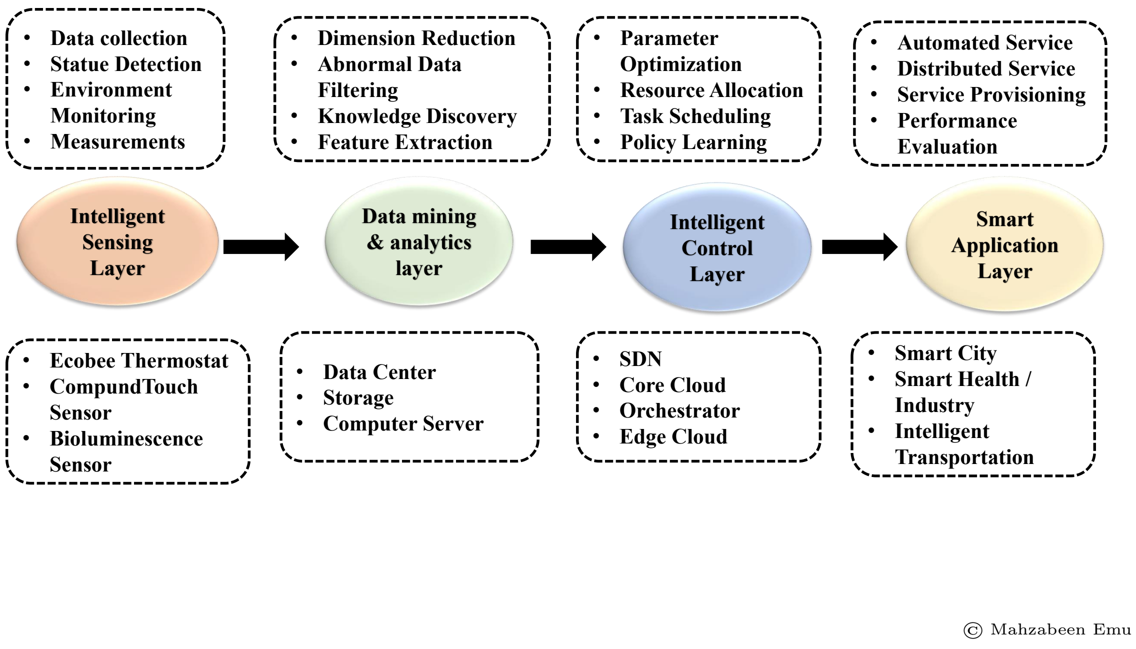
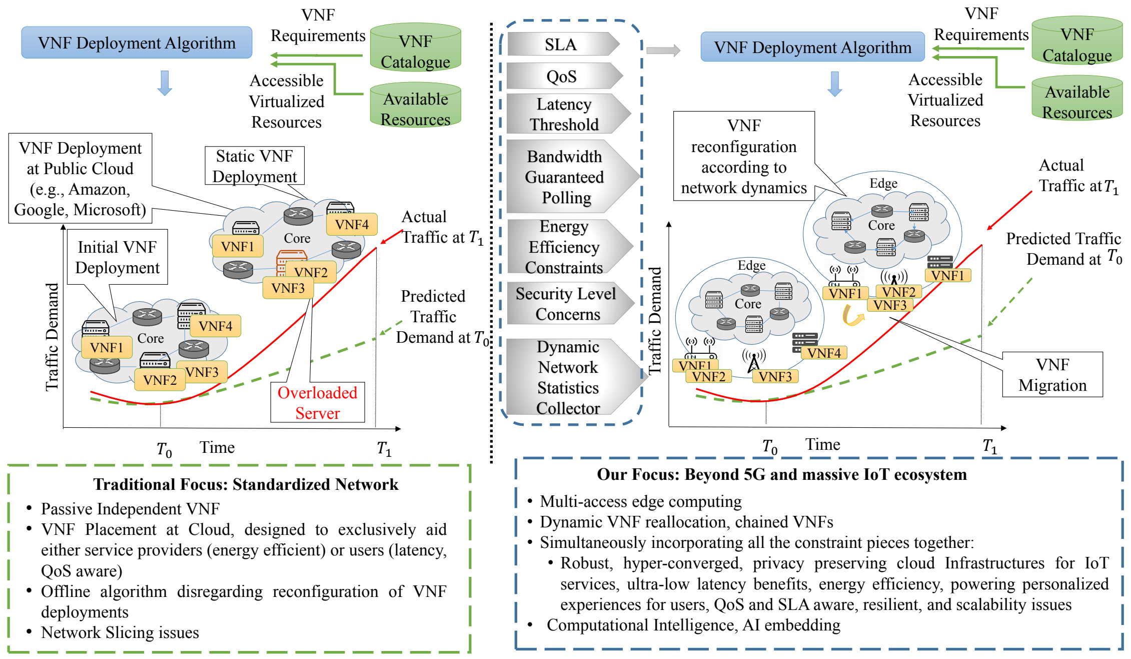


Figure 1.3: AI-enabled functions in 6G communications

We mainly focus on the network management system in Network Function Virtualization (NFV) [10] context. The reason being that traditional network functions (NF) are not suitable for next-generation communication industry due to various reasons [11]. In today's world, the integration of new NF services requires the installation of proprietary hardware devices into system infrastructure. These specialized hardware appliances are not straightforward to modify for supporting new services.

This static approach in service management involves several drawbacks, such as high capital and operational expenditure, poor resource utilization, and limited innovation of new services [12]. Hence, European Telecommunications Standards Institute (ETSI) has enabled the notion to run virtual network functions (VNFs) as cloud services [13]. VNFs are implemented as software modules by completely removing the dependency on the underlying hardware. Due to the virtualized form of NFs, the consolidation of numerous network appliances over conventional high volume servers, storage, and switches are indeed plausible. On top of it, NFV, along with Software-Defined Networking (SDN) [14], can serve the on-demand deployment of services at any point in the infrastructure, while being optimized over time to facilitate emerging business case demands. Multi-access Edge Computing (MEC) [15] expedites the NFV framework by offering a cloud computing environment at the network edge for content providers and service developers. With the deployment of VNF services at user edge premises, core networks are spared from congestion and enabled to serve other interconnected backbone services. In further chapters, we explain all these network key topics more elaborately.

All of the above-mentioned wireless technologies can collaboratively open a new horizon for business segments and enterprise customers. Network users request for a specific or chained series of VNFs to receive different virtual services. The accommodation of requested VNF to optimal hosts/devices is known as VNF allocation problem [16]. The orchestration of VNFs may consider different network resource requirements (e.g., bandwidth, resource, and latency). Hence, VNF resource allocation management is a very timely research problem in both academia and industry. Conventionally, VNF deployment researches are mostly for static environment and cloud-centeric [17]. Moreover, the predicted and actual traffic can vary a lot that disrupts the performance of an offline orchestrator and overload some servers. Our research focus has been more inclined towards futuristic network management to deal with various network issues (e.g., resource utilization, ultra-low latency) in dynamic IoT ecosystem. The conventional mathematical modeling and heuristics have very high computation complexity that are not suitable for delay-sensitive prospective IoT services. A pre-trained ML/DL model can significantly save the running time expenses for prospective communication networks [18]. Offline learning can spare the training time for servers/devices by only causing inference (prediction) time in action for orchestrators. Thus, our research outcomes are expected to provide better resource management and near-optimal end-to-end (E2E) latency in a significantly reduced response time by introducing computational intelligence. Figure 1.4 illustrates the research gap between traditional and our research focus. The rest of the chapters are organized as follows.


VNF Deployment Algorithm
VNF Requirements
VNF Catalogue
Accessible Virtualized Resources
Available Resources
VNF Deployment at Public Cloud (e.g., Amazon, Google, Microsoft)
Static VNF Deployment
Initial VNF Deployment
Core
VNF1
VNF2
VNF3
VNF4
Actual Traffic at T_1
Predicted Traffic Demand at T_0
Overloaded Server
Traffic Demand
Time
T_0
T_1
SLA
QoS
Latency Threshold
Bandwidth Guaranteed Polling
Energy Efficiency Constraints
Security Level Concerns
Dynamic Network Statistics Collector
VNF reconfiguration according to network dynamics
Edge
VNF Migration

Traditional Focus: Standardized Network

- Passive Independent VNF
- VNF Placement at Cloud, designed to exclusively aid either service providers (energy efficient) or users (latency, QoS aware)
- Offline algorithm disregarding reconfiguration of VNF deployments
- Network Slicing issues

Our Focus: Beyond 5G and massive IoT ecosystem

- Multi-access edge computing
- Dynamic VNF reallocation, chained VNFs
- Simultaneously incorporating all the constraint pieces together:
  - Robust, hyper-converged, privacy preserving cloud Infrastructures for IoT services, ultra-low latency benefits, energy efficiency, powering personalized experiences for users, QoS and SLA aware, resilient, and scalability issues
- Computational Intelligence, AI embedding

© Mahzabeen Emu

Figure 1.4: Divergence of research focus

Chapter 2 provides an overview of fundamental concepts, terminologies, and methodologies used in this thesis.

Chapter 3 considers the research problem to deploy standalone VNF at edge devices in IoT environment. The optimal placement of these VNFs at edge IoT devices in synergy with network attributes (e.g., latency on links) is a very challenging network management problem. In a large-scale real world network topology of hosts, the latency optimal VNF placement can radically enhance user experience for low-latency critical services. The unexpected latency violation in such services can degrade the user-enteric network performance metrics, even disrupt services in the worst cases. Example applications can include low-latency video encoders and content caches, personalised firewalls, web and P2P index engines, security functions, and tactile Internet. Thus, in this chapter, we solely focus on minimizing the overall latency of the network in a static environment. Since, the exact state-space size (number of IoT hosts) is massive in such problems, reinforcement learning is prone to exhibit inferior performance. Therefore, we propose an Artificial Neural Networks (ANN) based solution to predict the optimal hosts for VNFs. Simulation results demonstrate that our proposed methodology can ensure near-optimal latency with significantly reduced service response time, unlike traditional approaches. Additionally, we demonstrate that graphics processing unit (GPU) processed ANN can further expedite service response times, suitable for real-time IoT applications.

Chapter 4 extends the work of the previous chapter by simultaneously optimizing communication costs and relocation costs of a VNF in a mobile environment. A static orchestrator continuously has to monitor the dynamic network parameters and accommodate VNF placements accordingly. The static orchestrator may induce significant amount of VNF migration costs to maintain optimal E2E latency. The migration events are quite likely to happen every now and then, as the users are expected to constantly move due to small cell sizes considered in next-generation networks. This work aims to find the optimal new (migrated) hosts for VNFs such that the added relocation and communication costs are minimized. Hence, this research work focuses to eliminate the drawbacks mentioned with any static orchestrator beforehand. We suggest the use of ensemble deep learning approaches to solve this problem and demonstrate the performance aptitude through various KPIs. Through experimental results, it has been established that ensembling approach rather than using a standalone deep learning model improves the overall performance significantly. Finally, we strive to validate the generalization capabilities of the proposed models via performance verification in a real-word topology.

Chapter 5 introduces the notion of chaining multiple VNFs together to offer a particular service, unlike previous chapters. The delivery of Value-Added Services (VAS) depends greatly on actualization of the service function chaining (SFC) [19]. For instance, SFC can immunize cryptocurrency trading platforms by inserting firewall, even though no firewall record is to be found in the routing table perspective from one network infrastructure point to another. SFC is also very important for steering customized traffic for a specific group of users/applications [20]. To improve resource utilization, we enable the scheme of sharing the resources of already onboarded VNFs for satisfying a SFC request. The shared resources to serve multiple services, rather than initiating a new VNF instance can tremendously aid the cause of improving resource utilities. We also encourage subsequent VNFs in a requested chain to be allocated in consecutive nodes of IoT substrate network for preserving latency optimality as well. Due to the significant reduction in state space (feasible solution region), we propose and employ deep Q-learning for this particular problem. Then, we equip the learning process with Convolutional Neural Networks (CNN) for better convergence. In the previous chapters, we have avoided reinforcement learning due to the explosive state-space nature of the aforementioned problems in previous chapters. Subsequently, we establish that our proposed intelligent SFC orchestrator is suitable for resource-hungry and real-time IoT use cases.

Chapter 6 summarizes the research efforts conducted in this thesis and sheds light on the potential future works. This chapter also explains how researchers can adapt DL approaches for similar research enigmas and their related potential challenges.

Chapter 2

Background

This chapter includes a brief review of trending wireless network technologies, optimization criterion, and methodologies that have been considered for the rest of this thesis.

2.1 Virtual Network Functions

The trend of prospective telecommunication networks strengthening on SDN [21] is an initiative that expedites the network to be rationally administered utilizing software applications. NFV is one of the most popular kinds of technologies consolidated by SDN [22]. NFV is a paradigm that abstracts network services, which eliminates the requirement for proprietary, traditional dedicated hardware devices for each service [22]. The VNFs (e.g., routing, firewalls, deep packet inspection, load balancers, and intrusion security) can run on commodity hardware after packaging them together as virtual machines (VMs) for a group of users [21]. This concept ensures cost-effectiveness, flexibility, scalability, and more efficiency, along with excluding the concern of hardware limitations and truck rolls. Due to the virtualization, network providers hold the flexibility to move the VNFs across various servers according to the continually changing conditions in the network. Upon the request of a new network service function from a customer, providers create a new VM to manage the request. Once the service function is no longer required, it can be easily terminated. VNF deployments decrease capital expenses and operational expenditure by efficiently deploying new services and managing existing ones [23].

2.2 Multi-access Edge Computing

MEC offers to process, store, and compute data at edge devices that are close to users and data sources, rather than entirely depending on the cloud data centers [21]. These edge devices can be the home router, network gateways, routing switches, next-generation base stations, and integrated access devices, which reduce the obligation of the data to be traversed through the cloud data centers back and forth [24] [25]. The response time of services to the end users and unnecessary utilization of core networks can be reduced to a great extent in this way, however the cloud continues to persist.

IoT has already gained much attention due to the explosion of traffic and expansion of interconnected IoT devices rapidly [26]. The IoT services, different from traditional network services, expect to support automated provisioning of service composition along with real-time VNF deployment according to the requirement of users [22]. These service functions demand for a flexible and efficient placement mechanism that can handle constantly changing network dynamics (i.e., latency) and place VNFs at closer proximity to users supported by MEC [26] [25]. Latency on the links of the networks may continually fluctuate due to various reasons, such as traffic congestion, user mobility, and weather [22]. In the context of IoT, if the latency of the overall network goes beyond a certain limit, it may disrupt the network services and decrease the performance of the overall network leading to poor user experience and low QoS [26].

2.3 Service Function Chaining

SFC [27] can utilize SDN capabilities to form a service chain of interconnected network services (e.g., firewalls, network address translation, video optimizer, intrusion security, and parental control) and combines them in a virtual chain as shown in the Figure 2.1. SFC is considered to be operationally profitable by expediting automated provisioning of network applications and demands, hence enhancing the overall performance of the applications [27] [28]. This method guarantees that particular applications are provided with precise amount of network properties or resources (encryption, bandwidth, and QoS), which ends up optimizing the usage of network resources. For each service chain, it is necessary to support the desired QoS level. Otherwise, upon incompetency to do so, the service level agreement gets disrupted, which incurs unsatisfactory experience for users and non-negligible penalties for network providers.

2.4 Traditional Mathematical Programming Model

One of the most conventional way to design any resource allocation problem in the literature is known as Linear Programming (LP) and Integer Linear programming (ILP) [29]. LP [30] is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. Any LP formulation consists of the three key points: variable(s), objective function(s), and constraint(s). The generalized canonical form is represented as follows:

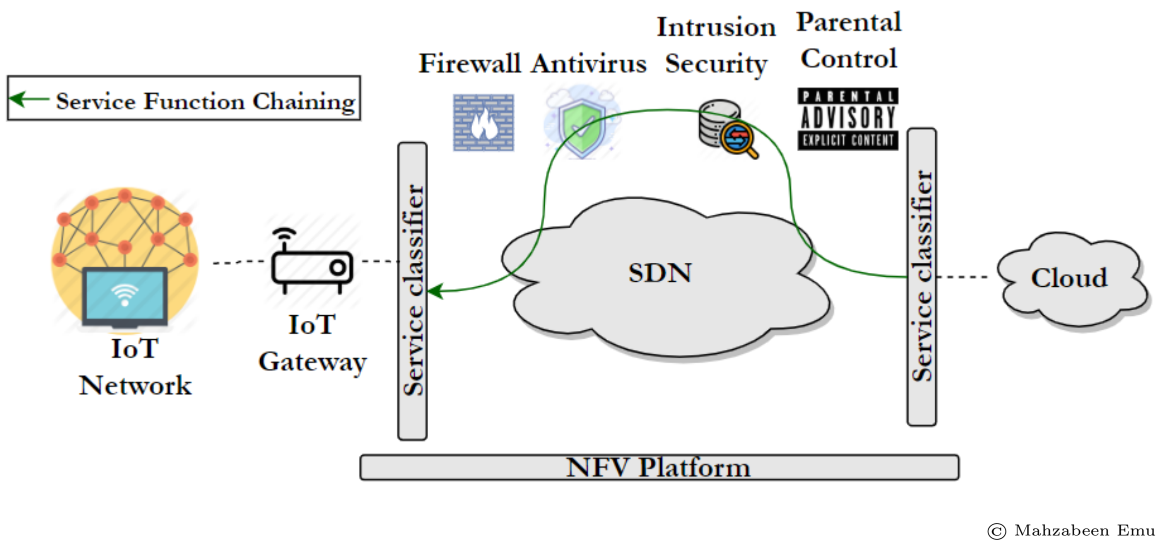


Figure 2.1: SFC (intrusion security, antivirus) by leveraging SDN capabilities in IoT context

$$maximize \sum c^T x \tag{2.1}$$

The objective equation above is subject to the following constraint:

$$Ax \leq b \tag{2.2}$$

here, x is a variable. Any particular choice for the values of x (not necessarily optimal) is known as a solution. A solution that satisfies all of the constraints is considered as a feasible solution. Yet, a feasible solution might not maximize the objective function. The solution that maximizes the objective function is regarded as a optimal solution.

An ILP problem is a mathematical optimization or feasibility program in which some or all of the variables are restricted to be integers. In that case, the second constraint of the above-mentioned LP problem can be transformed as $x \in \{0, 1\}$. ILP is a widely applicable problem-solving model in both academia and industry. Fast commercial solvers are available for use to solve ILP models, such as, CPLEX [31], OSL [32], GUROBI [33]. Moreover, powerful binding languages (AMPL, GAMS, PYTHON) exist as well in today's world [34].

Most of the literature choose ILP to formulate and solve VNF/SFC allocation problems for an optimal solution. However, there are some limitations to this approach, especially when applied in IoT ecosystem. The potential high-dimensionality of decision variables in IoT framework can render unreasonably high running time. Thus, the performance is most likely to be unacceptable due to the uninterrupted growth of IoT devices. The combinatorial explosion is the main drawback of this approach, as the running time grows exponentially with the problem size. Moreover, critical mathematical modelling can lead to various technical issues in practical cases. In this thesis, we aim to leverage ILP primarily for the following two reasons:

- To build labeled datasets with optimal solutions retrieved by ILP for training purpose, since there are no standardized datasets available in this research area
- For finding the optimality gap of our proposed AI based solutions and demonstrate their efficacy in the simulation environment

2.5 Meta-heuristics

Meta-heuristics [35] methods can not guarantee an optimal solution, unlike ILP. These methods are popularly utilized to invade the search space effectively for generating sub-optimal solutions within polynomial time. Moreover, these are problem independent methods, and the master strategy is relatively easy to adapt according to other heuristics. There are many meta-heuristics approaches that are inspired from the natural collective behavior of insects or animals.

Ant colony optimization (ACO) [36] is one of such algorithms that is known to able to find "good enough" solutions in relatively "small enough" computing time. The ACO method imitates the natural collective behavior of live ant colonies originating from a branch of the Swarm Intelligence (SI) [37] techniques. This algorithm considers a set of virtual agents/ants. Each of these ants retains a small amount of memory. These ants individually strive to produce their own solution depending on heuristic values. Later, the ants aim to enhance the quality of their solutions by information interchange through pheromone trails. Once the ants produce a locally optimal solution, they update their respective local pheromone trail values. Eventually, all ants generate a globally optimal solution by combining their individually built local optimal solutions.

The running time complexity of ACO is known to be quadratic [38]. Although these meta-heuristics strategies are not subject to exponential running time, they require extensive and large number of hyperparameter tuning [39]. Moreover, the quality of solution may vary significantly due to poorly chosen hyperparameters.

Hence, neural networks come to the rescue with more generalized hyperparameter optimization maneuvering.

2.6 Neural Networks for VNF Allocation

Deep learning has proven its potential through the different assignment and prediction based problems existing in the literature by providing a smart, holistic, and expeditious solution [40]. For cloud resource allocation and scheduling purpose, ANN has been used by combining the stochastic state transition and load prediction, while supporting expected performance levels [41] [42]. Subramanya et al. also discuss that machine learning can achieve promising results for different placement problems based on quantitative results [24]. An energy saving method with deep reinforcement learning has been approached to reduce power costs significantly without negotiating production for industrial facility [43]. This is very similiar to the service function chaining creation problem, which can be approached through deep reinforcement learning. Moreover, online caching prediction for edge computing using bidirectional deep recurrent neural network has performed remarkably well, which verifies the applicability of this approach through edge computation [44]. Deep neural networks over distributed infrastructures of computing hierarchies (e.g., the cloud, edge and end hosting devices) have reduced communication costs about 20 times compared to the traditional way of offloading sensor data in the cloud [45]. The creation of dynamic service chaining have been attempted through reinforcement learning by forecasting the consumption of physical and virtual resources (e.g., memory, CPU, and usage of service functions) in the NFV environment [46]. Another study on VNF service chaining claim that accelerated reinforcement learning performs remarkably well (ten times better in terms of cost efficiency) than the conventional reinforcement learning by monitoring and adapting environmental diversity continually [47].

Therefore, it is envisioned that an intelligent VNF orchestrator can serve the IoT services that demand fast response time (less delay or latency), resource utilization efficiency, and desired QoS. In this thesis, we aim to investigate how to leverage different deep learning techniques for the VNF and SFC orchestration to aid both providers and users in case of delay-sensitive massive IoT services. Hence, some of the commonly practiced neural networks and deep learning techniques can be utilized for the VNF orchestration and management of service chaining purposes in an automated manner.

2.6.1 Artificial Neural Networks

ANN is a popular approach from AI, which is a collection of artificial neurons that learn through training or experiences, analogous to the human brain [41]. The neurons or nodes of one layer in this network are connected to another layer through channels having some weights. The inputs are transferred to the next layer by passing through some activation function [48]. There are different kinds of activation functions mentioned in the following, for any input x:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

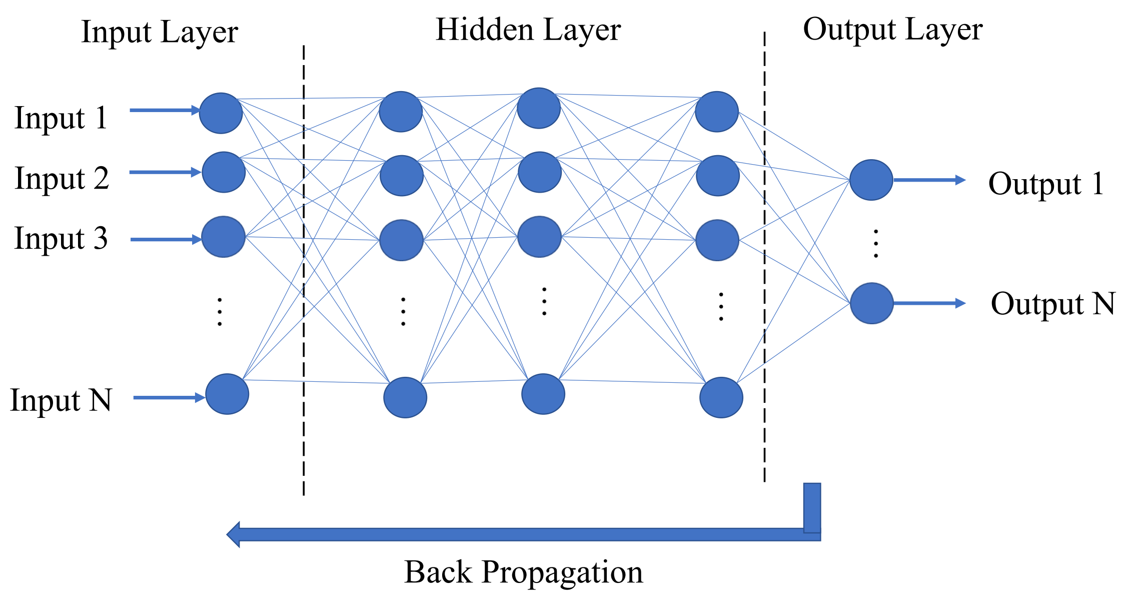
$$ReLU(x) = maximum(0, x) \tag{2.5}$$

The activation functions are introduced to bring non-linearity in ANN. Almost all the real-world cases have non-linear signal/pattern. Different algorithms are used to learn these weights of different channels. Gradient descent [49] based optimization is very popular among them. The weights of the channels are adjusted through backpropagation (BP) [50] until the end of the training process. The training time complexity of a typical ANN is polynomial [41]. Once the weights are learnt, the inference time of a pre-trained model grows linearly with the increasing number of hidden layers [41]. At last, the weights contribute to the final output that is calculated in the following way:

$$o = activation(\sum_{i=0}^{n} w_i x_i + b) \tag{2.6}$$

here, w_i is the weight for input x_i; o denotes the output; b is bias; $activation(\cdot)$ represents the activation function. A typical high level structure of ANN has been illustrated in Figure 2.2.

An ANN can be fed with given and derived input features set, such as latency characteristics, bandwidth requirements, locations of hosting devices and users, timestamped data of energy consumption, optimal placement decisions for previous scenarios, and chaining sequence for different services, etc. After a definite training phase, the machine can make placements and chaining arrangements accordingly, while obeying the latency threshold and other constraints for an unseen scenario. In conjunction,



© Mahzabeen Emu

Figure 2.2: A typical high level structure of ANN.

by utilizing the prediction ability, the trained model can select such placements and chaining that will induce fewer migration costs of VNFs in the dynamic network as much as possible.

2.6.2 Convolutional Neural Network

CNN [51] was originally developed for image classification with 2D pixels input representations for feature learning. However, one-dimensional CNN can be also used to analyze sequential signal data. 1-D CNNs are particularly known to be useful for extracting insightful features from fixed length (shorter length) of overall datasets, especially when the locations of features are not of high relevance [52]. Hence, unlike image analysis, 1-D CNN are more suitable for communication feature analysis. Moreover, a major difference between 2D and 1D CNN is the computational burden. The significant less computational complexity make 1D CNN more approachable for real-time and lightweight services. The training time complexity for 1D CNN is polynomial, while the prediction time complexity is known as linear [53]. Typically, CNNs are formed using convolution layers, pooling layers, and fully connected layers [54]. CNN is more towards regularization rather than being entirely fully connected, which reduces the chances of overfitting and the number of parameters significantly unlike deep neural networks. Pooling operation is performed in between two convolutional layers, where the number of features get reduced by sampling the from the convolution layer in a forward looking manner. Table 2.1 highlights the basic layer operations for feature extraction and classification with 1D CNN:

Table 2.1: Basic operations of a 1D CNN network

Layer	**Function of this layer**
Convolution layer	Generates feature/activation map
Pooling	Down-sampling operation to preserve detected features
Flattening	Prepares features for fully connected layer
Fully Connected Layer	Optimize target scores

A CNN network can be trained to select edge devices for VNF placement carefully and plan scheduling of service function chaining by learning to predict the network and device characteristics and resources over the training period. The inputs, in this case, can be deployment specifications and network parameters. At the same time, the output can be the selected nearby hosting device to place the VNF and provisioned sequence of service chaining. The error that the model is expected to minimize over time can be the differences of the output obtained by CNN from optimal cases.

2.7 Deep Reinforcement Learning

Deep Q-learning (DQL) is one variety of model-free reinforcement learning that can be implemented to select the policy concerning all finite Markov Decision Process (MDP) [54]. The inputs of the deep Q-networks are states ($s \in S$), and the quality of the actions ($a \in A$) are basically the output. After the training phase, an agent can determine the optimal policy provided a state, while maximizing the total expected reward value. In order to do so, a Q-table ($Q : S \times A \rightarrow \mathbb{R}$) is maintained containing the reward associated with each state-action pairs. For the decision making process by agent, ϵ-greedy algorithm [54] is used. Firstly, a random number is generated between 0 and 1. In case, the random number is greater then ϵ, the agent tends to maximize the reward by selecting $\underset{a}{argmax} Q(s, a)$ as an action. Otherwise, the agent completely executes a random action form A to encourage exploration. Initially, the Q-table is randomly populated. Next, the update process of Q-tables are formulated as follows:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \underset{a'}{max} Q(s',a') - Q(s,a)\right] \tag{2.7}$$

where, s and a are the present state and action; s' and a' are the next state and action; r is the instant reward after executing a; α is the learning rate; γ denotes the discount factor. $\gamma = 0$ implies that the agent should focus only on instant rewards.

To further expedite the learning abilities, multi-layered neural networks are used as Q-tables in Deep Q-network (DQN) [43]. Suppose, θ are the parameters of DQN, and $Q_\theta(s)$ is the outcome of DQN based on state s. The target Q-value is formulated in the following for a MDP transition (s, a, r, s'):

$$y(s,a) = r + \gamma \underset{a'}{max} Q_{\theta'}(s',a') \tag{2.8}$$

here, θ' is a historical memory copy of θ to prevent oscillations during the training phase. In this case, a gradient descent-based optimizer is used to update the parameters θ through minimizing the following loss function:

$$L_\theta = [y(s,a) - Q_\theta(s,a)]^2 \tag{2.9}$$

Overall, an agent can be continuously trained by perceiving information through its environment and be able to take satisfactory action from continuous action space (VNF placement and chaining decisions). This mechanism can evolve the learning abilities of the agent over time with the intent of improvising policies. The reward in such scenario can be considered as minimizing the overall latency, resource consumption, and migration costs. The inference time complexity of most of the deep reinforcement learning is linear [55]. The summarization of all the discussed methods have been provided in the Table 2.2 with their potential strengths based on VNF orchestration and SFC generation problem.

2.8 IoT Networks in AI-aided NFV Context

All of the major vendors of IoT application platform (Azure IoT, Amazon Web Services IoT, and Google Cloud IoT) are incorporating AI solutions (machine learning

Table 2.2: Summarization of commonly practiced DL approaches

Deep Learning Method	**Potential Strength**	**Some improvement scopes in the architecture**
Deep Q-Networks [54]	Combine the robustness of both supervised and unsupervised learning	Double Q-learning, dueling, and prioritized experience replay [54]
ANN [54]	Recognize patterns by quick formulation and analytical abilities	Dynamic layer widths selection for model calibration [54]
CNN [54]	Less chances of overfitting, progressive reduction in the spatial size of filters, and inclined towards regularization	Down-sampling transition and incremental feature construction for model simplification [54]

or deep learning based analytics) to wring pattern and insights from data in a faster way for different service purposes [54]. An efficient automated VNF and SFC orchestrator can induce a significant improvement in the overall performance of various IoT services. Figure 2.3 illustrates the high level concept of VNF placement at edge server and cloud, while generating different service chains to provide specific service for various IoT domains. However, various IoT domain has different kinds of specialized application demands that are need to addressed with AI integration. These aspects have been carefully taken care of while designing the service management orchestrators and selecting KPIs in the rest of chapters.

2.8.1 Powering Personalized Experiences for IoT Devices

IoT personalization is an obligation for the vendors to sustain in the perpetually competitive industry [56]. Many IoT consumer products (e.g., Google Glass, FitBit) require the ability to capture the personal experiences of individual users to provide deep insights into their routine based on the historic statistical behavior [57]. Various wearable IoT devices, for example, a device to track patient's vitals in the healthcare domain, demand expert prediction ability. This can be achieved by the provision the VNF requirements and chaining at different IoT edge devices. For the training purpose to learn VNF orchestration and management strategies, data concerning the users personalized interaction with the devices can be utilized. Such VNF placement and service chaining schemes can level up the game for new age IoT devices to create more intuitive experiences for users, while increasing the sales for vendors and service providers as well.

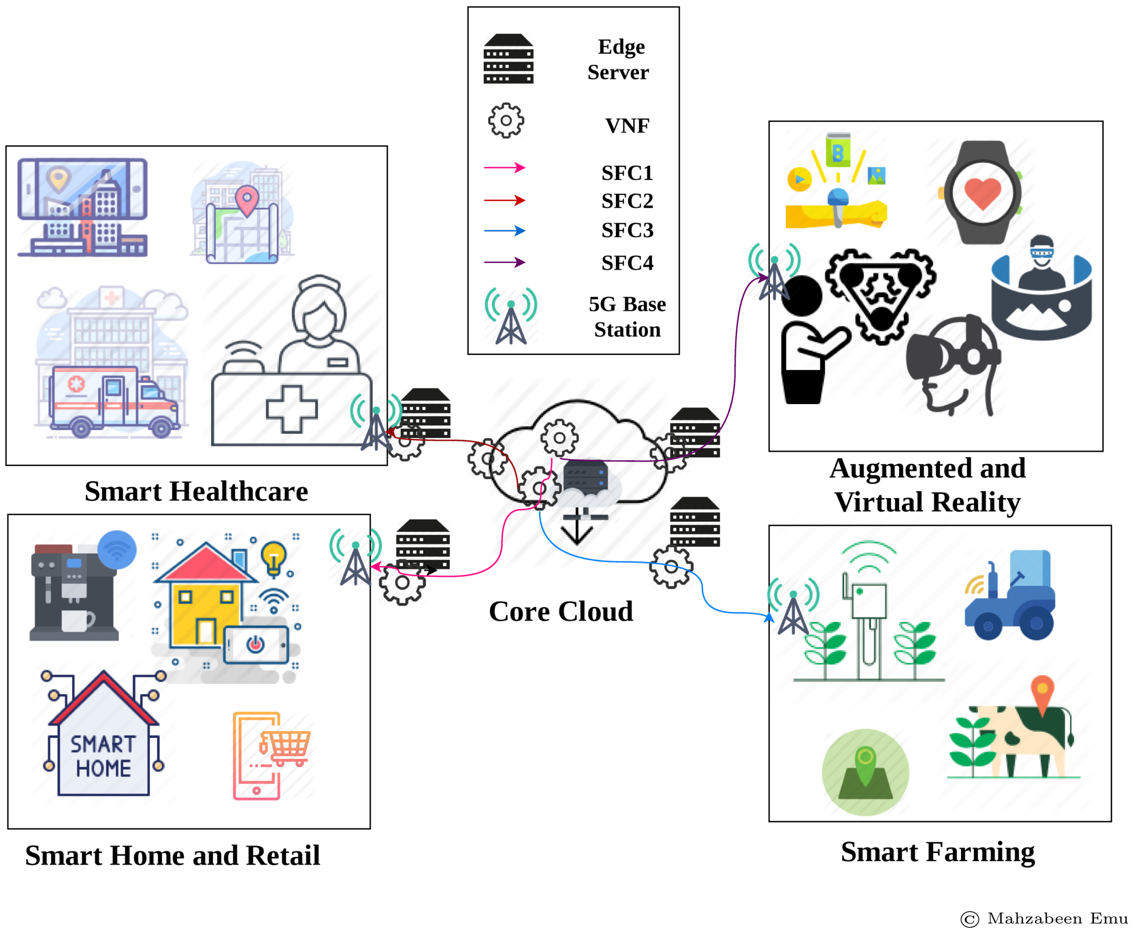


Figure 2.3: VNF and SFC deployment in context of various IoT domains

2.8.2 Robust and Privacy Preserving Cloud Infrastructures for IoT Services

Privacy is a primary concern and crucial requirement for any IoT enabled service [58]. IoT ecosystem can be a security threat because of storing users' personal data (e.g., arrival and leaving time at home or office, shopping details, health concerned data, and voice commands for home assistant) to the publicly exposed or private data centers [54]. These devices can control the environment of the users as well. By deploying some VNF at edge devices, it is possible to decline the obligation to upload or store personal data in the cloud.

2.8.3 Energy Efficiency and Ultra-low Latency Benefits

The careful selection for the training time (while charging the device or lying idle) and resources (over free WiFi) can reduce unnecessary power consumption to a great extent [57]. Therefore, the training of the model for orchestrating VNF and predicting service chain may occur when the device is idle to avoid any adverse impact on the device's performance. In such a case, the device predicts if a VNF requires to be placed in that particular device for current or near future usage. As the placement decisions are expected to be taken at the edge device, the latency experienced by the users of the IoT services will be much lower.

Energy efficiency has become a critical concern in today's world with the continuous expansion of data center traffic. Extensive power consumption induce escalating costs of ownership and carbon footprints as well [59]. It often becomes quite expensive for providers to pay high electricity costs due to the large volume of power dissipation from different computational infrastructures. Furthermore, to encourage environmental sustainability, carbon taxes have been introduced by some of the countries as well. Consequently, considering the energy efficient VNF placement and chaining can aid service providers to decrease their power (and carbon) costs, possibly to a great extent. Minimum power consumption and low latency are the key components for overall satisfactory and profitable encounters of applications in the IoT context. One of the key to enable low power consumption is to ensure balanced resource utilization [60].

2.8.4 Scalability Intelligence: From Micro Training to Macro Testing

The deep learning assisted VNF placement along with SFC generation techniques, require to be scalable in the sense that the model needs to maintain the performance in large scale scenarios, even after being trained with similar smaller cases [61]. Moreover, if any modifications that are required to be effected in the placement policies, a generalized update for the training model should be sufficient. Hence, scalability factor is a an important KPI to be assessed during any AI-aided VNF management solution.

2.8.5 Architecture Design: Emphasizing on the Selection of the Most Appropriate Model

A critical challenge is to select the most suitable design among various deep learning and neural network architectures [62]. It is further convincing that the integration of different learning strategy or individual techniques may be required to deliver the best performance for VNF orchestration and SFC. The inputs and outputs design must be carefully selected to ensure prediction capability. The higher number of parameters into the model or increasing the number of layers will eventually lead to better accuracy rates. However, significant number of factors considered into the input data is likely to increase the training and prediction time, which is not agreeable in the real-time VNF deployment and management context. Online training can easily capture the dynamic pattern of the environment [63]. Besides, offline training can be a solution to reduce the redundant communication overhead. Therefore, the design of the most appropriate AI technique for VNF placement and management strategy still requires much consideration, as it is not even easy for the expert researchers to always select the best serving fine-tuned model for various types of problems.

2.9 Summary

It is envisioned that VNF orchestration and SFC with fast response time, less power, and reduced migration costs conjointly can facilitate both the users and providers. However, due to the highly dynamic nature of the problem and many network factors or parameters being involved, traditional optimization approaches (i.e., ILP) are no longer suitable for the increasing number of time-sensitive massive IoT services (e.g., autonomous transportation, abstract virtualization, remote health monitoring and geographic information system). Hence, we propose the use of neural networks and deep learning techniques for VNF orchestration and SFC creation in the following chapters.

Chapter 3

Latency Aware VNF Deployment at Edge Devices for IoT Services

VNFs placed at the edge devices in the vicinity of users improve response time, avoid redundant utilization of core network, and reduce user-to-VNF end to end latency to a great extent. Different approaches for VNF placement have been proposed. However, the main concern has been to minimize the required number of servers to run VNFs for providing a specific service, without considering network conditions, for example, latency. In this chapter, we implement the optimal edge VNF placement problem as an Integer Linear Programming model that guarantees the minimum end to end latency, while ensuring Quality of Service by not overstepping beyond an acceptable limit of latency violation. Latency beyond such limits can be the cause of disruption and degradation of performance for time-sensitive IoT services. The time complexity of the existing optimal edge VNF placement algorithm is exponential. Thus, we further propose a VNF placement strategy using ANN trained by the assignment solutions generated from the ILP model for smaller instances of VNFs. This approach solves the VNF assignment problem at edge devices for a larger number of VNFs, while reducing the time complexity to be linear and providing similar results as the ILP model in terms of latency.

3.1 Introduction

The trend of future telecommunication networks building on SDN is an initiative that facilitates the network to be logically and centrally operated using software applications. This privileges network operators to maintain the extensive network notwithstanding of the underlying technology. Amongst the variety of technologies incorporated by SDN, network functions virtualization is a popular one [22]. NFV is a process to abstract network services that usually operate on traditional dedicated hardware, basically proprietary devices [22]. These VNFs such as firewalls, WAN accelerators, load balancers, deep packet inspection, and intrusion prevention, are allowed to be manipulated, regulated and placed on the software of different network nodes, diminishing the concern of hardware constraints [64]. NFV is a paradigm where the functionalities of the network are virtualized to be more efficient, cost-effective, scalable, and flexible. VNF deployments aid service providers by accelerating the deployment of new services, while managing the existing services in such an efficient way that significantly decreases capital expenditure and operational costs. With the exponential explosion of the IoT devices and growth of traffic, a network service provider can deploy new services for a specific group of users eliminating the need for truck rolls, hardware expenses by using this technology. Deploying VNFs at the IoT end devices efficiently to minimize user-to-VNF end-to-end latency (latency experienced between a user and a VNF) has been one of the key research concerns of this domain in recent times. One approach to fulfill the expectations of advancing networks to improve latency and deploying user-specific services in a notably efficient fashion can be considered as MEC [65]. The existence of cloud continues to persist, however, the edge devices process, compute and store the data instead. Therefore, an extensive variety of IoT applications for which the real-time response has to be strictly maintained, for example, augmented reality, autonomous vehicles, collaborative computing, and edge video caching can be supported by MEC [66]. The edge devices can be IoT gateways, sensors, actuators, and different IoT devices itself (i.e., FitBit, Smart Security Systems, Google Glass). The lightweight, docker containers can be used to deploy VNFs even on low-cost hosting edge IoT devices considering the conflicting dependencies [67]. Even though the docker container is much more preferable, for running services in the edge network, Virtual Machines can be delegated as well. While VNFs are being placed in closer proximity to end-user, the response time and extensive utilization of core networks can be reduced to a great extent, as it eliminates the obligation of personalized IoT data (e.g., mHealth related data) to traverse through the core network for providing a service [67]. As shown in Figure 3.1, the VNF orchestrator deploys and manages VNFs at edge devices so that the latency experienced between end users and VNF is minimized. Moreover, the

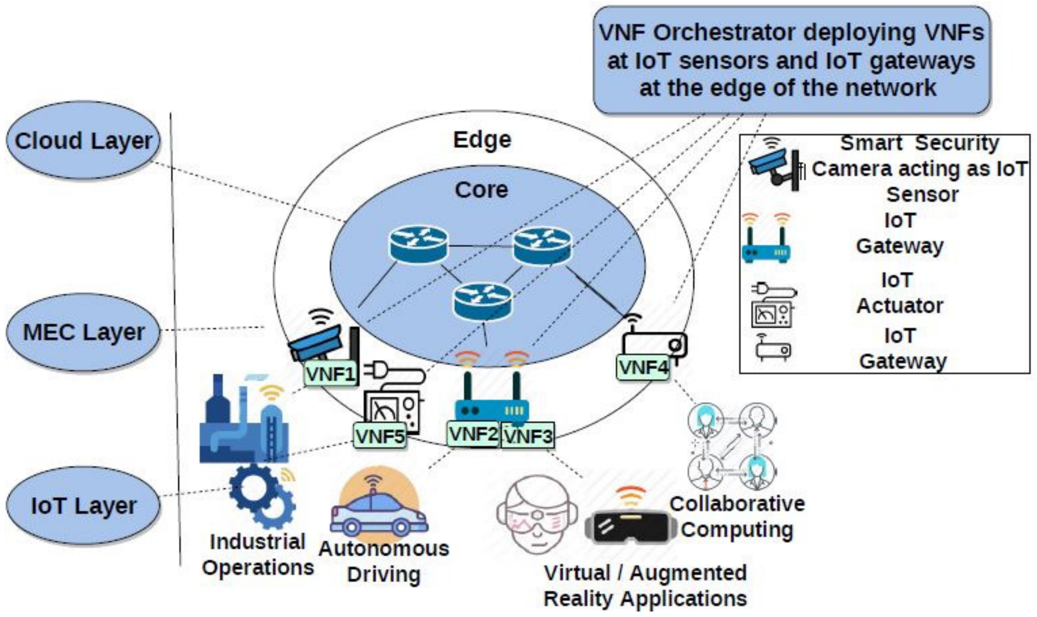

VNF Orchestrator deploying VNFs at IoT sensors and IoT gateways at the edge of the network
Cloud Layer
MEC Layer
IoT Layer
Edge
Core
VNF1
VNF2
VNF3
VNF4
VNF5
Smart Security Camera acting as IoT Sensor
IoT Gateway
IoT Actuator
IoT Gateway
Industrial Operations
Autonomous Driving
Virtual / Augmented Reality Applications
Collaborative Computing

Figure 3.1: A high level architecture of VNF Placement at IoT edge devices, managed by VNF orchestrator for latency critical IoT services.

orchestrator continuously monitors the dynamic network parameters and accommodates VNF placements accordingly. This is very obvious, as the users are mobile and expected to constantly move due to small cell sizes considered in next-generation networks. Latency on links keeps changing due to various factors other than the mobility of the user, such as weather, the configuration of hardware devices, and congestion in network as well [68]. The delay generated in a network, if somehow goes beyond a certain threshold may be the reason for the degradation of overall performance and disruption of network services.

To the best of our knowledge, no research work for placing VNFs based on the decision of ANN trained by optimal solutions has been approached before. We first train an artificial neural network with the VNF assignment results of optimal edge VNF placement and later test the artificial neural network-based placement approach using larger test instances having a large number of VNFs. In this chapter, we considered the optimal edge VNF placement strategy, which always ensures the total minimum latency from users to VNFs [69]. However, the ILP formulated placement strategy is NP-hard, which can be computationally expensive for large number of VNFs and hosts [70]. Hence, we propose an ANN-based method for assigning VNFs to edge devices, where the training phase involves optimal assignment solutions of different VNF and host pairs resulted from the ILP. The VNF placement via ANN algorithm exhibits promising results in terms of latency while reducing the time complexity to be linear compared to the optimal VNF placement method regardless of the size of training data samples.

3.2 Related Work

NFV originates with the transformation of approaching to the virtualization of network elements, which are presently stationed on hardware devices. The more networks become evolved towards NFV, the consideration of the data forward plane and control plane facilitate the management of existing and creation of new services.

Several studies to improvise the efficiency of this technology have been done, for example, task scheduling [71], allocation of VNFs [72], scaling [73], migration of VNFs [74], etc. Lately, a few additions have propelled the policy-aware traffic problem using hardware middle-boxes [75] [76]. Some research works involve in deploying and orchestrating VNFs [77] [78]. Nevertheless, the attention has been drawn towards architecture supporting and providing software operating middle-boxes, for example, NetVM [21] as the hardware capacity is not unlimited. Among different research works on this area, VNF allocation has gained much attention. Several approaches have been proposed for efficiently managing VNF assignments. A cut and solve based approach has been proposed to give a near-optimal solution [79]. Assigning VNFs to edge server rather than only depending on cloud centers can improve latency up to 70% using Integer Linear Programming [80]. Later, a solution to find the optimal time for VNF migration has been proposed while ensuring a certain level of QoS [69] [81]. A hybrid online algorithm aiming to minimize the error about prophesying the service chain requests and better competitive ratios are achieved with adaptive processing abilities [82]. A stable matching algorithm to reduce the run-time of the algorithm while introducing mixed integer programming to the problem definition and finding the near-optimal solution in terms of latency has been approached which resulted in lower time complexity in contrast to the optimal solution [83]. This prevented the failure of the model in such a case where a VNF can not be placed while trying to satisfy certain constraints such as staying below a certain threshold of latency.

In distinction to the previous works, our approach is to examine an optimized end-to-end latency problem overall. However, the need to reduce the time complexity of the optimization problem which is exponential in nature, we feed an Artificial Neural Network with different optimal assignment solutions so it can take better placement decisions in terms of latency while reducing the time complexity to be linear. Therefore, for a larger number of VNFs and hosts [70], the time complexity of assigning the VNFs to hosts will be much less than the optimal edge VNF assignment strategy, while ensuring latency values similiar as the optimal ones.

3.3 Optimal Edge VNF Placement

In this section, the VNF assignment problem is formulated as an ILP problem to find the assignments of VNF at edge devices in an optimal manner. The optimal Edge VNF placement strategy aims to find the latency optimal VNF assignments in edge devices or the distant cloud. To ensure optimal latency for users, network providers first aim to place the VNFs at edge devices closer to the user. However, in case the edge devices are out of capacity and fail to accommodate the VNFs, then the provider's internal cloud is used to host the VNFs.

3.3.1 System Model

In this chapter, we consider a set of users are connected to some hosting devices (IoT actuators, sensors, and gateways) through links within a network topology. Table 3.1 exhibits all the parameters considered for the formulation of the system model. We denote the physical network topology as an undirected graph $\mathbb{G} = \{\mathbb{H}, \mathbb{U}, \mathbb{E}\}$, where $\mathbb{H}$ and $\mathbb{U}$ are the set of hosting devices and users in the network topology connected by the set of links $\mathbb{E}$.

Table 3.1: Description of parameters for our system model

Parameters of Network	**Description**
$\mathbb{G} = \{\mathbb{H}, \mathbb{U}, \mathbb{E}\}$	Topology of the network
$\mathbb{H} = \{h_1, h_2, ..., h_H\}$	All the IoT edge devices defined as hosts available in network
$\mathbb{U} = \{u_1, u_2, ..., u_U\}$	All the users in the network
$\mathbb{E} = \{h_i u_j \mid h_i \in \mathbb{H}, u_j \in \mathbb{U}\}$	All the edges associated with hosts and users in the network
l_{ij}	Latency on the link $h_i u_j$
C_j	Hardware capacity of the host h_j
Q_{ij}	Capacity of the link $h_i u_j$
Parameters of VNF	**Description**
$\mathbb{N} = \{n_1, n_2, n_i, ..., n_N\}$	All network functions that are required to be assigned where $n_i \in N$
R_i	Host requirements of each VNF $n_i \in N$
θ_i	Maximum latency tolerance limit of each VNF $n_i \in N$
Decision Variable	**Description**
y_{ij}	Binary decision variable represents whether VNF n_i has been assigned to host h_j or not
Derived Parameter	**Description**
L_{ij}	Total latency between VNF n_i and host h_j
b_{ij}	Required bandwidth in-case VNF n_i is hosted at h_j

$\mathbb{N} = \{n_1, n_2, n_i, ..., n_N\}$ represents the set of network functions which are required to be assigned to different hosting devices. In practical settings, different types of VNFs have different hardware requirements to be placed on hosts. For example, firewalls demand less than deep packet inspection methods. Hence, we have introduced R_i to denote that requirements of each VNF n_i, and C_j to represent the hardware capacity of a host h_j.

All the VNFs along with computational requirements have a maximum tolerance limit of delay, denoted by θ_i according to the provider's Service Level Agreement (SLA). Each link in the network has a latency value of l_m. The total latency from a user to VNF is calculated by summing all the l_m of links between a VNF n_i and a host h_j, which is represented by L_{ij}. At last, y_{ij} is the key decision variable that describes if a VNF n_i has been assigned to a host h_j or not.

3.3.2 Problem Formulation

Given, a set of VNFs $\mathbb{N}$, a set of host devices $\mathbb{H}$, and a set of users $\mathbb{U}$ along with the latency matrix l_{ij}, the objective of this chapter is to find the optimal assignments of the VNFs at different host devices so that the latency experienced between user and the VNF is minimized. Mathematically, the objective function of this problem formulated using ILP can be represented by the following equation:

$$minimize \sum_{n_i \in \mathbb{N}} \sum_{h_j \in \mathbb{H}} y_{ij} \times L_{ij} \tag{3.1}$$

The object function finds the values of y_{ij}, whilst it is subject to the following constraints:

$$\mathcal{C}1 : \sum_{n_i \in \mathbb{N}} y_{ij} \times R_i \leq C_j, \forall_{h_j \in \mathbb{H}} \tag{3.2}$$

$$\mathcal{C}2 : \sum_{h_j \in \mathbb{H}} y_{ij} \times L_{ij} \leq \theta_i, \forall_{n_i \in \mathbb{N}} \tag{3.3}$$

$$\mathcal{C}3 : \sum_{h_j \in \mathbb{H}} y_{ij} = 1, \forall_{n_i \in \mathbb{N}} \tag{3.4}$$

$$\mathcal{C}4 : \sum_{h_j \in \mathbb{H}} y_{ij} \times b_{ij} \leq Q_{ij}, \forall_{n_i \in \mathbb{N}} \tag{3.5}$$

$$\mathcal{C}5 : y_{ij} \in \{0, 1\} \tag{3.6}$$

Hardware resources of each host are limited. Therefore, constraint $\mathcal{C}1$ represents that each device can host up to a certain number of VNFs until their capacity runs out. Constraint $\mathcal{C}2$ ensures that, while placing some latency-sensitive VNFs, their maximum tolerance delay threshold from the user is not violated. The third constraint $\mathcal{C}3$ secures a single assignment of a VNF to some host. Consequently, each VNF can be placed to exactly one of the hosts, which can be edge devices close to the user or the cloud. The fourth constraint $\mathcal{C}4$ ensures that any physical link should not be overloaded. Finally, $\mathcal{C}5$ represents that the value of decision variable y_{ij} will be either

0 or 1, in case n_i is hosted at h_j the value of y_{ij} will be 1, otherwise it will be 0.

3.4 VNF Placement Using ANN

In the worst case, the ILP model takes exponential time [84], which is inefficient while dealing with large number of VNFs for massive IoT services. An ANN has linear prediction time complexity, regarding the number of samples [85]. Therefore, training an ANN to predict the optimal VNF placement solution is promising in reducing the optimization time and improving the user experience. Since there are no standardized datasets for supervised learning in this research area, we propose to leverage the result of ILP to build labeled datasets. The labeled dataset contains VNF and optimal host pairs, with other network metrics as feature vectors.

We leverage the existing ILP solver to optimize the simulated VNF placement problems, and record the solutions to generate labeled data for training an ANN. Eventually, we train the ANN through the simulated datasets to imitate the performance of ILP. We have considered smaller instances for training the ANN, such as 10 - 45 number of hosts, and 100 - 450 number of VNFs. The details of the of the training data and simulation parameters are available in section V. If we represent the ANN as a function $\mathcal{F}$, then we have $\mathcal{F}(X_{ij}) = \hat{Y}_{ij}$, where $0 \leq \hat{Y}_{ij} \leq 1.0$. The training label for each sample X_{ij} is y_{ij}. We optimize the ANN parameters to minimize the mean squared error between y_{ij} and $\hat{Y}_{ij}$. The ANN aims to solve the VNF placement problem in a sequential style. We define the input features of the ANN as an octuple $X_{ij} = (\{C_k|1 \leq k \leq H\}, \{Q_{ik}|1 \leq k \leq H\}, \{L_{ik}|1 \leq k \leq H\}, L_{ij}, R_i, \theta_i, b_{ij}, Q_{ij})$, where i and j indicate the current VNF n_i and the host h_j respectively. Therefore, we interpret the output of $\mathcal{F}(X_{ij})$ as a confidence score of placing VNF n_i on host h_j. Algorithm 1 depicts the simulation and training process. The procedure of leveraging the trained ANN to optimize VNF placement is shown in Algorithm 2. Further details of the considered ANN architecture has been summarized in Table 3.2.

3.5 Experimental Results

We have implemented the optimal edge VNF placement and trained an ANN with the optimal VNF to host (edge devices available in the MEC or IoT layer) assignment solutions generated by the ILP solver. Later, we compare the performance of our proposed ANN placement strategy with the optimal edge VNF placement and

Algorithm 1: Simulation and Training

```
1  $\mathcal{D} \leftarrow \{\}$
2  for $epoch < total\ simulation\ epochs$ do
3      Generate a random system: $L_{ij}, b_{ij}, \theta_i, C_j, R_i, Q_{ij}$
4      $y_{ij} \leftarrow$ Solve the random system through the ILP optimizer
5      for $i = 1, 2, ..., N$ do
6          $j \leftarrow \arg\max_j y_{ij}$
7          $k \leftarrow$ get a random number from $\{1, 2, ..., m\} - \{j\}$
8          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(X_{ij}, 1), (X_{ik}, 0)\}$
9          $C_j \leftarrow C_j - R_i$
10         $Q_{ij} \leftarrow Q_{ij} - b_{ij}$
11 Normalize each feature in $X_{ij}$ where $(X_{ij}, y_{ij}) \in \mathcal{D}$ between 0 and 1.0
12 Train the ANN on every $(X_{ij}, y_{ij})$ pair in $\mathcal{D}$
```

Algorithm 2: VNF Placement via ANN

```
   Input: $L_{ij}, b_{ij}, \theta_i, C_j, R_i, Q_{ij}$
   Output: $S$
1  $S \leftarrow \{\}$
2  for $i = 1, 2, ..., n$ do
3      $M \leftarrow$ array of $H$ elements
4      for $j = 1, 2, ..., m$ do
5          if $L_{ij} > \theta i$ or $R_i > C_j$ or $b_{ij} > Q_{ij}$ then
6              $M_j \leftarrow 0$
7          else
8              $M_j \leftarrow \mathcal{F}(X_{ij})$
9      $k \leftarrow \arg\max_j M_j$ place $n_i$ on $h_k$
10     $C_k \leftarrow C_k - R_i$
11     $Q_{ik} \leftarrow Q_{ik} - b_{ik}$
```

Table 3.2: ANN architecture details

Operation/Layer	**Details**
Optimizer	SGD
Activation Function	ReLU
Number of hidden layers	3
Number of nodes in hidden layer	25

a greedy placement strategy based on latency and running time. The ANN implementation has been executed using python's TensorFlow packages. The greedy VNF assignment method always selects the best available VNF to host assignment at that moment which might be proven to be very bad in the long run in terms of latency. The greedy placement strategy from the set of candidate solutions picks the n_i and h_j pair with locally optimal latency L_{ij} at each stage. However, the greedy algorithm may fail to find the optimal solution, with the possibility of even offering the worst possible VNF to host assignment resulting in highly deviated total latency from the optimal case. Although the optimal edge VNF placement using ILP always ensures better latency benefits, for large number of VNFs and hosts [70], the time complexity remain exponential. On the contrary, VNF placement using the proposed ANN strategy reduces the time complexity to be linear, which reduces computational expenses at a significant rate for large scale VNFs to hosts assignments.

For running the simulations, we have used an INTEL® CORE™ CPU i9-7920X (12 cores at 2.9GHz to 4.4GHz) machine with NVIDIA® GTX1080Ti GPU and 128GB Memory. In the case of placement experimentation using ANN, we have run the placement algorithm with both two and four GPUs respectively. The performance of this proposed placement method can also be improved further by running it on different CPU threads in parallel. The latency violation limit of each VNF (θ_i) has been generated between $30 - 100$ milliseconds, as different IoT services may have different tolerable limits of latency. Similarly, the values of VNF requirements (R_i), capacity of hosts (C_j) in case VNF n_i is placed at h_j has been considered to be in the range of 10 - 50 and 10 - 800 respectively, due to dynamic capacity owning of different IoT devices. Moreover, the bandwidth requirement (b_{ij}) of a VNF n_i placed at host h_j can be 5 - 10, while the hosts have the bandwidth capacity (Q_{ij}) ranging between 50 - 100. The latency values on the links vary due to several reasons, hence L_{ij} values have been generated randomly to be varying from 5 to 100 milliseconds. All of the mentioned parameters have been adapted from the existing work in [83] [86].

Figure 3.2 illustrates the total latency deviation of greedy placement strategy and placement strategy using ANN, from the optimal edge placement with respect to the delay variation in percentage scale. In this case, latency deviation represents how

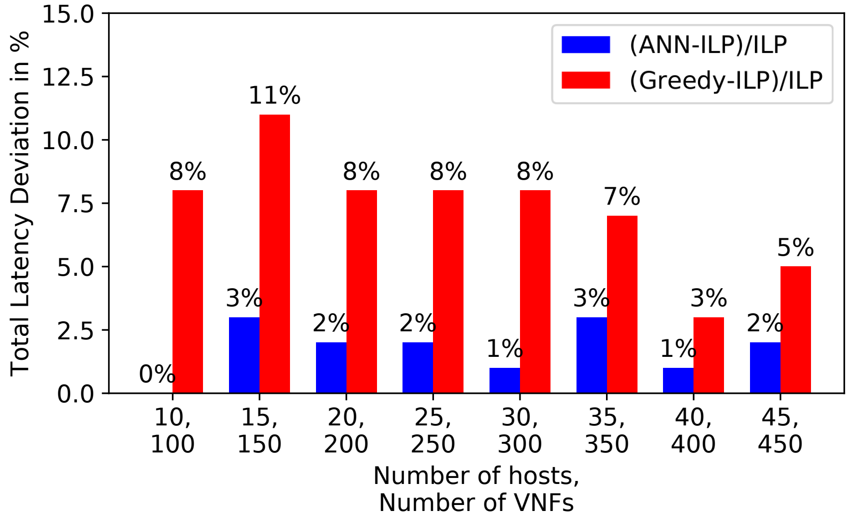
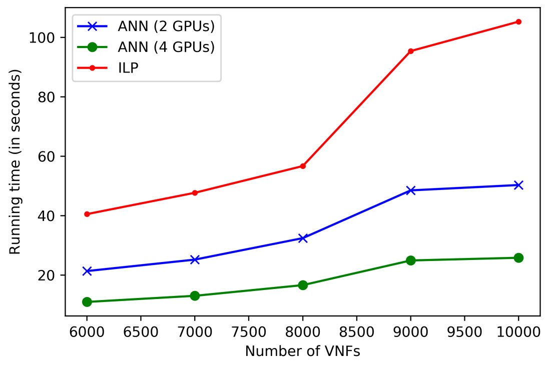


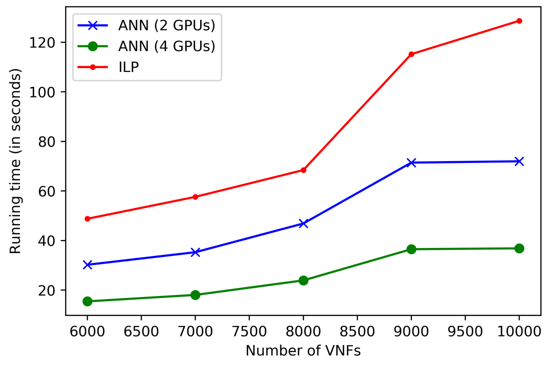
Figure 3.2: Comparison of the total latency deviation from ILP (Optimal edge VNF placement) in percentage achieved by ANN (VNF placement using ANN) and Greedy (Greedy placement strategy) for smaller instances of VNFs and hosts.

much the latency values of ANN and greedy placement mechanisms vary from the optimal latency value achieved by ILP. This experiment has been done with 10, 15, 20, 25, 30, 35, 40, and 45 number of hosts in combination with 100, 150, 200, 250, 300, 350, and 450 VNFs respectively. It is evident that ANN placement strategy exhibits promising total latency results resembling as the optimal, as the delay deviation from the optimal total latency can vary by 0% to 3% only, comparing to the optimal edge placement strategy. In contrast, the greedy placement approach increases the total latency deviation from optimal case by a maximum of 11%, while generating at least three times greater total latency than the VNF placement using ANN.

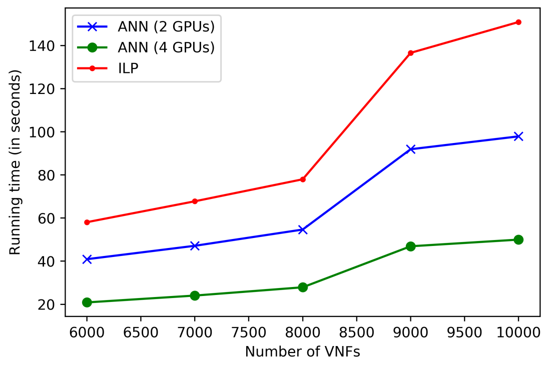
For the next experimentation, we have considered the host numbers to be 90, 110, 130, and 150 with the number of VNFs varying from 6000 to 10000. As shown in Figure 3.3a, 3.3b, 3.3c, and 3.3d the execution time for ANN placement strategy with two or four GPUs improve the running time by a significant amount (50% - 60% decrease of running time for ANN placement strategy using two GPUs and 65% - 85% in case of ANN placement strategy using four GPUs) comparing to the optimal



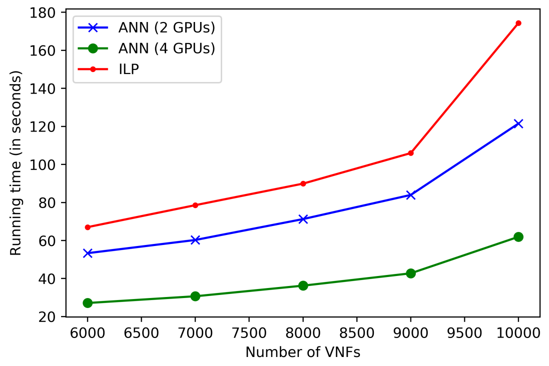
(a) Running time comparison for 90 hosts



(b) Running time comparison for 110 hosts



(c) Running time comparison for 130 hosts

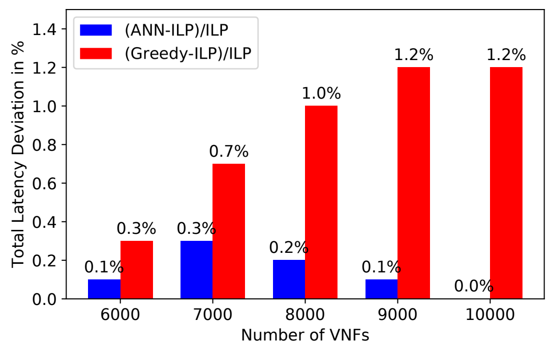


(d) Running time comparison for 150 hosts

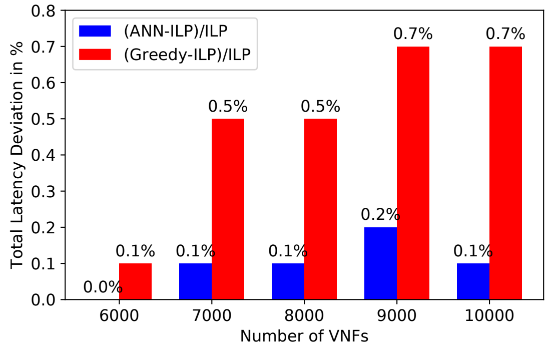
Figure 3.3: Comparison of the running time of different placement approaches for larger instances of VNFs varying from 6000 to 10000 and different number of hosts.

edge VNF placement with the growing number of VNFs.

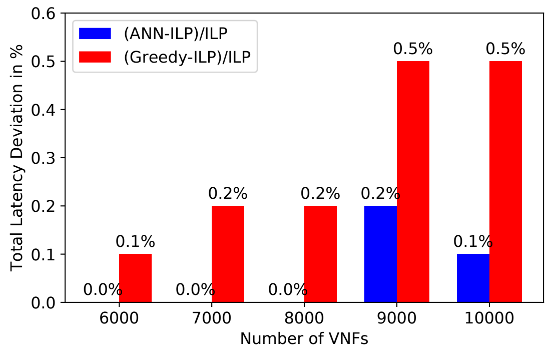
The trends in the increase of running time for optimal edge placement explain that it performs poor than VNF placement using ANN in terms of running time, while handling large scale VNF placement. Figure 3.4a, 3.4b, 3.4c, and 3.4d, demonstrate the total latency deviation of greedy placement strategy and placement strategy using ANN, from the optimal edge placement with respect to the delay variation in percentage scale. The (Greedy-ILP)/ILP and (ANN-ILP)/ILP represent the differences or deviations of latency values achieved by greedy placement strategy and ANN placement strategy from optimal latency values found by ILP in terms of percentage. This experiment has been done with 6000 to 10000 number of VNFs, which is significantly larger than the instance size of VNFs used in the training phase. It can be easily observed that the total latency occurred to place all the VNFs using ANN strategy gives very much reasonable cumulative latency results (0.0% - 0.3% deviated from



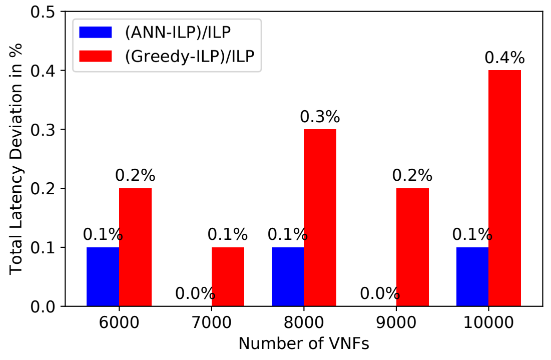
(a) Latency deviation comparison for 90 hosts



(b) Latency deviation comparison for 110 hosts



(c) Latency deviation comparison for 130 hosts



(d) Latency deviation comparison for 150 hosts

Figure 3.4: Comparison of the total latency deviation from ILP (Optimal edge VNF placement) in percentage achieved by ANN (VNF placement using ANN) and Greedy (Greedy placement strategy) for larger instances of VNFs varying from 6000 to 10000 and different number of hosts.

optimal only) in comparison to ILP solver. On the contrary, the greedy placement strategy can cause up to 1.2% deviated total latency from ILP, which is more than twice of total latency variation resulted from ANN. Therefore, with the growing number of VNFs, placement strategy using ANN delivers very similar total latency results as the optimal edge VNF placement strategy, while ensuring much shorter execution time by taking faster placement decisions.

3.6 Summary

In this chapter, our concern has been to design an alternative placement strategy expected to be faster than the ILP model of optimal edge placement. Since, in case of real-time IoT applications (e.g., Storage Incompatibility Detection, Perimeter Access Control, Forest Fire Detection, Smart Roads, Patients Surveillance, and Ultraviolet Radiation Detection), the VNF orchestrator is required to take VNF to host assignment decisions as quickly as possible. Therefore, we have considered the optimal edge VNF placement problem using Integer Linear Programming. To ensure optimum latency, the trade-off here is between time complexity and latency minimization, considering this placement strategy takes exponential time. Therefore, we have designed another placement strategy using ANN, where the training phase is completed based on the optimal VNF to host assignments generated by the ILP model of optimal edge VNF placement for smaller instances. This placement method is better considering the time complexity being linear, which reduces the computational complexity to a great extent when dealing with a large number of VNFs and hosts. Furthermore, this placement strategy shows promising results resembling the latency minimization capability of the optimal edge placement method. The total latency deviation of this strategy from the optimal case varies up to 3% for the training instances and up to 0.3% only in case of larger instances compared to the training phase, which is trivial. Moreover, this placement strategy decreases the running time of the optimal edge VNF placement up to 60% with two GPUs and 85% with four GPUs. The promising results in terms of reducing the running time of VNF orchestration and providing resembling optimal latency results, also indicate that artificial neural networks can be used for similar optimization research problems.

The main limitation of this study is that the orchestrator has been designed for a static environment. Thus, to keep up with the track of necessary VNF migrations, the orchestrator requires constant monitoring of various network dynamics. Moreover, the placement decisions have to be re-evaluated every now and then for keeping the overall latency minimal. Oftentimes, to accommodate the optimal VNF relocations, major VNF allocation changes are introduced in the network, while leading to unnecessary migration overhead. Hence, in the next chapter, we extend the research to address the aforementioned issues with a static VNF orchestrator.

Chapter 4

Real-time VNF Deployment for Mobile IoT Environment

In the 6G networks, due to the massive IoT connectivity and substantial growth of communication traffic, an effective VNF orchestration scheme is anticipated to function dynamically and intelligently. Moving beyond the traditional paradigm of the VNF orchestration and employing VNFs on the network edge located cloudlets based on the inspiration from multi-access edge computing can intensify the overall performance of delay-sensitive applications. In this chapter, we intend to investigate how to simultaneously leverage the ensembling of multiple deep learning models for proper calibration to provide real-time VNF placement solutions. We also address the challenges associated with state-of-the-art approaches to deal with dynamic network traffic and topology patterns. Our envisioned methods, based on Convolutional Neural Networks and Artificial Neural Networks named as E-ConvNets and E-ANN respectively, suggest two proactive VNF deployment strategies. These VNF placement strategies demonstrate (simulation results) encouraging performance (optimality gap nearly 7%) in terms of minimizing relocation and communication costs, and high scalability intelligence factor (around 0.93). Moreover, the presented results are further indications of integrating edge computing and deep learning-based strategies into similar research enigmas for future telecommunication networks.

4.1 Introduction

Based on the expectations to fulfill the demands of ultra-high processing speed and low communication delay sensitive applications, 6G cellular networks are envisioned to support an extensive variety of vertical use cases [87]. Some of the applications can be the massive connectivity of IoT, collaborative computing, remote surgery and machinery, augmented reality (AR), virtual reality (VR), and autonomous driving [69]. Nevertheless, the current network service orchestration schemes become incompetent to handle numerous service specifications and various device types due to not implying sustainability for real-time applications and poor administration capability [69]. Thus, NFV [82] pledges to facilitate network service provisioning at considerably decreased capital costs and operational expenditure. The intention is to replace the necessity of proprietary hardware devices with the software enabled implementation of VNFs on conventional virtualized platforms such as VMs running in cloudlets (small scale data centers at the edge of Internet) [82]. As suggested by the concept of MEC or fog computing [66], VNFs (e.g., firewall, load balancer, WAN accelerator, and intrusion detection system) placed at cloudlets in closer proximity to the users diminishes the burden of unnecessary data traversal and bandwidth consumption through the centralized cloud. The vision towards future telecommunication networks anticipates that the third parties will designate the content-aware and user-specific services along with their corresponding specifications, for example, the highest tolerable latency or least throughput limits, to the network administrator, expedited by NFV and SDN [88].

Accordingly, the deployment of these VNFs requires a highly efficient and scalable strategy to deal with the continually evolving network dynamic patterns and the large volume of traffic emerging from value-added services [70]. Mostly, state-of-the-art NFV resource orchestrators consider the static condition of networks, while ignoring the temporal differences in network traffic and topology due to mobility of users or congestion [83] [82]. Moreover, the lack of considering the re-computation of VNF placements in these methods makes them ill-equipped to be employed in practical settings, and often the consequences are violations of the QoS and SLA [89]. Ongoing researches evolve around different optimization formulations using ILP and MILP to outline the VNF orchestration scheme, which is NP-hard by nature [84] [39], and fail to offer fast VNF placements decisions at different times. To approach intelligent VNF orchestration, meta-heuristic based swarm intelligence algorithm, specifically, ACO, has been proposed in [39]. However, the family of swarm intelligence algorithms, including ACO, requires extensive parameter tuning of exploitation-exploration ratios, making them heavy weighted to accommodate for real-time orchestration [39]. Moreover, in some cases, the ACO or other meta-heuristics may provide network function placement solutions far off from in terms of reducing overall communication delay [39].

In distinction to the existing works, we have aimed to propose lightweight and dynamic deep learning [18] aided strategies for the VNF orchestration and deployment that facilitates both users and services provides exclusively by collaborative minimization of communication, relocation delay and costs in real-time. We have considered two popularly known approaches, Convolutional Neural Networks [18] and Artificial Neural Networks [18] blended with the twist of ensemble training and prediction fashion [90]. The pre-trained models can be placed on the cloudlets (local, device hosted or infrastructure based clouds) so the VNF orchestration process may be swift and prompt enough for delay-sensitive IoT applications [82]. Furthermore, the pre-trained proposed deep learning models employed for VNF placement dismiss the obligation to tune hyperparameters during service orchestration. The expeditious growth of interconnected devices and real-time IoT applications raises the concerns to direct towards adaptive orchestration schemes for VNFs that can function intelligently. The major contributions and findings of this chapter have been outlined in the following:

- Firstly, we explain the necessity of integrating deep learning-based cloudlet VNF deployment in the next-generation networks to offer ultra-low communication delay for real-time IoT applications. As a use case, we select the system model and problem formulation of mobility aware VNF deployment from an existing literature [39]. Then, we urge on the demands for deep learning enabled ensemble approaches and eventually validate through extensive simulation studies for prospective service orchestration based telecommunication researches.

- We propose the employment of intelligent VNF orchestration using two ensemble deep learning techniques that are E-ANN and E-ConvNets. The primary concern remains to offer ultra-low response time for dealing with large-scale VNF placement due to progressively increasing IoT devices and exhibit low running time compared to the traditional approaches. Ensembling is used to serve the purpose of calibration in the VNF orchestration process.

- The proposed techniques suggest to ensure minimum communication delay in the network with the least possible migration overhead collaboratively, that can ensure the utmost privileges both for the providers and users.

- The effectiveness of the deep learning-based edge inspired VNF orchestration is evaluated by extensive experimental analysis in different settings. Based on the experimental studies, this chapter demonstrates that E-ANN and E-ConvNets are capable of providing near optimal real-time solutions for large-scale IoT services, unlike traditional mathematical programming approaches, for example, ILP. Ensemble deep learning-based methods show a significant escalation in the performance evaluation even though compared to the improved and hyperparameter tuned version of ACO [39], named as t-ACO hereafter. The results also designate the potentiality of different deep learning-based methods to be applied for intelligent orchestration services in future networks.

- This chapter elaborately justifies the generalization aptitude of proposed models through validation experiments: a) simulation results on a National Research and Education Network (NREN) and b) bias-Variance trade-off analysis.

Figure 4.1 outlines the research gap between our work and traditional focus.

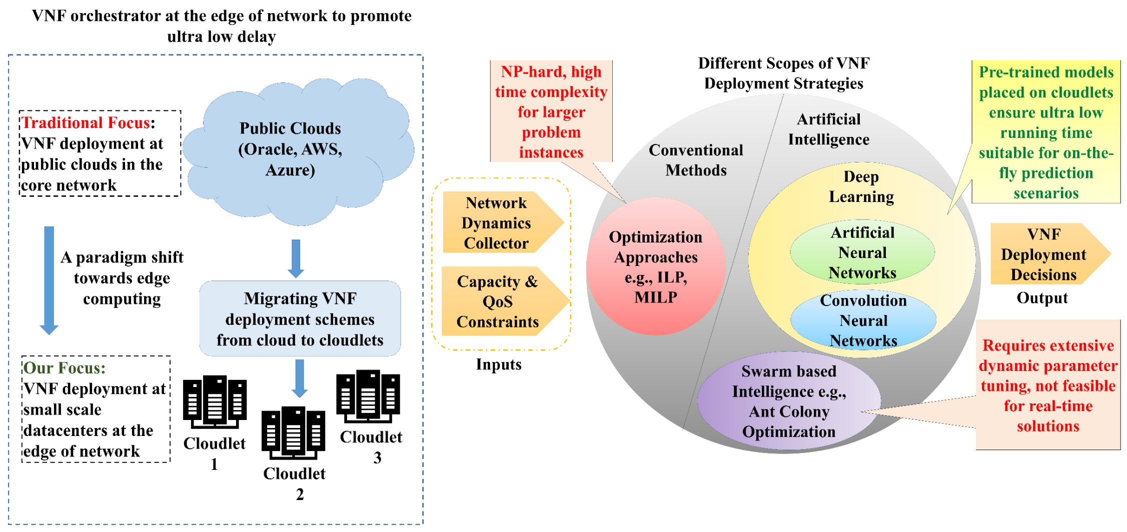

VNF orchestrator at the edge of network to promote ultra low delay
Traditional Focus: VNF deployment at public clouds in the core network
Public Clouds (Oracle, AWS, Azure)
A paradigm shift towards edge computing
Migrating VNF deployment schemes from cloud to cloudlets
Our Focus: VNF deployment at small scale datacenters at the edge of network
Cloudlet 1
Cloudlet 2
Cloudlet 3
Network Dynamics Collector
Capacity & QoS Constraints
Inputs
NP-hard, high time complexity for larger problem instances
Different Scopes of VNF Deployment Strategies
Conventional Methods
Artificial Intelligence
Optimization Approaches e.g., ILP, MILP
Deep Learning
Artificial Neural Networks
Convolution Neural Networks
Swarm based Intelligence e.g., Ant Colony Optimization
Pre-trained models placed on cloudlets ensure ultra low running time suitable for on-the-fly prediction scenarios
VNF Deployment Decisions
Output
Requires extensive dynamic parameter tuning, not feasible for real-time solutions

Figure 4.1: Deep learning assisted VNF Deployment at the cloudlet data centers

4.2 Related Work

NFV technology emerges around different researches to enhance the efficiency regarding scaling [26], allocation of VNFs [91], task scheduling [92] [71], and migration of VNFs [93], etc. Recently, a few extensions have driven the policy-concerned traffic problem utilizing hardware middle-boxes [26] [92]. There have been several kinds of research considering VNF deployment in the context of heterogeneous networks. A broad plethora of research studies related to VNF deployment in the hybrid cloud has been emerging lately [94] [79] [95]. To enhance the Quality-of-Experience (QoE) of the users, massive computation time critical applications require a large volume of communication resources. Thus, a lot of focus has been drawn towards resource constraints management based VNF deployment strategies [96] [97]. The authors of [97] have proposed distributed VNF deployment by caching resources, yet this mechanism is unable to manage dynamic network situations. Moreover, some researches focus solely on different VNF migration schemes [98] [99]. Ben et al. have proposed a capacitated VNF migration scheme with the help of Virtual Network Embedding (VNE) [89]. However, the main focus has been deviated away from communication delay that may affect the overall user experience.

A stable matching algorithm to reduce the execution time by introducing mixed-integer programming and obtaining the near-optimal results in terms of latency has been approached [83]. This algorithm can prevent the failure of the model in extreme cases. Even so, this model has been designed according to static network arrangements, which would require to be initiated every time instance not being feasible for online applications. A further extension to this work by utilizing local search has been proposed in [100]. Fei et al. use artificial intelligence to predict the service chain requests that have been able to gain improved competitive ratios, but ignores dynamic processing capability [82].

As a step closer towards intelligent VNF orchestration, swarm-based intelligence inspired by the natural behavior of ant colony has been approached [39], which considers both user mobility based VNF relocation and communication costs. However, the difficulty is that these types of algorithms need a lot of attention towards parameter tuning that require longer execution times to deal with large scale and real-time scenarios [70]. Hence, these algorithms are not generally suitable for critical use cases, such as, services regarding Unmanned aerial vehicle (UAV), space-air-ground integrated network, and Unmanned underwater vehicle (UUV) [102] [103]. A service placement using federated learning on edge clouds, while focusing privacy concerns of users have been proposed in [101]. Table 4.1 summarizes the relevant literature and provides brief comparison among the existing research works. Deep learning also appears to exhibit promising results in intelligent resource allocation for cloud based services [104]. Ensemble methods are the combination of different machine learning models into a single predictive model to minimize bias, variance and improve the capability of prediction [90].

Among all the presented literature works, we have preferred [39] over others to compare the performances of our proposed deep learning models. The reason being that only this specific literature among others that have been listed in Table 4.1 integrates both VNF allocation and migration resembling our research focus. To the best of our knowledge, none of the existing works suggested the use of ensemble

Table 4.1: Summarization of related research works

Ref.	Target of Concern	Approach	Limitations
[79] [96]	VNF deployment in hybrid network architecture and cloud	Cut-and-solve, mixed integer gaming	Static, not suitable for real-time applications, no consideration for edge computing, non scalable, and not QoS aware
[97] [89]	VNF deployment	Resource caching	Static, not suitable for real-time applications, non scalable, and no support for edge driven services
[98] [99]	Capacitated VNF Migration Scheme	Virtual Network Embedding (VNE)	Not QoS optimal, high communication delay, non scalable, and not edge driven
[83] [100]	VNF allocation	Stable matching, local search	Static, non scalable, and not fit for real-time use cases
[82]	Virtual service prediction	Online learning	Not suitable for real-time applications, non scalable, and no support for edge driven computation
[39]	VNF allocation and migration in the cloud	Meta-heuristics based ant colony optimization	Extensive range of online parameter tuning, no support for edge driven computation, non scalable, and high orchestration time makes it incapable for real-time or time critical, large scale use cases
[101]	Privacy awareness	Distributed Learning	Non QoS Optimal, non scalable, not suitable for real-time use cases

deep learning assisted strategies for VNF orchestration in order to function within reasonable running time limits, promote scalability and support both the providers and users interest mutually, while approaching towards 6G cellular networks.

4.3 System Model

The system architecture [39] of the network, as shown in Figure 4.2 includes two distinct domains. One of them is the cloudlet domain or edge domain, while the

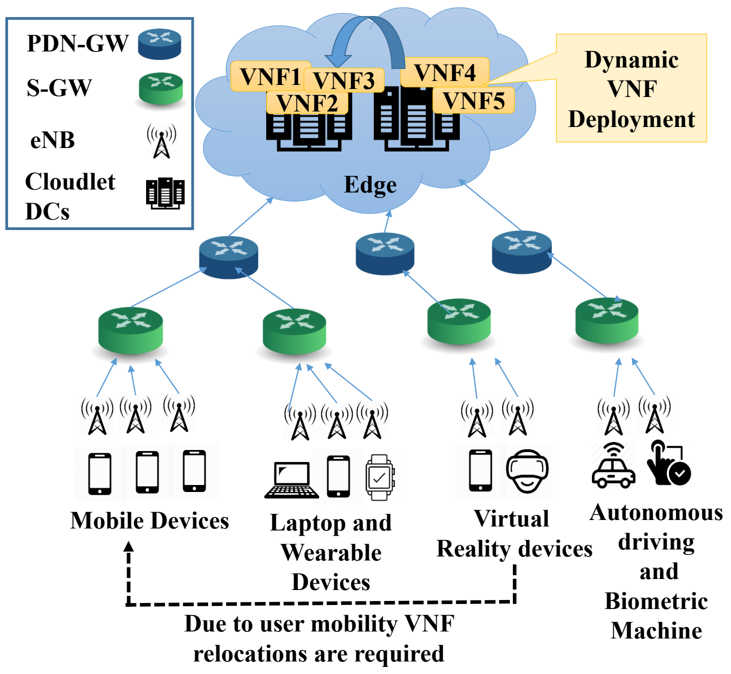

PDN-GW
S-GW
eNB
Cloudlet DCs
VNF1
VNF3
VNF2
VNF4
VNF5
Dynamic VNF Deployment
Edge
Mobile Devices
Laptop and Wearable Devices
Virtual Reality devices
Autonomous driving and Biometric Machine
Due to user mobility VNF relocations are required

Figure 4.2: A high level system architecture for VNF orchestration.

other one is the Radio Access Network (RAN) domain. The cloudlet domain consists of a set of small scale data centers (DCs), D at the edge of the Internet, having secure and robust wired connections among them. On exploiting cloudlet confederation, the connected cloudlet DCs can offer and receive services from one another. Besides, the domain of RAN incorporates a set of access points, for example, base stations termed as evolved NodeB (eNB). A number of users can be connected to each eNB through radio signal. A base station controller usually manages a collection of eNBs. A single eNB is allowed to be connected to only one cloudlet DC via the Serving Gateway (S-GW) and Packet Data Network Gateway (PDN-GW) of the particular DC. However, numerous numbers of eNBs can be connected to a cloudlet DC. In certain occupied and busy zones, cloudlet DCs serve an enormous number of eNBs, and in case of lightly packed regions, cloudlet DCs serve limited numbers of eNBs associated with them.

The Virtual Network Functions (VNFs) are run on different cloudlet DCs, where each DC has a restricted capability of supporting service oriented or application VNFs. A set of eNBs, E can receive service from its associated DC. A data center can provide direct services by running the corresponding VNF of a client or user under the eNB connected to that cloudlet DC. Moreover, it can offer passive services via neighbouring DCs, which demands additional service cost. Due to user mobility, V_j can be considered as the set of VNFs of eNB, $e_j \in E$ that are required to be relocated. The requests for VNF migration usually occur because of hand off between an eNB, $e_j \in E$ and other eNBs that are connected to different cloudlet DCs unlike e_j. Table 4.2 exhibits the major notations along with their description used to implement the optimization framework.

Table 4.2: Description of parameters for our system model

Notation	**Description**
D=$\{d_1, d_2, ..., d_D\}$	The set of cloudlet DCs in the network
E=$\{e_1, e_2, ..., e_E\}$	The set of all eNBs connected to the cloudlet DC where the system is running
V=$\{v_1, v_2, ..., v_V\}$	The set of all VNFs
V_j	The set of VNFs of eNB, $e_j \in E$ that are required to be relocated, where $V_j \subseteq V$
ϑ_{worst}	Maximum communication delay toleration limit of the network
$t_{k,l}$	The communication delay between a cloudlet DC, $d_k \in D$ and the cloudlet DC, $d_l \in D$, provided that $k \neq l$
$\tilde{t}_{j,k}$	The communication delay between eNB, $e_j \in E$ and the cloudlet DC, $d_k \in D$
S_i	Size of VNF, $v_i \in V$
ϕ_k	Cost to place any VNF to some cloudlet DC, $d_k \in D$
ψ_k	Cost to take service from cloudlet DC, $d_k \in D$
C_k	Capacity of the cloudlet DC, $d_k \in D$ for holding VNFs
σ_i	Execution time of VNF, $v_i \in V$
η	Priority factor of VNF migration or relocation
τ_i	Transfer time of VNF, $v_i \in V$
N_k	Number of VNFs that are already executing in cloudlet DC, $d_k \in D$
$\Upsilon^i_{k,j}$	Summation of communication, relocation, and execution time if VNF, $v_i \in V_j$ of eNB, $e_j \in E$ is placed at cloudlet DC, $d_k \in D$

4.4 Optimization Framework for VNF Deployment

The primary objective of ILP formulation for VNF deployment is to ensure an optimal placement of VNFs that are required to be relocated due to the mobility of the users [39]. The ILP formulated optimization framework considered in this chapter has been proposed in [39]. The idea is to run the ILP based VNF placement strategy in each cloudlet DC to manage the VNF requests coming from the eNBs under it or the eNBs that are connected to different data centers. The ILP design focuses to minimize the number of relocations and maximize QoE for users. To introduce a trade-off between these two conflicting objectives a priority factor η has been considered. According to network size and Service Level Agreement (SLA), the priority factor η can be set carefully by the service providers. In order to deploy VNF, $v_i \in V_j$ of eNB, $e_j \in E$ to cloudlet DC, $d_k \in D$, the relocation time $R^i_{k,j}$ can be calculated using the following equation:

$$R^i_{k,j} = \{(1 - \epsilon^i_k) \times b^i_{k,j}\} \times Q^i_k \tag{4.1}$$

where, ϵ^i_k holds 1, if the VNF instance have been earlier running on cloudlet, DC $d_k \in D$, otherwise 0. Therefore, in case $(1 - \epsilon^i_k)$ is 1, the corresponding VNF instance can be considered for relocating to a cloudlet, DC $d_k \in D$. Likewise, the decision variable $b^i_{k,j}$ holds 1 if VNF, $v_i \in V_j$ of eNB, $e_j \in E$ is placed at cloudlet DC, $d_k \in D$, otherwise 0. The relocation cost to migrate VNF, $v_i \in V_j$ of eNB, $e_j \in E$ to some cloudlet DC, $d_k \in D$ can be represented by Q^i_k and calculated as follows:

$$Q^i_k = (1 - n^i_k) \times \tau_i \tag{4.2}$$

Again, n^i_k holds 1, if the expected VNF, $v_i \in V_j$ of eNB, $e_j \in E$ is running on the cloudlet DC, $d_k \in D$, otherwise 0. Therefore, upon the value of $(1 - n^i_k)$ being 1, we need to relocate or transfer the VNF from the previous DC. In such case, the relocation cost is equal to transfer time τ_i, which can be calculated using the following equation:

$$\tau_i = \frac{S_i}{r} \tag{4.3}$$

where, r is the achievable data rate to relocate any VNF and S_i represents the size of the VNF, v_i. The communication delay to get service for a VNF, $v_i \in V_j$ is calculated from the following equation:

$$T^i_{k,j} = b^i_{k,j} \times (\tilde{t}_{j,k} + t_{k,l}) \tag{4.4}$$

here, the summation of communication delay between eNB, $e_j \in E$ and the cloudlet DC where the solution is executing $(\tilde{t}_{j,k})$ along with the communication delay between cloudlet DC, $d_k \in D$ and the cloudlet DC, $d_l \in D$, which holds the running VNF $(t_{k,l})$

contribute to the total communication delay. In the case of taking direct services from own cloudlet DC, where the solution is running, there is no requirement to consider and calculate $t_{k,l}$. On the contrary, for taking services via neighboring cloudlet DCs, we need to add both $\tilde{t}_{j,k}$ and $t_{k,l}$ together to find the cumulative communication delay $T^i_{k,j}$. Finally, the objective function formulation using ILP can be presented as the following:

$$minimize \sum_{e_j \in E} \sum_{v_i \in V_j} \sum_{d_k \in D} \{\eta \times R^i_{k,j} \times \phi_k + (1-\eta) \times T^i_{k,j} \times \psi_k\} \tag{4.5}$$

The first part of the objective function interprets the relocation costs (ϕ_k) multiplied by required relocation time ($R^i_{k,j}$) to DC, $k \in D$ for VNFs, $v_i \in V_j$ of eNB, $e_j \in E$. The following part refers to the communication cost in terms of communication time ($T^i_{k,j}$) and costs (ψ_k) for eNBs to take service from their directly connected DC or distant cloudlet DCs. A trade-off is introduced by estimating the priority factor denoted as η. The objective is to minimize the overall network relocation and communication costs of the network. This particular and most suitable way of integrating multiple objective functions is known as the weighted sum method. In contrast, the other popularly known state-of-the-art approaches (e.g., $\epsilon-$ constraint and weighted metric method) require prior knowledge of posterior facts, which is nearly impossible for real-life use cases. The objective function presented in the Eq. (4.5) is subject to the following constraints:

$$\mathcal{C}1 : \sum_{d_k \in D} b^i_{k,j} = 1, \ \forall_{e_j \in E}, \ \forall_{v_i \in V_j} \tag{4.6}$$

$$\mathcal{C}2 : \sum_{v_i \in V_j} \sum_{d_k \in D} b^i_{k,j} = |V_j|, \ \forall_{e_j \in E} \tag{4.7}$$

$$\mathcal{C}3 : \sum_{d_k \in D} (R^i_{k,j} + T^i_{k,j} + \sigma_i) \leq \vartheta_{worst}, \ \forall_{e_j \in E}, \ \forall_{v_i \in V_j} \tag{4.8}$$

$$\mathcal{C}4 : \sum_{e_j \in E} \sum_{v_i \in V_j} b^i_{k,j} \leq C_k, \ \forall_{d_k \in D} \tag{4.9}$$

$$\mathcal{C}5 : b^i_{k,j} \in \{0,1\}, \ \forall_{e_j \in E}, \ \forall_{v_i \in V_j}, \ \forall_{d_k \in D} \tag{4.10}$$

The constraint $\mathcal{C}1$ is basically an atomicity constraint, ensuring the single assignment of each VNF, $v_i \in V_j$ of eNB, $e_j \in E$ to exactly one cloudlet DC, $d_k \in D$. Another constraint $\mathcal{C}2$ specifies that all VNFs, $v_i \in V_j$ of eNB, $e_j \in E$ must be allocated to some cloudlet DC, $d_k \in D$. Next, The QoS constraint $\mathcal{C}3$ guarantees the summation of relocation delay, communication delay, and execution time of VNFs to remain below a certain pre-defined threshold ϑ_{worst}, which can be varied according to application nature. The capacity constraint $\mathcal{C}4$ assures not to overload cloudlet DCs. Hence, the number of VNFs executing in a cloudlet DC is not allowed to exceed the capacity of that cloudlet DC. Finally, the constraint $\mathcal{C}5$ is a binary constraint representing the value of decision variable $b^i_{k,j}$ to be 1, in case VNF, $v_i \in V_j$ of eNB, $e_j \in E$ is placed at cloudlet DC, $d_k \in D$, otherwise remains 0.

4.5 Metaheuristic based ACO approach for VNF deployment

We consider an existing AI-based ACO algorithm that claims to find the near-optimal solution in a reasonable amount of time in case of practical settings [39]. The proof to the NP-hardness of this problem can be found in [39] as well. The master strategy of swarm intelligence algorithms is simple to modify for different problem domains by generating appropriate heuristics in order to acquire solutions close to optimal.

The ACO has been inspired by the collective behavior of real ant colonies [39]. In this problem, a set of virtual ants is created, where each ant possesses a short memory. These ants attempt to build a solution using heuristic values and improve the state of the solution by interchanging learning via pheromones among themselves. In a distributed manner, each ant tries to construct a local solution and updates its local pheromone trail eventually. Finally, the locally found solutions are consolidated to construct a global solution. The VNF deployment algorithm concerning ACO based strategy has been presented in algorithm 3 [39].

First of all, the values of several system parameters and set of virtual ants are initialized in lines 1-2. Next, the line numbers 3 and 4 calculate the initial pheromone value ζ_0, and generate an initial set of solution employing the First Fit VNF (FF-VNF) deployment algorithm, respectively. Each ant produces a local set of solutions for deploying VNFs of all eNBs to appropriate DC based on initial or updated pheromone values and local heuristic values in lines 8 to 10, while satisfying capacity and QoS constraints. For each solution of VNF placement to some cloudlet DC found by an ant, the local pheromone trail value is updated based on a relative weight factor ω_l to encourage exploration of the search space and diversification of solution by making

Algorithm 3: ACO based VNF Deployment algorithm at each cloudlet DC $d_k \in D$

```
   Input: $E$, $Vj$, $D$
   Output: cloudlet DC-VNF pairs for each eNB
1  Initialize system parameters $\alpha$, $\beta$, $\omega_l$, $\omega_g$
2  Initialize a set of ants $A$
3  Compute initial value of pheromone $\zeta_0$ using Eq. (4.11)
4  Construct an initial solution using Algorithm 4
5  Set the value of $total\_iterations$
6  while ($iteration \leq total\_iterations$) do
7      foreach ant $a_z \in A$ do
8          foreach eNB $e_j \in E$ do
9              foreach VNF $v_i \in V_j$ do
10                 Assign VNF $v_i \in V_j$ of eNB $e_j \in E$ to some DC $d_k \in D$ using
                    Eq.(4.14)
11         foreach VNF $v_i \in V_j$ do
12             $\zeta^i_{k,j} \leftarrow \omega_l \times \zeta_0 + (1 - \omega_l) \times \zeta^i_{k,j}$
13     Update the value of global pheromone using Eq. (4.16)
14     $iteration = iteration + 1$
15 return cloudlet DC-VNF pairs for each eNB
```

the already found solutions less desirable for ants in lines 11-12. The system constant ω_l shows the relative priority of historical and current pheromone values. The global solution set is obtained as the best local solution set among all the solution sets found locally by all ants after repeating the steps mentioned so far. Then, the value of global pheromone is updated in line 12. The overall algorithm has been elaborately discussed in the following subsections.

4.5.1 Calculation of Initial Pheromone Value

In the case of the VNF deployment problem, the pheromone value indicates the potentiality metric of placing a VNF to a cloudlet DC. Every ant begins with an underlying pheromone estimation for each VNF to cloudlet DC pair. The primary arrangements of VNF- cloudlet DC pairs are generated utilizing the FF-VNF deployment approach recorded in algorithm 2. This algorithm allocates the VNFs of all the eNBs to the cloudlet DCs based on first fit approach in lines 2-9, while securing the placements to avoid violation of application tolerable QoS limit and without exhausting cloudlet DCs.

Algorithm 4: FF-VNF Deployment at each cloudlet DC $d_k \in D$

```
Input: $E$, $Vj$, $D$
Output: cloudlet DC-VNF pair for each eNB in initial solution $\delta_0$
1 $\delta_0 \leftarrow \varnothing$
2 foreach eNB $e_j \in E$ do
3     foreach VNF $v_i \in V_j$ do
4         foreach DC $d_k \in D$ do
5             if ($N_k < C_k$ and $\Upsilon^i_{k,j} < \vartheta_{worst}$) then
6                 $\delta_0 \leftarrow \delta_0 \cup (e_j, v_i, d_k)$
7                 $N_k = N_k + 1$
8                 Break
9 return $\delta_0$
```

The initial pheromone value is determined by the inversed summation of the total relocation delay and total communication time of the system, hence calculated as follows:

$$\zeta_0 = \sum_{e_j \in E} \sum_{v_i \in V_j} \sum_{d_k \in D} \frac{1}{(R^i_{k,j} + T^i_{k,j})} \times \varrho^k_{j,i} \tag{4.11}$$

where $\varrho^k_{j,i}$ is a decision variable which can be represented as following:

$$\varrho^k_{j,i} = \begin{cases} 1, & if\ (e_j, v_i, d_k) \in\ \delta_0 \\ 0, & otherwise \end{cases} \tag{4.12}$$

δ_0 is the initial solution created by FF-VNF deployment strategy. If a solution for VNF, $v_i \in V_j$ of eNB, $e_j \in E$ being placed on cloudlet DC, $d_k \in D$ exists in initial solution δ_0, then the value of decision variable $\varrho^k_{j,i}$ will be 1, otherwise 0. The higher the inverse summation of relocation time and communication delay, the more pheromone is deposited towards that solution.

4.5.2 Heuristic Formulation

With the intent to minimize both relocation and communication delay with costs, the heuristic value can be determined by the following equation:

$$H^i_{k,j} = \frac{1}{\eta \times R^i_{k,j} \times \phi_k + (1 - \eta) \times T^i_{k,j} \times \psi_k} \tag{4.13}$$

The Eq. (4.13) ensures that the more inferior the weighted total of relocation and communication cost for deploying a VNF to a cloudlet DC, the greater will be the heuristic value used to select that cloudlet DC.

4.5.3 Cloudlet DC Selection Technique

We assume that $D_c \subseteq D$ is the set of candidate cloudlet DCs, which have available capacity to further accommodate VNFs. To select the cloudlet DC, the pseudo random proportional action rule has been defined below:

$$s = \begin{cases} \underset{d_k \in D_c}{\operatorname{argmax}}([\zeta^i_{k,j}]^{\alpha} \times [H^i_{k,j}]^{\beta}), & if\ p \leq p_0 \\ \underset{d_k \in D_c}{\operatorname{argmax}}\, p^z_{k,j,i}, & otherwise \end{cases} \tag{4.14}$$

p_0 and p are system parameters and randomly chosen from uniformly distributed values respectively, and both range between 0 and 1. In case of $p \leq p_0$, exploitation of the search space occurs, on the other way around exploration is performed based on the following equation:

$$p^z_{k,j,i} = \begin{cases} \frac{([\zeta^i_{k,j}]^{\alpha} \times [H^i_{k,j}]^{\beta})}{\sum\limits_{d_k \in D_c} ([\zeta^i_{k,j}]^{\alpha} \times [H^i_{k,j}]^{\beta})}, & d_k \in D_c \\ 0, & otherwise \end{cases} \tag{4.15}$$

4.5.4 Global Pheromone Update

After the construction of a set of local solutions, the global pheromone value is calculated as follows:

$$\zeta^i_{k,j} = \omega_g \times \Delta\zeta^i_{k,j} + (1 - \omega_g) \times \zeta^i_{k,j} \tag{4.16}$$

ω_g exhibits the relative importance factor of $\Delta\zeta^i_{k,j}$ and $\zeta^i_{k,j}$ in Eq. (4.16). We assume that the set of global solutions is denoted by Γ. The variable $\Delta\zeta^i_{k,j}$ can be represented in the following equation:

$$\Delta\zeta^i_{k,j} = \begin{cases} \zeta^i_{k,j}, & if\ (e_j, v_i, d_k) \in \Gamma \\ 0, & otherwise \end{cases} \tag{4.17}$$

If a solution for VNF, $v_i \in V_j$ of eNB, $e_j \in E$ being placed on cloudlet DC, $d_k \in D$ exists in the global solution Γ, then the value of variable $\Delta\zeta^i_{k,j}$ will be $\zeta^i_{k,j}$, otherwise

0.

4.5.5 System Parameters

All the parameters to implement ACO have been listed in Table 4.3.

Table 4.3: Description of parameters for ACO inspired VNF orchestration

Notation	Description
ζ_0	Initial pheromone value
$\zeta^i_{k,j}$	Pheromone value for deploying VNF, $v_i \in V_j$ of eNB, $e_j \in E$ to cloudlet DC, $d_k \in D$
$H^i_{k,j}$	Local heuristic value for deploying VNF, $v_i \in V_j$ of eNB, $e_j \in E$ to cloudlet DC, $d_k \in D$
$p^z_{k,j,i}$	Probability for selecting cloudlet DC, $d_k \in D$ for deploying VNF, $v_i \in V_j$ of eNB, $e_j \in E$ by an ant a_z
ω_l	Weight constant for local pheromone value update
ω_g	Weight constant for global pheromone value update
$\Delta\zeta^i_{k,j}$	Global pheromone value for deploying VNF, $v_i \in V_j$ of eNB, $e_j \in E$ to cloudlet DC, $d_k \in D$
α	Weight constant for pheromone value
β	Weight constant for local heuristic value

The dynamic parameter tuning of system constants for ACO implementation demand extensive research work. Through simulation results and researches existing in literature, we have considered $\omega_l = 0.3$, $\omega_g = 0.4$, $\eta = 0.7$, number of ants = 20, and maximum 200 iterations for all the performance evaluations [39]. Unlike the literature, we dynamically vary the value of α and β to improve the performance of existing ACO algorithm. The detailed process on how we tune these parameters has been explained in Section 4.7.

4.6 Proposed Deep Learning Aided VNF Deployment

The future of cellular networks and NFV infrastructure manager expect to exploit AI for offering intelligent orchestration and management systems [87]. In this section, we propose two ensemble deep learning techniques using E-ANN and E-ConvNets for AI enabled VNF deployment [18]. The concept is to locate the pre-trained models in cloudlet DCs so that the deployment decisions induced by the testing phase can offer real-time solutions with ultra-low execution time required for prediction.

4.6.1 Labeled Dataset Generation

We leverage the ILP optimization framework solver described in section 4.4 to optimize different VNF deployment scenarios, and then record the respective solutions to generate labeled data for training purpose. There are two primary reasons behind generating the labeled datasets using ILP:

- There are no standardized datasets for VNF resource allocation or related researches. Thus, for such problems, ILP (optimal) results can be leveraged to create labeled datasets for supervised learning.
- Reinforcement learning for resource allocation problems where the outcome numbers are too high (feasible solution space) are not effective [105] [106]. It is known that reinforcement learning can learn well when the number of actions are marginal [105] [55].

The input features for the training models are considered as D, E, V_j, ϑ_{worst}, $R^i_{k,j}$, $T^i_{k,j}$, ϕ_k, ψ_k, C_k, σ_i, η, τ_i, and N_k as presented in the algorithm 5. We merge the features along with the decision variable $b^i_{k,j}$ or target variable found by ILP solver to produce a labeled dataset for training purpose in lines 5-7 of algorithm 5. We denote the labeled dataset as $\mathcal{L}$.

Algorithm 5: Ensemble training phase of deep learning aided VNF Deployment at each cloudlet DC $d_k \in D$

```
Input: $D$, $E$, $V_j$, $\vartheta_{worst}$, $R^i_{k,j}$, $T^i_{k,j}$, $\phi_k$, $\psi_k$, $C_k$, $\sigma_i$, $\eta$, $\tau_i$, and $N_k$
Output: Set of trained models $\mathcal{M}^*$
1 $\mathcal{M}^* \leftarrow \varnothing$
2 foreach model $m_t \in \mathcal{M}$ do
3     $\mathcal{L} \leftarrow \varnothing$
4     for epoch < total simulation epochs do
5         $\mathcal{S} \leftarrow$ Generate a random system using input parameters
6         $b^i_{k,j} \leftarrow$ Assign the decision variable by solving system $\mathcal{S}$ through the
            ILP optimizer framework
7         $\mathcal{L} \leftarrow \mathcal{S} \cup b^i_{k,j}$
8     Train the model $m_t$ using labeled dataset $\mathcal{L}$
9     $\mathcal{M}^* \leftarrow \mathcal{M}^* \cup m_t$
```

4.6.2 Ensemble Convolutional Neural Netowrks (E-ConvNets)

We have applied the Convolutional Neural Network due to its extraordinary performance in pattern identification that may aid VNF deployment strategies for user specific services and content aware networks [18]. However, to generate a well-calibrated model due to the uncertain nature of the network parameters, we have incorporated the ensembling technique into the model utilizing E-ConvNets [90]. The E-ConvNets method consists of a set of alternative different convolutional network models $\mathcal{M}$. Each model $m_t \in \mathcal{M}$ is trained by different randomly generated datasets as explained in lines 2-7 of algorithm 5. Finally, at the end of this algorithm as suggested in line 8, we receive a set of trained ensemble models $\mathcal{M}^*$ that are further used for deploying VNF $v_i \in V_j$ of eNB $e_j \in E$ to some cloudlet DC $d_k \in D$ using algorithm 6. The testing or prediction phase of ensemble techniques for E-ConvNets have been

Algorithm 6: Ensemble testing phase of deep learning aided VNF deployment at each DC $d_k \in D$

```
   Input: $D$, $E$, $V_j$, $\vartheta_{worst}$, $R^i_{k,j}$, $T^i_{k,j}$, $\phi_k$, $\psi_k$, $C_k$, $\sigma_i$, $\eta$, $\tau_i$, and $N_k$
   Output: A set of solutions $\mathcal{F}$
1  $\mathcal{U} \leftarrow$ Generate a random system using input parameters
2  $\mathcal{F} \leftarrow \varnothing$
3  $\mathcal{X}^i_{k,j} \leftarrow 0$
4  foreach trained model $m_t \in \mathcal{M}*$ do
5      foreach eNB $e_j \in E$ do
6          foreach VNF $v_i \in V_j$ do
7              foreach cloudlet DC $d_k \in D$ do
8                  if ($N_k > C_k$ and $\Upsilon^i_{k,j} > \vartheta_{worst}$) then
9                      $\hat{\mathcal{Y}^i_{k,j}} \leftarrow 0$
10                 else
11                     $\hat{\mathcal{Y}^i_{k,j}} \leftarrow$ Set the confidence score between 0 and 1 by
                        applying trained model $m_t$ on $\mathcal{U}$
12     $\mathcal{X}^i_{k,j} \leftarrow \mathcal{X}^i_{k,j} + \hat{\mathcal{Y}^i_{k,j}}$
13 foreach eNB $e_j \in E$ do
14     foreach VNF $v_i \in V_j$ do
15         $temp_k \leftarrow \varnothing$
16         foreach DC $d_k \in D$ do
17             $temp_k \leftarrow temp_k \cup \mathcal{X}^i_{k,j}$
18         $max \leftarrow \underset{k}{\mathrm{argmax}}\ temp_k$
19         $\mathcal{F} \leftarrow \mathcal{F} \cup (e_j, v_i, d_{max})$
```

exhibited in algorithm 6. We have generated random unlabeled data $\mathcal{U}$ for performance evaluation in line 1. In lines 5-11, we verify the constraints and apply trained models $m_t \in \mathcal{M}^*$ exploiting E-ConvNets on unlabeled dataset $\mathcal{U}$ to generate the confidence scores for placing all VNF, $v_i \in V_j$ of every eNB, $e_j \in E$ to each cloudlet DC, $d_k \in D$. We update this confidence score obtained by each model in variable $\mathcal{Y}_{k,j}^{\hat{i}}$ that can range between 0.0 and 1.0. We accumulate the cumulative prediction confidence score of all trained models for deploying VNFs, $v_i \in V_j$ of eNB, $e_j \in E$ to cloudlet DC, $d_k \in D$, and store it in the variable $\mathcal{X}_{k,j}^{i}$ through line 12. In lines 14-18, we select the cloudlet DC, $d_k \in D$ that holds the highest cumulative confidence score generated by all trained models for every VNF, $v_i \in V_j$ of eNB, $e_j \in E$ pair. Finally, a set of solutions $\mathcal{F}$ is constructed iteratively containing respective eNB, $e_j \in E$, VNF, $v_i \in V_j$, and selected cloudlet DC, $d_k \in D$ in line 19.

The abstract architecture of E-ConvNets has been presented in the Figure 4.3. The sequential styled E-ConvNets feature a set of typical model designs [18]. Each

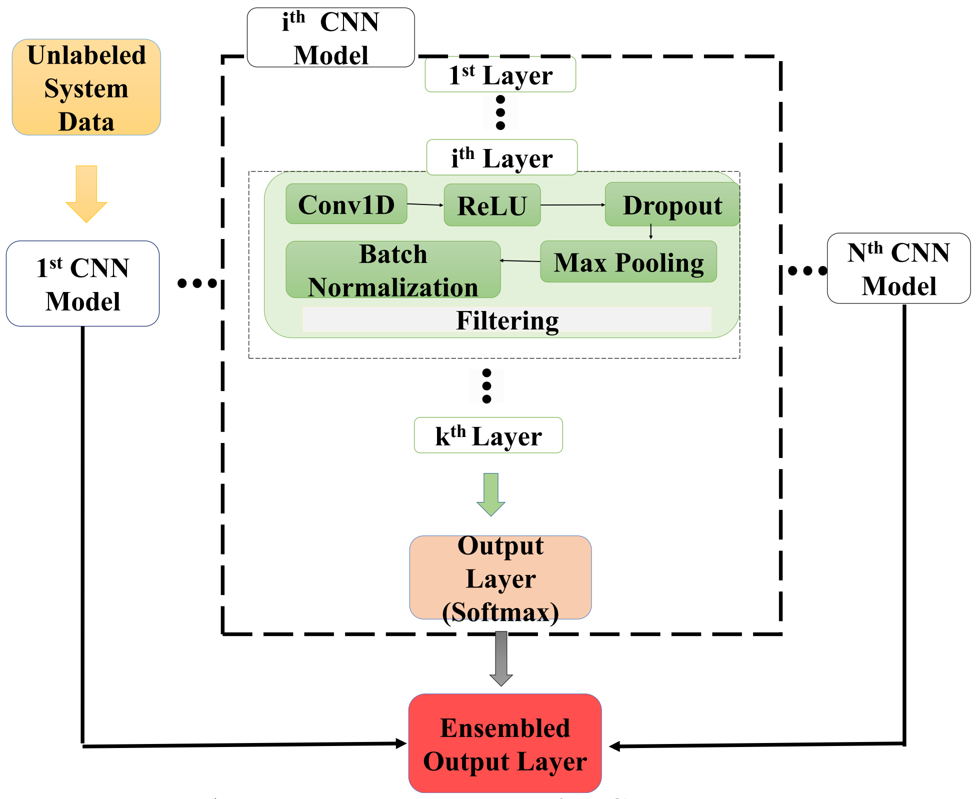


Figure 4.3: A high level description of E-ConvNets architecture.

model includes several convolutional layers that are followed by a pooling layer. Batch normalization is performed to enhance the performance, speed, and stability of models, thus require less computational complexity. For the convolutional layers, we select rectified linear (ReLU) activation function, while output layers use softmax function by following the cross-entropy loss function [18]. The detailed description on state-of-the-art deep learning architectures have been described in [18]. For ensembling, multiple CNN models are incorporated for the training and prediction phase rather than solely depending on the prediction of a single trained network.

4.6.3 Ensemble Artificial Neural Networks (E-ANN)

The ANN [18] [41] is a paradigm for processing information and usually configured according to the requirement of applications through the learning phase. Our proposed E-ANN consists of multiple models $m_t \in \mathcal{M}$. In order to train the E-ANN models, we apply the algorithm 5 on some randomly generated labeled data $\mathcal{L}$, as explained earlier in the subsection 4.6.1. The output layer of the E-ANN has equal number of nodes specifying cloudlet DCs for every model of the ensemble learning. The output nodes indicate the probability or confidence score of placing a VNF under some eNB to each cloudlet DC provided an unlabeled random system $\mathcal{U}$. Finally, we place a VNF on the cloudlet DC of the corresponding output node having the highest cumulative confidence score summed up from all the trained models. We employ the same algorithm 6 for the prediction phase of VNF deployment, while applying the E-ANN architecture. The detail explanation of the working procedure of this algorithm has been discussed under subsection 4.6.2.

Figure 4.4 illustrates the architecture of E-ANN. Instead of employing a single architecture of ANN, we utilize a set of trained models that altogether contribute to the final outputs. We assemble these models to accumulate the confidence score of each output node to interpret the ultimate set of output nodes for proper calibration [90]. The final layer of the E-ANN architecture utilizes softmax function under cross-entropy loss regime, while the nodes from hidden layer employ hyperbolic tangent function [18].

4.7 Performance Evaluation

In this section, we first discuss the procedure of hyperparameter tuning to upgrade the performance of t-ACO (tuned and improved version) comparing to ACO [39]. The original ACO existing in the literature use fixed hyperparameter (exploitation-

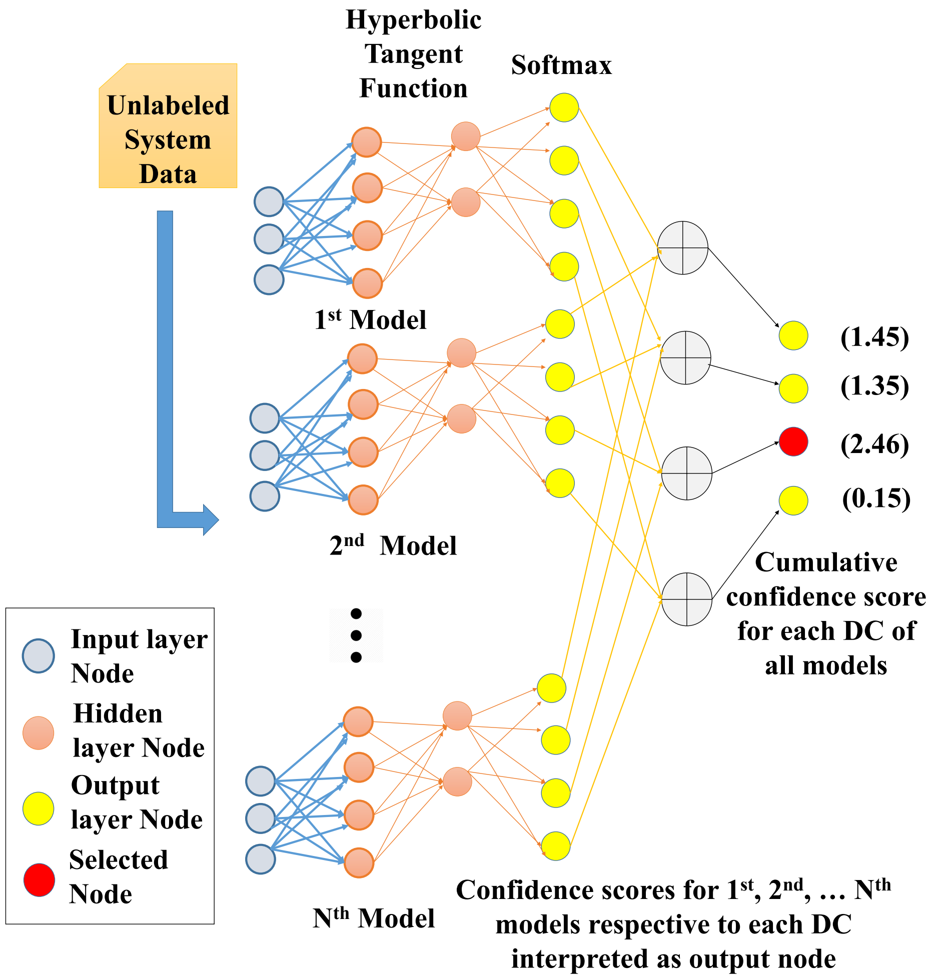

Hyperbolic Tangent Function
Softmax
Unlabeled System Data
1st Model
2nd Model
Nth Model
(1.45)
(1.35)
(2.46)
(0.15)
Cumulative confidence score for each DC of all models
Input layer Node
Hidden layer Node
Output layer Node
Selected Node
Confidence scores for 1st, 2nd, … Nth models respective to each DC interpreted as output node

Figure 4.4: A high level description of E-ANN architecture.

exploration) and disregard the tuning process completely for solving the dynamic VNF orchestration problem [39]. To ensure a fair comparison with the proposed deep learning-based VNF placement methods taking into account hyperparameter tuning, we have also tuned the exploitation-exploration control variables of t-ACO. Even though, the performance of t-ACO is significantly lower than our proposed ensemble deep learning approaches. Next, we discuss regarding the hyperparameter selection of proposed deep learning models. Then, we define some performance metrics adopted in our experiments to evaluate VNF orchestration strategies. Finally, we demonstrate the simulation results of the ILP Optimization Framework, E-ConvNets, E-ANN, CNN, ANN, and compare with the improved state-of-the-art t-ACO based VNF deployment [39]. ACO has been adopted to solve the dynamic, mobility aware VNF placement problem while considering relocation overhead and QoE in the literature [39]. Simultaneously, other methods discussed in the literature section are either non-QoS optimal or static, ignoring the movement of users. For these reasons, we have chosen t-ACO to compare with the performance of our proposed deep learning-driven VNF placement strategies.

To implement the ILP formulation of the problem, we have used the Gurobi optimization solver. Python's TensorFlow libraries have been utilized to support the experiments concerning our proposed deep learning-based approaches (E-ConvNets and E-ANN). We are not proposing any particular network architecture or routing protocol. Therefore, the experiments do not require to be implemented in the CloudSim or any other kind of network simulator. However, the proposed algorithms can be integrated in these types of simulators as well.

4.7.1 t-ACO : Hyperparameter Tuning of ACO

ACO is one of the swarm intelligence inspired meta-heuristic algorithms. This algorithm has two control parameters α and β to regulate the trend of exploration and exploitation nature over search space [39]. To understand the impact of these parameters over iterations, we plot the best solution (closest to optimal) found by ACO as a single data point in Figure 4.5 for each iteration against best-found values of α and β. To determine each iteration's best solution, we let ACO to try different values of α and β ranging from 1 to 10 and 1 to 5 respectively. Then, the collaborative parameters, α and β, are recorded along with each iteration's best-found solution. The colors of these data points determine the solution's quality illustrating the deviation from optimal in percentage. It is noteworthy from Figure 4.5 that the best choice of α and β values do not remain static rather change over iterations.

Moreover, it can be observed from Figure 4.5 that the solutions converge more toward optimal, along with the advancement of iterations. Figure 4.5 also suggests that during the early stage of the search, the best selected values of α are small to support an extensive exploration of the search space. Then, the best values of α increase over time to enhance the ACO's local searchability (exploitation).

The opposite occurs for β. Thus, for t-ACO, over the iterations, we increase the values of α, starting from 1 to 10. Concurrently, we decrease the values of β from 5 to 1.

Contrarily, the existing ACO in the literature employs fixed value of α and β to be 5 and 1 respectively [39].

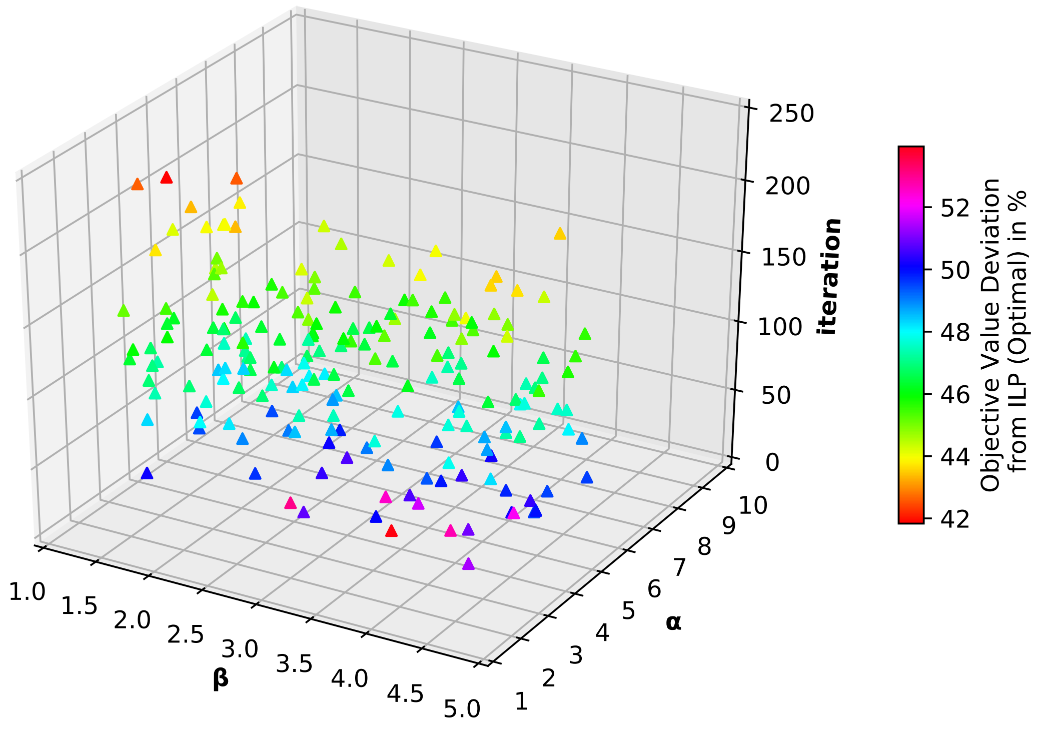


Figure 4.5: Best selected duo of α and β hyperparameters for each solution over iterations. Each data point represents a solution and the color of the data point express the quality of the solution by considering deviation from ILP (Optimal) in percentage (the lower, the better).

Figure 4.6 illustrates that t-ACO can improve the performance up to 6.3% in terms of objective function value minimization for VNF orchestration compared to ACO [39] mentioned in the literature.

Therefore, we select t-ACO to distinguish the performance evaluation of our proposed VNF placement strategies for the comparison to be just and equitable. The other parameters for these experiments have been discussed elaborately in the subsection 4.7.3 and randomly selected following a normal distribution for each simulation run.

4.7.2 Hyperparameter Selection of Proposed Deep Learning Models

In this subsection, we address the study comprising the hyper-parameter tuning to determine the optimal structure of the deep learning models for both ensemble and

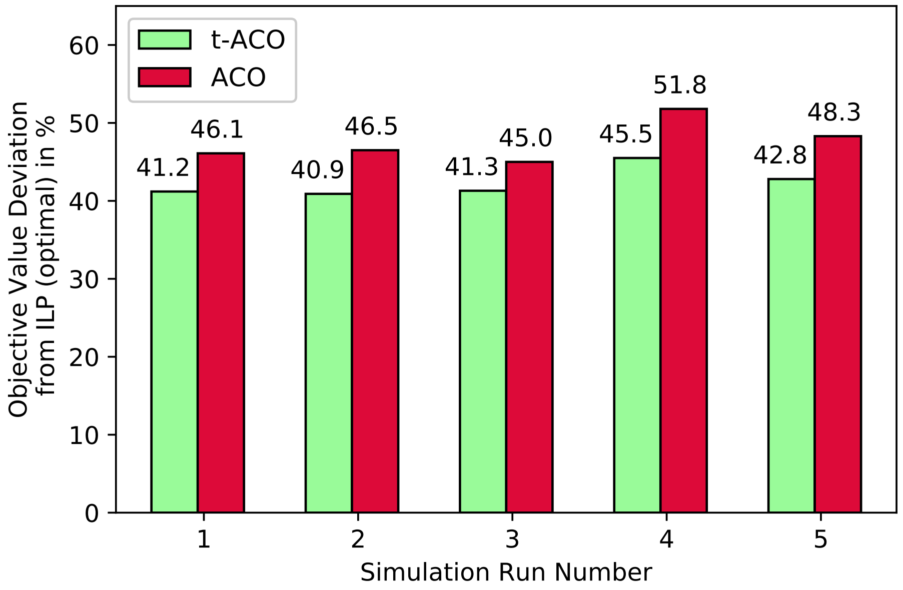


Figure 4.6: Objective value comparison between ACO and t-ACO over different simulation runs (the lower objective value deviation from ILP, the better).

standalone ones. We performed the grid search [107] technique to select the hyperparameters of proposed models. The grid search results have been manifested in Table 4.4 and 4.5, where we outline the best performing combination of hyperparameters against each optimizer for a single layer. We conducted the grid search amidst six broadly utilized optimizers, as listed in both the tables. For electing the activation function, we experimented with a set of four commonly utilized functions: relu, selu, tanh, and sigmoid [18]. Concerning the batch size, we tuned the estimation by applying a set of different equidistant values from 100 to 500 with a gap of 50. In order to select the optimal dropout rate, we examined values ranging from 0.1 to 0.5. We

Table 4.4: Selected parameters of CNN models for each optimizer after employing grid search.

Optimizer	**Selected parameters**					**Deviation from ILP (optimal) in %**
	Activation Function	Kernel Size	Batch Size	Dropout Rate	Epochs	
Adadelta	ReLU	3	450	0.4	5	8.55%
Nadam	ReLU	3	350	0.4	10	8.73%
SGD	ReLU	3	500	0.5	10	9.81%
RMSprop	ReLU	2	450	0.4	10	9.19%
Adagrad	ReLU	2	400	0.3	5	8.56%
Adam	**ReLU**	**3**	**500**	**0.3**	**10**	**7.53%**

Table 4.5: Selected parameters of ANN models for each optimizer after employing grid search

Optimizer	**Selected parameters**				**Deviation from ILP (optimal) in %**
	Activation Function	Batch size	Hidden layer nodes	Epochs	
Adadelta	ReLU	350	50	10	13.34%
Nadam	ReLU	400	35	10	14.23%
SGD	ReLU	350	45	10	15.78%
RMSprop	ReLU	350	45	5	14.57%
Adagrad	ReLU	300	40	5	15.43%
Adam	**ReLU**	**450**	**50**	**10**	**12.76%**

varied the number of epochs using ten values extending from 1 to 10. For the CNN model, the kernel size employed for tuning has been considered from 2 to 5. Moreover, the number of nodes in hidden layers has been studied from 30 to 60 by 5 differences.

The best performing combination for CNN models is obtained for the adam optimizer along with activation function ReLU, kernel size 3, batch size 500, dropout rate 0.3, and the number of epochs 10 as depicted in Table 4.4. Hence, for further experiments of standalone and ensemble CNN models (CNN and E-ConvNets), we employed these hyperparameter values. Likewise, for the ANN models, the adam optimizer, activation function ReLU, batch size 450, hidden layer nodes 50, and epochs 10 appear as best hyperparameters set represented in Table 4.5.

After the fixation of hyperparameters for each layer, we experimented with tuning the number of CNN layers and hidden layers in the ANN model. According to Figure 4.7, the number of optimal convolution layers has been considered as 4. Moreover, Figure 4.7 illustrates that the number of hidden layers set as 3 ensures the best results for ANN models considering being close to ILP (optimal).

All the hyperparameter selection process have been conducted using 5-fold cross-validation to enhance model's generalization abilities. Furthermore, we have used the same hyperparameters to compile and train each model in order to retain simplicity. To implement all the DL models, we have used python's TensorFlow packages.

4.7.3 Simulation Environment

For our experimental analysis, we have studied a network consisting of 12 cloudlet DCs. These DCs are heterogeneous in terms of capacity, hence can host up to a certain number of VNFs. The number of eNBs under each cloudlet DC can differ

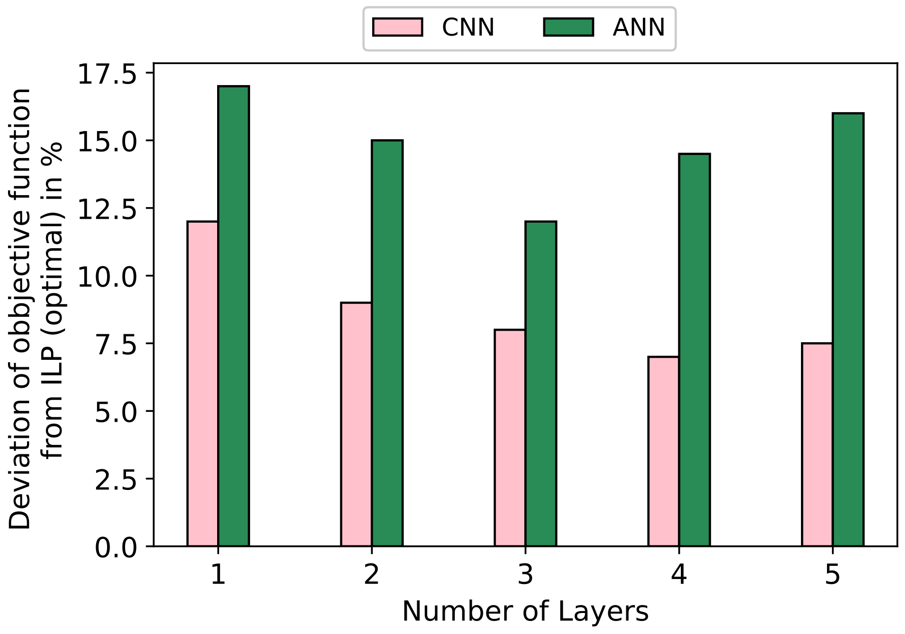


Figure 4.7: Number of CNN layers and hidden layers selection for CNN and ANN models, respectively.

in the range of 5 - 25, while the number of VNFs under each eNB can be between 500 - 2500. The communication delay between different pairs of cloudlet DCs can vary between 10 - 200 milliseconds. However, to get service from the cloudlet DC directly connected to the respective eNB, a trivial amount of time ranging between 2 - 5 milliseconds has been considered. We assume the data rate of transferring VNFs between distinct pairs of cloudlet DCs to be 1 - 50 Mbps, and the size of VNFs are allowed from 100 to 300 KB. The value of priority factor η for minimizing the overall relocation costs has been selected as 0.7. All the mentioned network parameters have been adapted from the existing literature [39].

4.7.4 Performance Metrics

The performance metrics analyzed for the evaluation of different VNF deployment strategies have been described in the following:

- *Total Weighted VNF Relocation and Communication Costs Deviation from ILP (Optimal) in %:* This metric can be considered as an interpretation of how much the objective function value mentioned in the equation Eq. (4.5) deviates away from ILP (optimal) in percentage for each VNF placement strategy. It provides an overall idea concerning to what extent the swarm intelligence based methods (t-ACO), ensemble, and standalone deep learning-based approaches (E-ConvNets, E-ANN, CNN, ANN) can accomplish its objective comparing to the ILP optimization framework. The lower the VNF orchestration strategy achieves the percentage deviation from ILP, the more it is considered to be efficient for providing near-optimal placement solutions.

- *Running Time*: By reporting the execution times of algorithms, we can recognize how quickly a VNF placement method can extend its orchestration services to the users.

- *Number of VNF Relocation*: The total number of VNF relocations required to migrate the VNFs to the selected cloudlet DC is defined by this metric, which directly impacts the administration of the whole network.

- *Scalability Intelligence Factor*: This metric depicts the scalability power of our proposed deep learning assisted methods. Due to the predictable exponential growth of IoT devices in 6G cellular networks, the VNF orchestration technique has to perform equally well on different kinds of network arrangements. Hence, to calculate this metric, we have trained the models on dense and sparse networks individually and tested on the other way around to illustrate how much the performance of these models deviate from the optimization framework. Then, we have normalized the resultant metric values between 0 and 1.

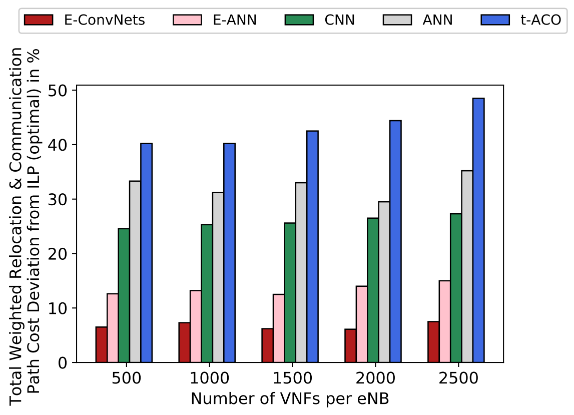
  To generate a sparse network system, we have varied the communication delay 10 times more than the dense networks. Concerning the dense networks, we have considered the same parameters mentioned and utilized throughout the entire experiments.

4.7.5 Results and Discussion

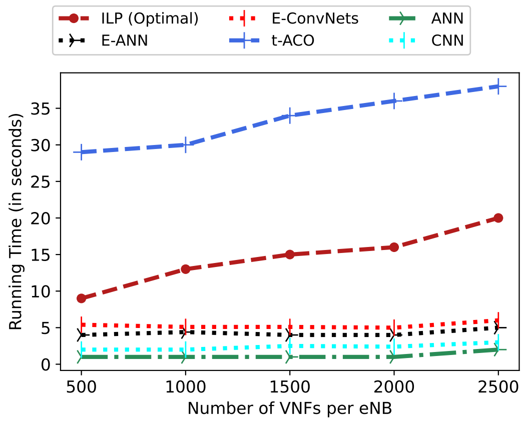
We have categorized the analysis of the results into two kinds. A brief description of these simulation results have been provided below:

Varying number of VNFs under each eNB

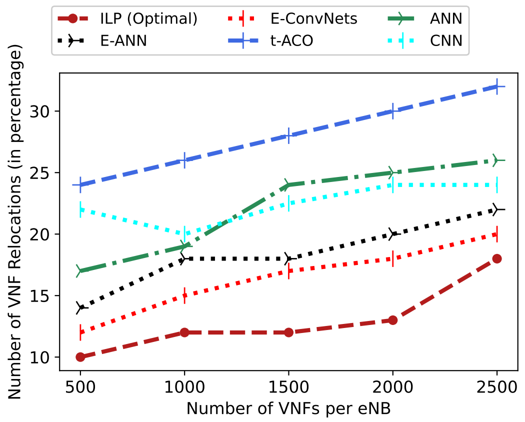
Figure 4.8 represent the effects on the mentioned performance metrics associated with the experiment of varying the number of VNFs under each eNB. From Figure



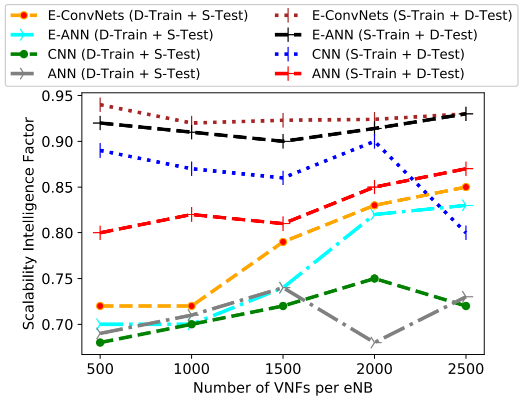
(a) VNF relocation and communication costs deviation from ILP (optimal) in percentage



(b) Running time comparison



(c) Number of VNF relocations (%) comparison



(d) Scalability intelligence factor comparison

Figure 4.8: Comparison of the performance impacts of different VNF deployment strategies for varying number of VNFs under each eNB for 12 data centers in total. In case of Figure 4.8d, S and D represent sparse and dense networks respectively.

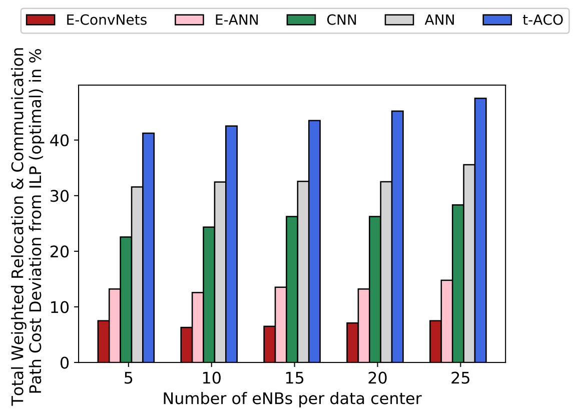
4.8a, it can be easily seen that the t-ACO based deployment is the worst performing one irrespective of the number of VNFs. Being a minimization problem, the higher the objective value, the lower the performance is considered. However, the ensemble strategies E-ConvNets and E-ANN continue to provide better performances through the increasing number of VNFs comparing to standalone CNN and ANN models. E-ConvNets exhibit the most promising objective function value resembling the ILP formulation for all the cases. The running time of ANN has been shown to be the lowest in 4.8b, and CNN manifest very similar execution time as well. E-ConvNets and E-ANN require trivial amount of additional running time than standalone models, yet significantly lower than ILP and t-ACO. Since, the traditional ILP and t-ACO based approaches require higher running time to take VNF deployment decisions comparatively, these are not suitable for latency sensitive real-time IoT applications. The number of VNF relocations affect the migration overhead of networks that have been presented in the Figure 4.8c. E-ConvNets and E-ANN incur migrations ranging between around 12% - 20%. The standalone ANN and CNN models seem to cause 16% - 25% migration overhead. However, ACO based placement strategy induces relocations above 25%, while the optimal percentage of relocations shown by ILP remains approximately 10% - 15% for different numbers of VNFs. For future networks, orchestration systems demand scalability. Hence, to support the scalability experiments of ensemble and standalone deep learning models, we train the models in different settings of sparse and dense networks and test their performance on vice-versa. Figure 4.8d illustrates that all the deep learning models perform significantly well when they are trained using a sparse network. Specifically, E-ConvNets incorporated with sparse training present high scalability intelligence factor of around 0.95.

Varying number of eNBs under each cloudlet DC

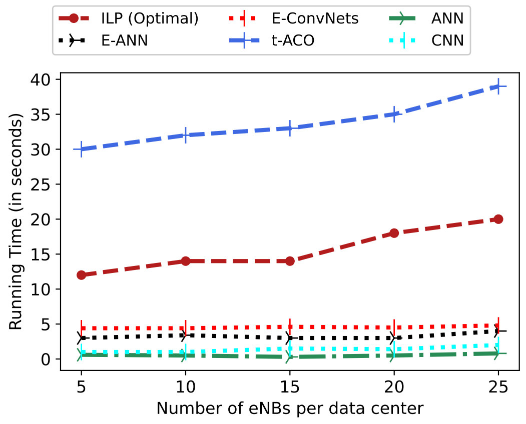
Figure 4.9 illustrates the performance impacts due to the varying the numbers of eNBs from 5 to at most 25 under each cloudlet DC, while keeping the number of VNFs fixed at 1000. The results found in this experiment somewhat resemble the ones found in the earlier simulations represented in this chapter.

The ensemble VNF placement strategies (E-ConvNets and E-ANN) are able to accomplish their goal significantly better than the standalone models (CNN, ANN) and t-ACO based method by minimizing the summation of weighted relocation and communications costs, while considering ILP as the baseline for a diverse number of eNBs as shown in Figure 4.9a. However, E-ConvNets appears to be the most promising approach in terms of being able to perform similar as ILP with least amount of deviation. From Figure 4.9b, it can be easily observed that all the deep learning empowered methods (E-ConvNets, E-ANN, CNN, and ANN) undertake considerably lower the running time that makes these methods feasible to offer real-time VNF placement solutions. On the other hand, t-ACO based and conventional ILP implementations require much longer time to deliver services to the users comparatively.

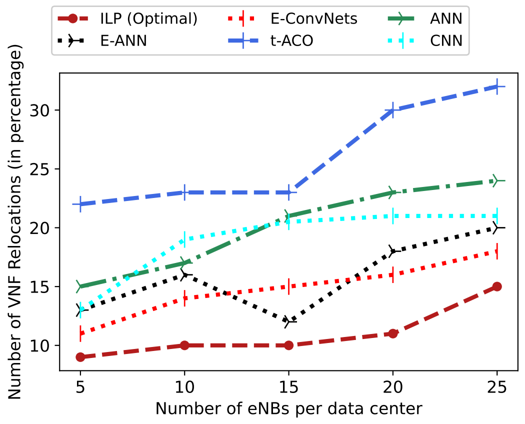
The percentage of relocations in the case of E-ConvNets and E-ANN seem to differ by around 10% from optimal scenarios as represented in Figure 4.9c. The standalone models (ANN and CNN) cause $13\% - 15\%$ more relocation overhead than



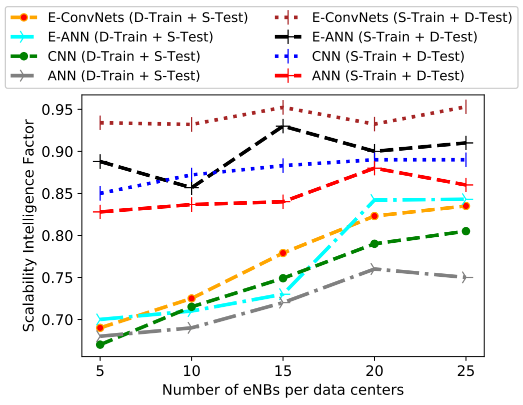
(a) VNF relocation and communication costs deviation from ILP (optimal) in percentage



(b) Running time comparison



(c) Number of VNF relocations (%) comparison



(d) Scalability intelligence factor comparison

Figure 4.9: Comparison of the performance impacts of different VNF deployment strategies for varying number of eNBs under each data center for 12 data centers in total. In case of Figure 4.9d, S and D represent sparse and dense networks respectively.

ILP and $3\% - 10\%$ more comparing to the ensemble ones. Nevertheless, the t-ACO dependent VNF strategy incurs at least 20% extra VNF relocations comparing to optimal solutions. Next, Figure 4.9d illustrates that E-ConvNets trained on sparse networks have been proven to be functioning effortlessly well rather than all the other considered models in context of various network environments with respect to the scalability intelligence factor. This factor differs between 0.7 - 0.9 range for other models due to variety of training.

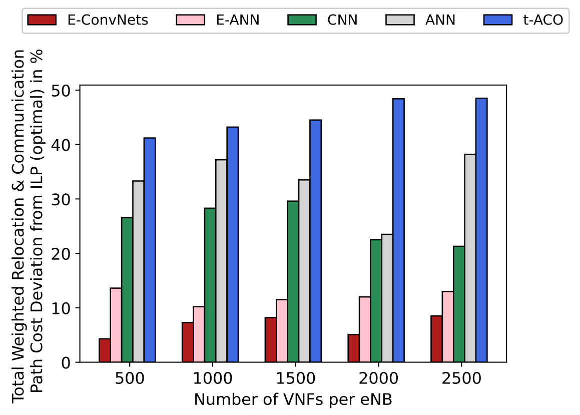
These results justify the requirement for ensemble training and prediction rather than relying on the decisions of an individually trained solo model. Since, the nature of networks can be extremely dynamic and influence the VNF placement result of standalone models to be deviated away from ILP (optimal) more often than ensemble ones. Therefore, from the results, it can be undoubtedly deduced that the E-ConvNets model pre-trained on sparse networks emerges as the most effective strategy to provide real-time solutions for VNF placements in terms of minimizing costs and relocation overhead, with ultra-low prediction time, and noticeably enhanced scalability intelligence factor applicable for various dynamic practical settings.

4.7.6 Case study on Generalization

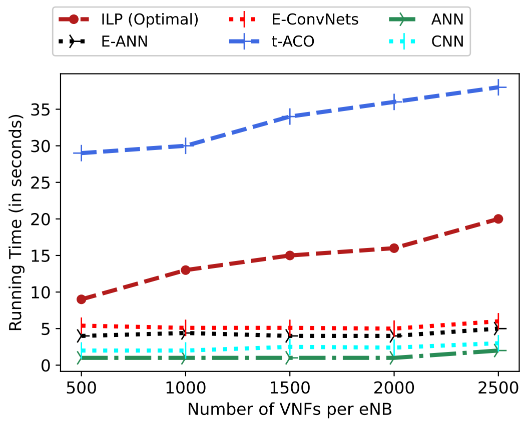
In order to prove the generalization capability of our proposed model, we have utilized the Jisc nation-wide NREN backbone network, as reported by Topology-zoo [108] for this case study. We have assumed cloudlet DCs, each capable of running a limited number of VNFs to simulate provider's NFV infrastructure at randomly chosen points of presence of the Jisc network topology. We have modeled the topology into our system model according to previously discussed simulation environment. Then, for various number of VNFs, we analyze the performances of our proposed pre-trained (on random networks) models on Jisc topology. As per the results presented in Figure 4.10, our proposed E-ConvNets and E-ANN models exhibit and maintain substantially well performance on Jisc network, even though the models are pre-trained on completely different random networks.

4.7.7 Generalization Settings: Bias-Variance Trade-off Analysis

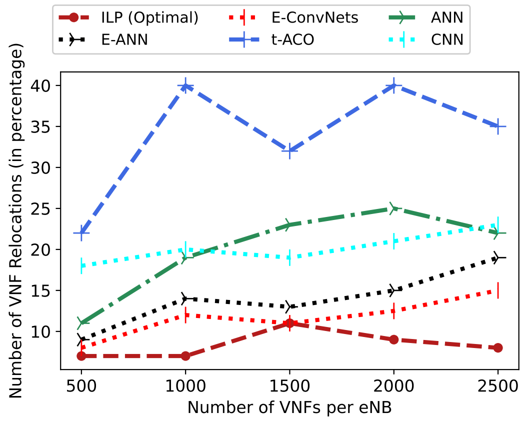
To further explain our proposed models' generalization assurance, we have studied the bias-variance trade-off effects through experiments. The prediction error bias reports the differences between the model's average prediction and actual (optimal) values. Variance refers to the dispersion of predictions over actual values due to different training datasets. This metric helps to evaluate the model sensitivity towards various training observations. Any model with high variance leads to overfitting training data and cannot generalize on unseen test instances. As a result, high bias and variance result in higher training and test errors. To maintain performance consistency in both training and test cases, low bias and variance are desirable. The bias-variance dilemma is the dispute to simultaneously minimize these two aforementioned prediction errors that limit models to generalize beyond training instances.



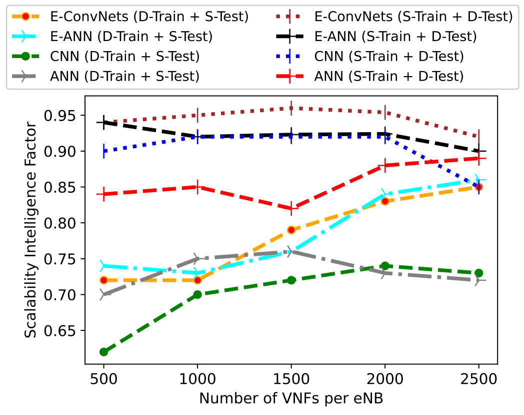
(a) VNF relocation and communication costs deviation from ILP (optimal) in percentage



(b) Running time comparison



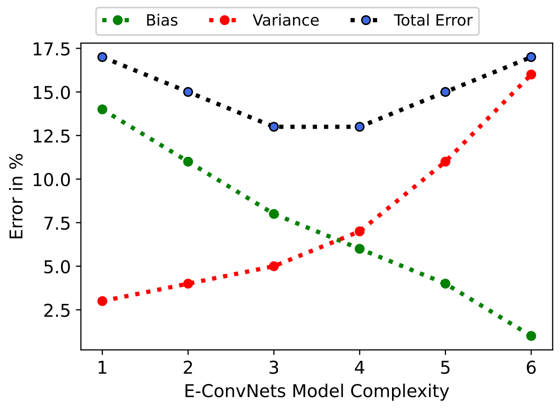
(c) Number of VNF relocations (%) comparison



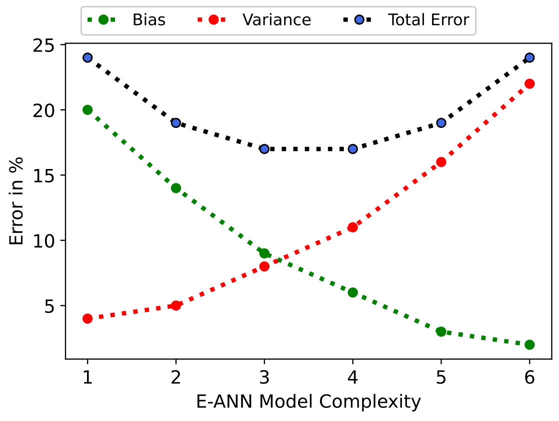
(d) Scalability intelligence factor comparison

Figure 4.10: Comparison of the performance impacts of different VNF deployment strategies for varying number of VNFs in Jisc topology. In case of Figure 4.10d, S and D represent sparse and dense networks respectively.

Theoretically, the total error of any modeling technique has been decomposed as: $Err(x) = (\mathbb{E}[\hat{f}(x)] - f(x))^2 + \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])] + \mu^2 = \text{Bias}^2 + \text{Variance} + \text{Irreducible}$ error. Here, $\hat{f}(x)$ is an approximation of $f(x)$ achieved through a particular modeling or learning technique for any instance x. On ensuring an optimal bias-variance trade-off, a model is proven to achieve generalization that neither underfits nor overfits. To proceed with this study, we have determined the bias and variance of our proposed models with the growing complexity of the models. Again to quantify errors, we have considered the results of ILP (optimal) as ground truth. The complexity of models grows with increasing convolutional layers or hidden layers by expanding the number of trainable parameters proportionately. According to Figure 4.11a and 4.11b, both



(a)



(b)

Figure 4.11: Bias-Variance trade-off analysis for (a) E-ConvNets and (b) E-ANN model

E-ConvNets and E-ANN are able to achieve optimal balance between bias-variance trade-off, while the models are not oversimplified or too complex. To be specific, E-ConvNets and E-ANN minimizes both sources of errors at layer number 4 and 3, respectively. Hence, based on this study, we can confidently conclude that the model is able to learn the underlying pattern from system data and export the learned knowledge on unseen test cases.

4.8 Summary

Recently, the VNF orchestration over resource-constrained infrastructure is gaining much attention from the researchers to emphasize on different optimization techniques. However, the conventional optimization techniques due to the various drawbacks, mostly lacking agility, fail to be qualified for real-time adaptions in dynamic network perspectives. Therefore, we have stressed on designing a prompt technique for intelligent networks to proactively assign VNFs to the edge cloudlets DCs with best possible relocation and communication costs as the outcome, while considering the predictable rapid growth of IoT services in near future. For the sake of the model calibration process, we have considered utilizing multiple models instead of relying on a single one for the training and prediction phase. Hence, we have applied E-ConvNets and E-ANN in simulated network arrangements and compared the results with other existing conventional approaches. Experimental results suggest that E-ConvNets outperforms all other methods in terms of minimizing costs and relocation burden with significantly improved scalability intelligence factor. Although, E-ANN performs best according to running time, yet being very close to execution times of E-ConvNets.

The limitation of this research is not considering resource optimization concerns by incorporating them into the objective function. The main motivation for resource optimization is to improve the battery life of IoT devices and energy efficiency overall. Moreover, this work does not handle the notion of chaining multiple VNFs together to create a specific network service. Thus, in the next chapter, we focus on resource utilities by introducing the sharing concept for chained VNF resources.

Chapter 5

Intelligent SFC Orchestrator for Time and Resource Intensive Ultra Dense IoT Networks

Among the massive pool of IoT devices in NFV context, the urgency for efficient service orchestration is constantly growing. The emerging challenges can be addressed as collaborative optimization of resource utilities and ensuring QoS with prompt orchestration in dynamic, congested, and resource-hungry IoT networks. Traditional mathematical programming models are NP-hard, hence inappropriate for time sensitive IoT scenarios. This chapter promotes the need to go beyond the realms and propose an intelligent DQN driven SFC orchestration, named as DSO hereafter. We further equip this proposed DSO model with the notion of sharing the flow of already deployed network function rather than urging a new instantiation. The sharing conceptualization improves resource utilization, and DQN is employed for adaptive, robust, and swift orchestration. Our extensive simulation results demonstrate the remarkable capability and adaptability of the proposed DSO model for cutting back running time ($\approx$ 10 hours) and ensuring near-optimal resource utilization across extremely dense IoT substrate network settings.

5.1 Introduction

Due to the ever-growing number of IoT devices, the IoT networks have transformed to be multivariate and ultra-dense [109]. Hence, the entire IoT service orchestration system is gradually converting to be unmanageable for traditional network frameworks. NFV [65] appears to be a promising technology for facilitating efficient IoT resource management, scalability, and flexibility. This mechanism decouples the conventional network functions from specially-designed proprietary hardware. Moreover, it allows the telecommunications service providers (TSPs) to implement and run VNFs on top of an IoT network's substrate or physical nodes (e.g., IoT sensors, actuators, controllers, wearables, and smart devices) [110]. IoT users utilize a heterogeneous ordered chain of VNFs deployed at cloud, carrier networks (SGi-LAN), and edge (consumer premises devices) for processing the massive flow to avail complex service chaining [111] [66].

MEC [66] strengthens the SDN [65] for enhanced QoS and context aware delivery for users. This mechanism enables the viability of hosted services to access resource and traffic flow measurements for improving content specific services [66]. As another complimentary benefit, MEC aids monetization aspects for TSPs with effective edge resource utilization. However, the capacity of edge resources and battery life of mobile devices are considerably limited in IoT networks [66]. A sinking battery of edge IoT sensors can be severely damaging, such as causing halt to an industry's production, misinterpretation of valuable data for research, false fire alarms, and life-threatening situations in case of malfunctioning surgical robots and autonomous driving. Hence, the optimizing the edge resource consumption is mutually beneficial for both TSPs and IoT users. Thus, the SFC orchestrator discussed in this chapter emphasizes the utilization of underused, sharable, and identical type previously deployed VNF instances [112].

An SFC deployment involves selecting a path in the network while simultaneously creating new instances or sharing VNFs and forwarding packets through the chain [111]. An SDN controller [65] is assumed to be responsible for forwarding traffic through the deployed chain by configuring switches of the forwarding plane. The bump-in-the-wire (BITW) [111] technique is supported by both Open Virtual Networking (OVN) and OpenStack Tacker [112]. This mechanism allows communications over a series of logical links for configuring SFC. Several existing literature suggest the use of combinatorial optimization and meta-heuristics for SFC deployment [39] [111] [113] [27]. However, mathematical programming models are NP-hard [84]. On the other side, meta-heuristics involves extensive model-specific hyperparameter tuning that also involves high running time [39]. Therefore, these approaches are not feasible for time-critical IoT applications (e.g., collaborative computing, telesurgery, bio-metric, smart grid, remote machinery, uncrewed aircraft system, and AR/VR) [109] [66] [65]. In this chapter, we propose the employment of sharing based SFC orchestration inspired by advanced Deep Q Networks [55], acknowledged as DSO afterwards. The motivation comes from DQN's successful employment for IoT network management with added facility of ultra-low running time [55]. The major contributions of this chapter have been listed as follows:

- We consider and improvise an ILP formulation to solve sharing based SFC deployment, while ensuring the predefined QoS [112]. Current ILP model proposed in [112] sometimes fails to provide solution even for fairly simple cases. We have fixed the issues with the ILP model and implemented it as baseline (optimal) for our performance metrics.

- Next, we formally prove the sharing based SFC orchestration problem to be NP-hard.

- Then, we implement our proposed DSO−sharing based SFC orchestration scheme harnessing DQN combined with sophisticated techniques (experience and replay, target network, and convolutional neural networks), unlike vanilla Q-learning with convergence insufficiency [55]. This advanced DSO model acts upon effective resource utilization, QoS sustainability, and last but not the least significantly improves running time with the intention to unravel the full potentials of present and future IoT applications.

- Finally, with extensive simulation analysis, we demonstrate the remarkable capability and adaptability of the model for improving running time and resource cost minimization across different IoT substrate network settings.

5.2 Related Work

Various studies focus on a broad plethora of diverse research topics related to individual VNF, such as task scheduling and dependency, offloading, allocation, scaling, and migration of VNFs, etc [69] [110]. Presently, the research regarding the SFC or VNF chain orchestration is in its infancy, while directing towards the consolidation and expansion of a single VNF. The SFC related specific researches are mainly categorized into two types: a) orchestration and b) traffic forwarding path configuration [112] [111] [113] [114]. Mostly, the state-of-the-art works regarding allocation involve traditional mathematical programming models [69] [112] [27]. Farkiani et al. have applied Benders decomposition technique to introduce master-slave ILP formulation for faster solution [27]. Many other research articles, including theirs, solely center around energy awareness, disregarding delay and hardware resource optimization constraints for IoT environment [111]. Other existing works in the literature emphasize QoS requirements and end-to-end (E2E) delay without considering resource overhead [39] [110] [69]. Contrarily, some literature only pay attention towards CPU utilization rate, bandwidth, and traffic flow configuration [113] [111]. Except for the literature [112], no other study demonstrated the SFC placement gain due to sharing the flow of already deployed VNFs to maximize resource utilization, and maintain QoS simultaneously. However, they have proposed an ILP framework, and these ad-hoc combinatorial optimization problems are NP-hard [84] because of its extensive time and computational complexity. Thus, these approaches are not suitable for a time-sensitive IoT platform [109]. With the advent of modern deep learning approaches, many research articles analyze the provisioning of services and resource demands, eventually aiming to reduce overall execution time for service orchestration [114] [113] [55]. Yet, these approaches either consider the QoS and resource optimization targets exclusively, or overlook the demonstration of their performance comparison the with regard to optimal [27] [69] [113]. To bridge the research gap and serve a vertical range of resource exhaustive and time-sensitive IoT use cases [109] [66], we have proposed a paradigm shift to a modern and adaptive DSO model. With the intent to overcome the mentioned shortcomings, we incorporate the concept of sharing based SFC deployment in the proposed DSO model, considering resource and QoS concerns mutually.

5.3 Optimized SFC Orchestration Framework

The IoT substrate network model can be considered as a graph $\mathcal{G} = \{\mathcal{U}, \mathcal{E}\}$ including a set of nodes $\mathcal{U}$ connected by a set of bidirectional links $\mathcal{E}$. The nodes in the network serve as VNF hosting devices. Since resources and traffic flow are heterogeneous in the IoT environment, each node and link has its specific computing resources or traffic requirements. In order to deliver a specific service, an ordered set of chained VNF instances is formed as an SFC request. Every instance from the set of VNFs $\mathcal{V}$ has a particular type along with different computational requirements for processor and memory. Upon the allocation of required resources, the maximum traffic flow that can be handled by each VNF varies. The maximum flow a VNF can control is directly proportional to the total resources assigned to it [112]. Sometimes VNFs (e.g., parental control, firewalls, and video optimizer) cause packets to drop, reflected in outflow comparing to inflow. Otherwhiles, for VNFs excluding dropping characteristics (e.g., load balancer), the amount of outflow remains exactly the same as inflow. The outflow of preceding VNF is considered as the inflow of the next consecutive VNF in the chain. For example, the load balancer virtual function has more CPU and RAM allocated to it. Hence the maximum flow handled by this VNF instance is 285, being comparatively greater than the other ones, as shown in Figure 5.1.

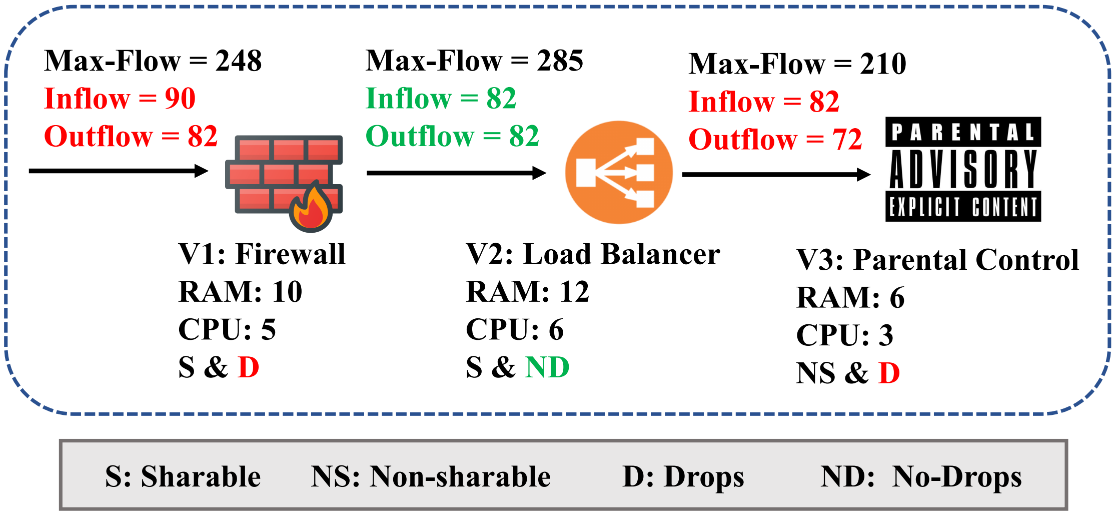


Figure 5.1: VNF specifics of a SFC request

The SFC deployment framework has been formulated and proposed as an ILP model with binary decision variables and some quadratic constraints in the existing literature [112]. However, this formulation has some issues involved and fails to provide an SFC deployment solution, even for simple cases due to erroneous model constraints. The consecutive node verification constraint (7) of their ILP model [112] leads to infeasibility. This can happen due to any consecutive node pairs that are not qualified to host two VNF pairs, while other solution exists. Firstly, we resolve the ILP formulation related issues to ensure the satisfiability of SFC requests whenever possible. Then, we discuss the improvised ILP model and prove the formulation to be NP-hard [84], addressing it unacceptable for a real-time or time-sensitive IoT ecosystem. Table 5.1 contains parameters with a concise description essential to formulate the ILP model.

The objective of this problem formulation aims to either deploy or share VNFs of an SFC so that the overall resource utilization (CPU, RAM, and bandwidth costs) is minimized. Since deploying a new instance of VNF requires more resources (CPU, RAM, and bandwidth), the objective function defined in Eq. 5.1 encourages the sharing based VNF allocation. The sharing of VNF instances only require bandwidth consumption disregarding the demand for additional CPU and RAM resources, unlike new VNF instantiation. The first term of the objective function determines the computational (CPU and RAM allocation) costs in case a VNF is deployed. The latter term defines the bandwidth costs required for both sharing or deploying a VNF.

$$minimize \sum_{i=1}^{|\hat{\mathcal{S}}_j|} \sum_{u_n \in \mathcal{U}} \{(\eta_c \times cpu_i^j + \eta_r \times ram_i^j) \times x_{i,n}^j + \mathcal{O}_i^j \times \eta_b \times (x_{i,n}^j + y_{i,n}^j)\} \tag{5.1}$$

The objective function presented in the Eq. 5.1 is subject to the following constraints:

$$\mathcal{C}1 : \sum_{u_n \in \mathcal{U}} (x_{i,n}^j + y_{i,n}^j) = 1,\ \forall_{i \in [1-|\hat{\mathcal{S}}_j|]} \tag{5.2}$$

$$\mathcal{C}2 : \sum_{u_n \in \mathcal{U}} (x_{i,n}^j + \pi_{i,n} \times \varrho_i \times y_{i,n}^j) = 1,\ \forall_{i \in [1-|\hat{\mathcal{S}}_j|]} \tag{5.3}$$

$$\mathcal{C}3 : \mathcal{I}_i^j \times y_{i,n}^j \ \leq\ f_{i,n},\ \forall_{i \in [1-|\hat{\mathcal{S}}_j|]} \tag{5.4}$$

Table 5.1: Description of the parameters for SFC orchestration

Parameters of Substrate Model	
Notation	**Description**
$\mathcal{G} = \{\mathcal{U}, \mathcal{E}\}$	Topology of the substrate network
$\mathcal{U}$	Set of all the VNF hosting substrate nodes $\{u_1, u_2, ..., u_{\mathcal{U}}\}$
$\mathcal{E}$	Set of bidirectional links in substrate network
$\mathcal{C}_n$	Total CPU capacity of a node $u_n \in \mathcal{U}$ (in cores)
$\mathcal{M}_n$	Total memory capacity of a node $u_n \in \mathcal{U}$ (in GBs)
$\tilde{\mathcal{C}_n}$	Available CPU of a node $u_n \in \mathcal{U}$ (in cores)
$\tilde{\mathcal{M}_n}$	Available memory of a node $u_n \in \mathcal{U}$ (in GBs)
$\mathcal{D}_{nn'}$	The propagation delay of a link connection from a node u_n to another node $u_{n'}$, where $u_n, u_{n'} \in \mathcal{U}$ and $n \neq n'$
$\mathcal{B}_{nn'}$	Total bandwidth capacity of link form a node u_n to $u_{n'}$, where $u_n, u_{n'} \in \mathcal{U}$ and $n \neq n'$ (in Mbps)
$\tilde{\mathcal{B}_{nn'}}$	Available bandwidth of link form a node u_n to $u_{n'}$, where $u_n, u_{n'} \in \mathcal{U}$ and $n \neq n'$ (in Mbps)
SFC Request and VNF Related Parameters	
Notation	**Description**
$\hat{\mathcal{S}}_j$	A SFC request where $\hat{\mathcal{S}}_j \subseteq \mathcal{V}$
$\lvert\hat{\mathcal{S}}_j\rvert$	Length of a SFC $\hat{\mathcal{S}}_j$
$\mathcal{V}$	Set of all VNF instances $\{v_1, v_2, ..., v_{\mathcal{V}}\}$
cpu_i^j	Required CPU for deploying VNF $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$
ram_i^j	Required RAM for deploying VNF $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$
$\mathcal{F}_i$	Maximum controllable flow of VNF $v_i \in \mathcal{V}$
δ_i	Binary input flag to denote whether a VNF $v_i \in \mathcal{V}$ drops inflow or not
ϱ_i	Binary input flag to denote whether a VNF $v_i \in \mathcal{V}$ is shareable or not
$\mathcal{I}_i^j$	Inflow of VNF $v_i \in \mathcal{V}$ to satisfy SFC $\hat{\mathcal{S}}_j$
$\mathcal{O}_i^j$	Outflow by VNF $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$
Θ_j	Maximum tolerable E2E delay threshold of SFC $\hat{\mathcal{S}}_j$
Constants, auxiliary, and decision variables	
Notation	**Description**
$x_{i,n}^j$	Decision variable for deploying a new instance of VNF $v_i \in \mathcal{V}$ belonging to SFC $\hat{\mathcal{S}}_j$ at node $u_n \in \mathcal{U}$
$y_{i,n}^j$	Decision variable for sharing the flow of VNF $v_i \in \mathcal{V}$ belonging to SFC $\hat{\mathcal{S}}_j$ with previously deployed same type VNF at node $u_n \in \mathcal{U}$
$\pi_{i,n}$	Binary input parameter indicating if a similar type instance of VNF $v_i \in \mathcal{V}$ has been deployed at node $u_n \in \mathcal{U}$ or not
$f_{i,n}$	Unconsumed available flow of already deployed VNF $v_i \in \mathcal{V}$ at node $u_n \in \mathcal{U}$
η_c, η_r, η_b	Unitary costs of CPU, RAM, and bandwidth

$$\mathcal{C}4: \sum_{i=1}^{|\hat{\mathcal{S}}_j|} cpu_i^j \times x_{i,n}^j \ \leq \ \tilde{\mathcal{C}}_n, \ \forall_{u_n \in \mathcal{U}} \tag{5.5}$$

$$\mathcal{C}5: \sum_{i=1}^{|\hat{\mathcal{S}}_j|} ram_i^j \times x_{i,n}^j \ \leq \ \tilde{\mathcal{M}}_n, \ \forall_{u_n \in \mathcal{U}} \tag{5.6}$$

$$\mathcal{C}6: (x_{i,n}^j + y_{i,n}^j) \ \leq \sum_{\{u_n, u_{n'}\} \in \mathcal{E}} (x_{i+1,n'}^j + y_{i+1,n'}^j), \forall_{u_n, u_{n'} \in \mathcal{U}}, \ \forall_{i \in [1-|\hat{\mathcal{S}}_j|]} \tag{5.7}$$

$$\mathcal{C}7: \sum_{u_n \in \mathcal{U}} \sum_{u_{n'} \in \mathcal{U}} \mathcal{O}_i^j \times (x_{i,n}^j + y_{i,n}^j) \times (x_{i+1,n'}^j + y_{i+1,n'}^j) \ \leq \tilde{\mathcal{B}}_{nn'}, \ \forall_{i \in [1-(|\hat{\mathcal{S}}_j|-1)]} \tag{5.8}$$

$$\mathcal{C}8: \sum_{i=1}^{|\hat{\mathcal{S}}_j|-1} \sum_{u_n \in \mathcal{U}} \sum_{u_{n'} \in \mathcal{U}} \mathcal{D}_{nn'} \times (x_{i,n}^j + y_{i,n}^j) \times (x_{i+1,n'}^j + y_{i+1,n'}^j) \ \leq \Theta_j \tag{5.9}$$

$$\mathcal{C}9: x_{i,n}^j, y_{i,n}^j \in \{0, 1\}, \ \forall_{u_n \in \mathcal{U}}, \ \forall_{i \in [1-|\hat{\mathcal{S}}_j|]} \tag{5.10}$$

Constraint $\mathcal{C}1$ ensures the single mapping (placement or sharing) of every VNF $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$ into a physical node. The constraints $\mathcal{C}2$ and $\mathcal{C}3$ verify that in case of sharing, the same type already on-boarded VNF $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$ has to be present at node $u_n \in \mathcal{U}$ with sufficient unconsumed/available flow for presently considered VNF's inflow. For a valid deployment decision $x_{i,n}^j$, constraints $\mathcal{C}4$ and $\mathcal{C}5$ secure the availability of enough computational resources (CPU and RAM). Next, the constraint $\mathcal{C}6$ guarantees the deployment or sharing of any consecutive VNF pair $v_i \in \mathcal{V}$ and $v_{i+1} \in \mathcal{V}$ from SFC $\hat{\mathcal{S}}_j$ to mapped on two such nodes $u_n \in \mathcal{U}$ and $u_{n'} \in \mathcal{U}$ in the substrate graph $\mathcal{G}$ that are connected by direct link or edge. Subsequently, the constraint $\mathcal{C}7$ assures the available bandwidth $\tilde{\mathcal{B}}_{nn'}$ in a link connection of two nodes $u_n \in \mathcal{U}$ and $u_{n'} \in \mathcal{U}$ to be enough for accommodating the outflow generated by VNF $v_i \in \mathcal{V}$ in both the cases of either deployment or sharing. Furthermore, the QoS performance (E2E delay) requirement of an SFC $\hat{\mathcal{S}}_j$ is satisfied by the constraint $\mathcal{C}8$, while preventing the total propagation delay to be overboard beyond a permissible threshold Θ_j. According to application domain's nature and Service Level Agreement

(SLA) [69], this threshold can be carefully set by the TSPs. Finally, the constraint $\mathcal{C}9$ is binary constraint indicating the value of decision variable $x_{i,n}^{j}$ to be 1, in case VNF $v_i \in \mathcal{V}$ of SFC request $\hat{\mathcal{S}}_j$ is deployed at node $u_n \in \mathcal{U}$, otherwise remains 0. Likewise, $y_{i,n}^{j}$ is 1 if a VNF $v_i \in \mathcal{V}$ of SFC request $\hat{\mathcal{S}}_j$ shares the traffic flow of the identical type already deployed VNF at node $u_n \in \mathcal{U}$, else 0.

Theorem 1. *The aforementioned sharing based SFC orchestration problem is NP-hard.*

Proof: Let us consider the 0/1 minimization multiple knapsack problem. Given a knapsack instance $I = (P, K, W, V, \Omega)$, where P is the set of items, W and V represent the set of weights and values for selecting each item respectively. Moreover, K denotes the set of knapsacks, while Ω specifies the set including capacities of each knapsack. Taking another instance I' of the SFC deployment problem, we can map $I' = (P \Leftarrow \hat{\mathcal{S}}_j, K \Leftarrow \mathcal{G}_c, W \Leftarrow H_j, V \Leftarrow \rho_j, \Omega \Leftarrow \theta_{\mathcal{G}_c})$ to I. Here, $\hat{\mathcal{S}}_j$ is basically an SFC request containing an ordered set of VNFs, $\mathcal{G}_c$ represents a clique (to validate constraint $\mathcal{C}6$) substrate graph containing certain nodes, H_j defines the combined required resources (CPU, RAM, and bandwidth) to satisfy $\hat{\mathcal{S}}_j$ once deployed or shared. Seemingly, ρ_j is the set of costs to orchestrate a VNF $v_i \in \mathcal{V}$ belonging to SFC $\hat{\mathcal{S}}_j$, and $\theta_{\mathcal{G}_c}$ is a set of the available resource capacities of each node on substrate graph $\mathcal{G}_c$. Firstly, we apply the restriction $\varrho_i = 0$ in ILP mode, which means that the sharing option is completely disabled for all the VNFs $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$. Then, we apply the restrictions of $\tilde{\mathcal{B}_{nn'}} = +\infty$, $\Theta_j = +\infty$, and $f_{i,n} = +\infty$, leading to ignore the QoS, bandwidth, and flow constraints in the ILP instance I'. Lastly, we consider the outflow of $\mathcal{O}_i^j$ to be 1 for every VNF $v_i \in \mathcal{V}$ of SFC $\hat{\mathcal{S}}_j$ in the considered special case. Hence, the restricted case I' of the ILP model transforms into I, a general case of known NP-hard problem. Thus, the optimization problem for SFC orchestration is NP-hard as well. ■

5.4 DSO: Proposed DQN driven Approach for Sharing based SFC Orchestration

In this section, we present our proposed DSO model for SFC orchestration approach in details. The DSO approach leverages its persuasive mechanisms termed as experience and replay, occasionally frozen target network with convolutional neural networks (CNNs) for robustness, efficiency, and better convergence [55]. Any typical reinforcement learning involves four key components: agent, environment (optional), reward, and value function [55]. The agent's ultimate goal is to maximize the long-term (myopic) rewards by interacting with the environment and looping through observed feedback. A transition in state occurs once the agent takes action depending on the corresponding action's reward value. Besides, value function plays an essential role in determining a state-action pair's goodness by predicting the likely future rewards affiliated with it. Afterwards, we discuss how we have mapped the considered SFC orchestration problem to be unraveled by DQN inspired DSO with elegant training particulars.

5.4.1 State and Action Space

We interpret the state space as currently available resources within nodes, underlying link bandwidths, and other specifications of the substrate IoT network. Therefore, for the DSO based SFC embedding process, we have considered the currently available quantities of resources (CPU, RAM), unconsumed flow, and bandwidths as some of the state configuration elements. Additionally, we take other IoT network parameters into account, such as whether a VNF has already been deployed into some node, the sharing flag, dropping, and flow characteristics of the deployed VNF. The combination of these pieces of information form a vector indicating the present state $s = \{\tilde{\mathcal{C}}_n, \tilde{\mathcal{M}}_n, \mathcal{D}_{nn'}, \tilde{\mathcal{B}}_{nn'}, \mathcal{F}_i, \delta_i, \varrho_i, \pi_{i,n}, f_{i,n}\}$. According to the current state of IoT network dynamics inferred from the environment, the agent selects the substrate nodes to be allotted for VNF deployment or sharing purposes in order to satisfy a particular SFC request. Hence, the action space is defined by the number of nodes in IoT substrate network. The agent is authorized to execute one action at a time step from the action space represented by $A = \mathcal{U}$. As mentioned earlier, $\mathcal{U} = \{u_1, u_2, ..., u_{\mathcal{U}}\}$ is a set of all the VNF hosting IoT substrate nodes. After each valid mapping (feasible action) of a VNF to a substrate node in a time step, the agent observes a state transition (varying resources and IoT network dynamics). A feasible action is a mapping of a VNF to a substrate node, where all the constraints of the ILP model ($\mathcal{C}1$ through $\mathcal{C}9$) are satisfied. Upon the successful embedding of a VNF, the agent proceeds towards the next VNF in the requested SFC provided the updated state space. The entire SFC orchestration process terminates after the orchestration of the last VNF from the requested chain of services. Apparently, the action space may seem to be huge, yet the validation of the consecutive VNF mapping constraints reduce the action space size significantly.

5.4.2 Reward Function Design

Usually, the agent strives to maximize the cumulative reward in the long run. In a general context, the total rewards accumulated through each time step can be represented by $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},\ 0 \leq \gamma \leq 1$, where γ acts as a discount factor. The discount factor $\gamma \in [0,1]$ penalizes the future rewards, since there are various uncertainties involved. With the aid of this factor, a more precise balance between instant and future rewards can be ensured. We consider the reward obtained at each time step to supervise the agent towards a better solution with respect to our objective function described in Eq. 5.1 for every time step. As previously mentioned, this objective function leads to efficient resource utilization and prefer sharing the flow of on-boarded VNFs over deploying a new instance. We define the reward function in Eq. 5.11, which is inverse of the SFC orchestration ILP model's objective function. Thus, maximizing the cumulative rewards over time mimics the minimization of overall resource utilization costs.

$$reward = \alpha \times \{(\eta_c \times cpu_i^j + \eta_r \times ram_i^j) \times x_{i,n}^j + \mathcal{O}_i^j \times \eta_b \times (x_{i,n}^j + y_{i,n}^j)\}^{-1},\ \ \alpha > 0 \tag{5.11}$$

DQN is an off-policy training algorithm that enables the agent to learn through temporal differences [55]. Temporal difference [55] is a very unique technique by replacing the actual complex calculation of future rewards with an estimation or prediction, which is expected to keep improving over the time. In this off-policy training period, the agent ignores to find the best policy rather tries to learn the appropriate Q-function. Say, we denote the state space as $S = \{s_1, s_2, ..., s_m\}$ and action space as $A = \{a_1, a_2, ..., a_n\}$. At the time t, the agent chooses an action $a_t \in A$ depending on the current state $s \in S$ of the environment mode. Then, the system transfers to a new state $s_{t+1} \in S$ depending on the largest Q-value ($\underset{a}{\text{argmax}}\ Q(s_t, a_t)$). However, in actual rather than always maximizing the Q-value, exploration in the action space is emphatically encouraged. We employ an exploration rate of $\epsilon \in [0,1]$ to balance exploitation versus exploration, popularly acknowledged as ϵ-greedy algorithm [55]. According to this algorithm, for every decision-making process, we generate a random number between 0 and 1. In case the generated random number is greater than pre-defined ϵ, the agent intends to maximize Q-value (selects action that contributes to $\underset{a}{\text{argmax}}\, Q(s_t, a_t)$). On the contrary, the agent accepts any randomized action from the action space A. Q-function is actually the action value function intended to be learnt by the agent. The recursive Q-function learning is updated by Eq. 5.12. The update process is completed through the information regarding present time step (s_t, a_t, r_t), following time step $(s_{t+1}, a_{t+1}, r_{t+1})$, and learning rate α. As suggested earlier, the discounting factor γ can be tuned and $\gamma = 0$ tend to focus on instant rewards only, which is undesirable.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[\gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{5.12}$$

Then, We exploit the "memory and replay" [55] mechanism to replay the learning experience saved as tuple (s_t, a_t, r_t, s_{t+1}) in memory mini-batches. Since we let the algorithm bootstrap [55] (sampling with replacement) on previously stored experiences, a sample can be picked up multiples times. Hence, the significant outcome is the disruption of correlation and non-stationary distribution of the observation sequences, eventually leading to more efficient update of CNN. Next, to stabilize the overall training process, we have utilized a target network [55] that is a copy of Q-function. The target network is a CNN, and this network's parameters remain fixed for some training time steps. On the other hand, the parameters of another CNN are constantly being updated. The parameters of both the mentioned CNNs are synchronized after a certain period. Though the target network mechanism adds delay, it prevents the havoc in the training process due to oscillations and divergence created by agent for chasing non-stationary future rewards. Due to the uncertain nature and dynamics of IoT environments, this technique is particularly useful. Suppose, $Q(s_{t+1}, a_{t+1}; w^-)$ is generated by target network with parameters w^-, while the DQN network estimates $Q(s_t, a_t; w)$ with parameters w. Finally, we apply the gradient descent method to update the DQN parameters w by optimizing the following loss function:

$$loss = (r_t + \gamma \max_{a \in A} Q(s_{t+1}, a_{t+1}; w^-) - Q(s_t, a_t; w))^2 \tag{5.13}$$

The entire DQN based DSO algorithm for SFC orchestrator module has been described in Algorithm 7.

Algorithm 7: DSO for sharing based SFC embedding

```
    Input: $\mathcal{G}, \hat{S}_j, \tilde{\mathcal{C}}_n, \tilde{\mathcal{M}}_n, \mathcal{D}_{nn'}, \tilde{\mathcal{B}}_{nn'}, \mathcal{F}_i, \delta_i, \varrho_i, \pi_{i,n}, f_{i,n}, A$
    $\gamma, \epsilon, reset\_limit$
1  Model the environment with IoT substrate nodes, links
2  Set DQN and target network by random $w$ and $w^-$
3  foreach episode $e$ do
4      Initialize state $s$ and counter $\leftarrow 0$
5      foreach timestep $t$ do
6          $\eta \leftarrow$ Pick a random number from [0,1]
7          if $\eta < \epsilon$ then
8              $a_t \leftarrow$ Select a random action from $A$
9          else
10             $a_t \leftarrow \underset{a_t}{\text{argmax}}\ Q(s_t, a_t; w)$
11         if $a_t$ is a feasible action then
12             Perform action $a_t$ by SFC orchestrator according to reward
                function
13             Collect $r_t$ through state transition to $s_{t+1}$
14             Save tuple $(s_t, a_t, r_t, s_{t+1})$ in buffer
15             Bootstrap $(s_k, a_k, r_k, s_{k+1})$ for replay
16             if episode $e$ terminates at $k+1$ then
17                 $z_k \leftarrow r_k$
18             else
19                 $z_k \leftarrow r_k + \max_{a_{k+1}} \hat{Q}(s_{k+1}, a_{k+1}; w^-)$
20             Utilize Gradient Descent to optimize the loss function
                $(z_k - Q(s_k, a_k; w))^2$
21             counter $\leftarrow$ counter $+ 1$
22             if counter $\mod reset\_limit = 0$ then
23                 $w^- \leftarrow w$
```

5.5 Performance Evaluation

We simulate the IoT environment and evaluate different performance metrics of DSO, considering ILP as the baseline. All the experiments have been carried out on DELL ALIENWARE m15 R3 machine of Intel core i7-10750H CPU @2.6 GHz equipped with 16 GB RAM and Windows 10 Home. We have used Gurobi to solve the ILP model. In the simulation environment, substrate IoT networks are created using NetworkX with varying nodes (700-1000), and the connectivity probability of links differs between 0.2 and 1.0. The generated topology is assigned resource capacities randomly from the inclusive ranges of 8-64 CPU cores, 16-128 GB RAM, and 100-1000 Mbps bandwidth, respectively. Also, the propagation delay varies from 50-1000 m on a random basis. Next, the SFC lengths can be any random value between 5 and 25. The total E2E latency is derived from the term $(|\hat{\mathcal{S}}_j| - 0.5)$ multiplied by average link delay. In addition, the resource and flow requirements of VNFs that build up an SFC also have been drawn randomly from pre-defined ranges 2-8 cores for CPU, 4-16 GB for RAM, maximum flow derived as a function of CPU and RAM, and inflow [0.15$\times$maximum flow - maximum flow] Mbps. Furthermore, if a VNF has dropping characteristics, then outflow is between 0.4$\times$inflow and inflow, otherwise exactly the same as inflow. We set the unit costs of CPU, RAM, and bandwidth as 2.5, 1.7, and 2, respectively, ultimately having no effect on results being static for all simulation cases. All the mentioned network parameters have been adapted from the existing literature [112]. The results have been presented as an average of 10 simulation runs.

With the above mentioned settings, we implement the proposed DSO process using python's TensorFlow packages. However, the solutions can be easily integrated in any other network simulator (e.g., CloudSim) via customized interface. We have considered three fully connected layers with 32 nodes for CNNs. All the hyperparameters of the DSO model operating DQN underneath have been selected through hyperparameter tuning and simulation analysis.

Figure 5.2 illustrates the effect of learning rate versus obtained rewards and suggests 0.05 to be the most viable option. Since, this selected learning rate is neither too high to over jump the global optimum nor too low for slower convergence, we have selected 0.05 as learning rate. The loss in the training steps can vary due to batch size as shown in Figure 5.3. Batch size of too large may lead to overfitting, leading to poor local optima. On the contrary, smaller batch size cause variance and slow convergence to minimize the loss using batch gradient descent. According to the experimental results, we have considered batch size as 32 for our experiments. Moreover, the loss leading to as close as 0 proves the efficiency of our proposed DSO approach in terms of achieving convergence. We have set the other hyperparameters as follows: $\epsilon = 0.2$, $\gamma = 0.7$, and episodes=1000.

One of the two performance metrics, total resource utilization costs deviation, denotes how far the overall performance of DSO is from ILP (optimal) in terms of minimizing the objective function. The other metric gives us the idea of how much the running time can be saved or improved by the DSO model comparing to the ILP model [112]. We have performed two sets of experiments. The first one is to explore the effects of varying connectivity probability (the probability that determines

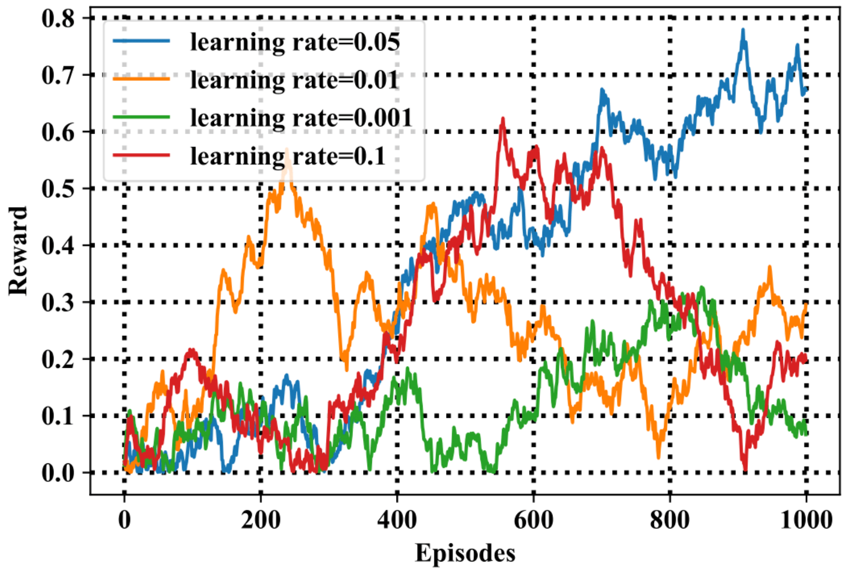


Figure 5.2: Effects of learning rates against increasing episodes

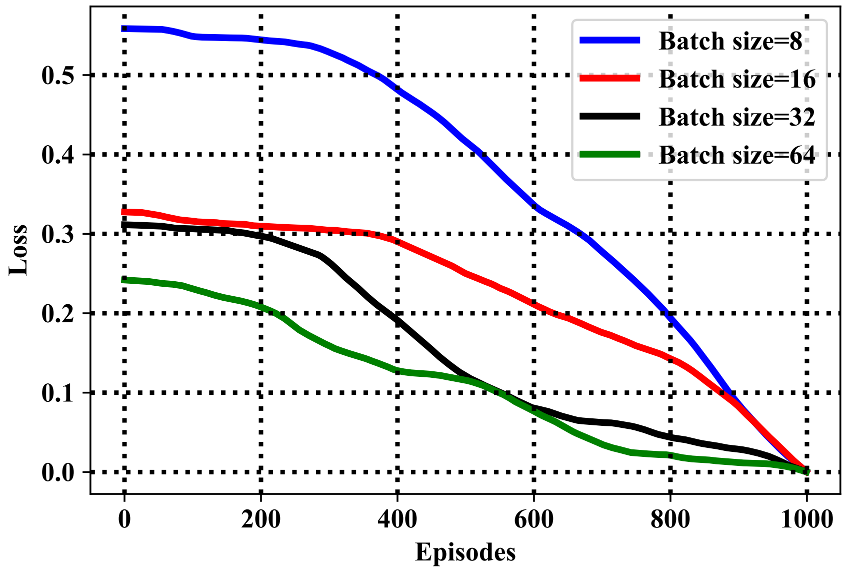
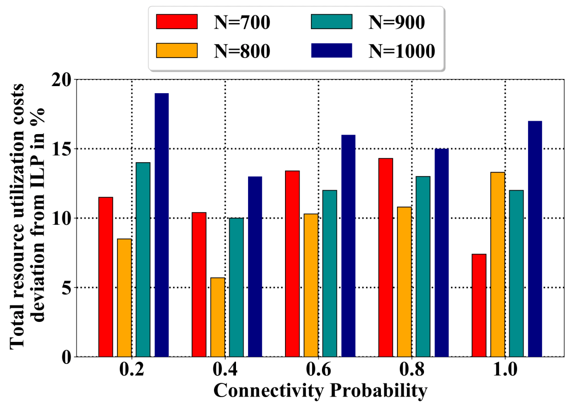
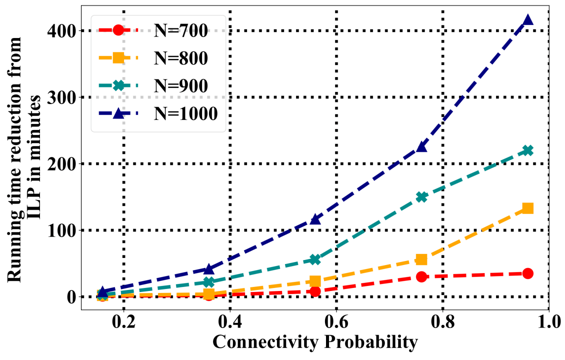


Figure 5.3: Effects of batch size on minimizing loss

if two nodes are connected by a link) among different sized topology. The second one whereas is to evaluate the performances with various lengths of SFCs. Figure 5.4a and 5.4b depict the performance of DSO regarding the first set of experiments, where we have considered network topology with nodes varying from 700 to 1000, and the nodes may have varying connecting probability of links between 0.2 and



(a)

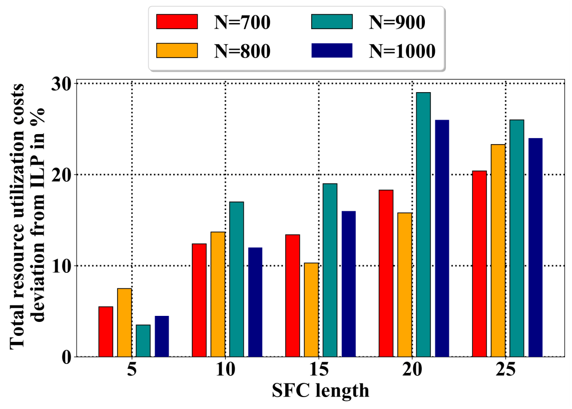


(b)

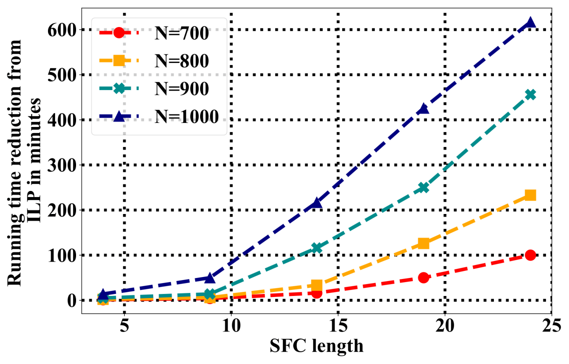
Figure 5.4: (a) Resource utilization costs, and (b) Running time comparison. For (a) & (b), we have considered different density of networks and ILP as baseline for the results shown.

1.0. The connectivity probability 0.2 refers to very sparse graphs, whereas 1.0 forms a clique/ultra-dense IoT substrate graph. Particularly, Figure 5.4a illustrates that resource utilization costs derived by DSO in sparse and dense networks can be from as low as 6% to as high as 19%. However, it is noteworthy from Figure 5.4b that the proposed DSO approach for sharing based SFC orchestration can save up to 400 minutes (approximately 6.66 hours) comparing to ILP in case of large and dense substrate graphs. The graph's exponential trend confirms that in the case of more massive and dense networks, the DSO approach will reduce even more time and be able to extend its fast IoT service to the users.

Figure 5.5a demonstrates that the DSO model deviates from optimal with the growing number of VNFs in an SFC, termed as SFC length. However, according to literature [113] [112] [27], the usual length of SFCs can be around $10 - 15$, for which the DSO derived costs are $12\% - 16\%$ off from ILP. On the other hand, in addition to the near optimal solutions, SFC orchestration with DSO reduces $70 - 600$ minutes worth of running time overall in different network settings. Hence, it can be summarized that regardless of connectivity probability, topology size, and SFC length, DSO provides near optimal solution with significantly reduced time, highly desirable for time critical IoT domain.



(a)



(b)

Figure 5.5: (a) Resource utilization costs comparison, and (b) Running time comparison. For (a) & (b), we have illustrated the effects of different SFC lengths and considered ILP as baseline.

5.6 Summary

Even with fluctuating IoT dynamics, proposed DSO model maneuvers timely SFC orchestration with adaptive resource utilization. This chapter justifies the reliability of intelligent SFC orchestration to serve machine type communications (mMTC), ultra-reliable low-latency communications (URLLC), narrow-band IoT (NB-IoT), enhanced mobile broadband (eMBB), and so forth for 5G and beyond use cases. Additionally, DSO−the sharing based placement incorporated with DQN utilizes the flow of on-boarded VNFs to save massive IoT resources in dense networks, unlike other existing researches. Moreover, our simulation results illustrate the significant running time performance gain for ultra-dense networks (high connectivity probability), counterfeit of future IoT. To the best of our knowledge, crowdsourcing shared VNF resource utilization at a comprehensive pace, QoS cognizant, and prompt SFC orchestration through DQN intelligence has not been approached before. The synergistic concerns prefer the mutual interests of IoT resource manufacturers, TSPs, and IoT users jointly. A future research direction to this work can be integrating SFC orchestration for several categories of users (e.g., prioritizing premium subscribers over the regular charged users).

Chapter 6

Conclusion and Future Work

In this thesis, we proposed the application of machine learning techniques in future-generation wireless network optimization problems. We mainly studied the implications of three AI based techniques: a) supervised learning, b) meta-heuristics, and c) reinforcement learning. For supervised learning, a labeled dataset should be available consisting a collection of input and target pairs. Since there were no standardized labeled datasets, we leveraged the optimal solutions generated by ILP for our proposed supervised learning methods (ANN, E-ANN, and E-CNN). Meanwhile, ACO, a meta-heuristic algorithm, imitated the collective behavior of ants to effectively invade search space. On the other side, we proposed DSO as an intelligent orchestrator that learns through experiences via interaction with environment. The environment of our considered system was virtual service management in IoT context.

The first ANN based static VNF orchestration model simulated a large number of scenarios through ILP. Then, the simulation model attempted to convert the VNF and network attributes as input and corresponding optimal hosts as target pairs to enable supervised learning. Due to the explosive number of output state space (feasible VNF hosting devices), reinforcement learning was prone to perform poorly in such problems. Then, we introduced ensembling techniques to improvise variety of KPIs (e.g., scalability, relocation overhead, communication costs, and running time) for a mobility aware VNF orchestrator. We also strived to apply t-ACO (improvised meta-heuristic) to this problem, yet this algorithm exhibited larger optimality gap and running time due to scenario-specific hyperparameter tuning. Ensemble and standalone pre-trained ANN and CNN models fixed optimal hyperparameter on validation sets, hence there was no requirement for tuning at test cases. Finally, our proposed DSO intended to serve SFC orchestrators that targeted reduced feasible region specific problems, with significantly lower state space. DSO mainly aimed to optimize the total resource consumption by enabling the services to share the resources of likewise on-boarded VNFs.

Additionally, in this thesis, we strived to transform the running time complexity of latency sensitive next-generation service orchestration from exponential to linear. All of our above-mentioned proposed AI algorithms had shown linear inference time complexity with reasonable optimality gap. On the contrary, the time complexity of t-ACO (meta-heuristic) approach grows in quadratic manner. Table 6.1 summarizes the specifics of the methodologies used in this thesis:

Table 6.1: Summary of the proposed AI-based algorithms

	ANN	**E-CNN, E-ANN**	**DSO**
Orchestrator handler	Standalone VNF	Standalone VNF	Service function chaining
Type	Supervised Learning	Supervised Learning	Reinforcement learning
Design philosophy	Optimize through ILP	Optimize through ILP; Meta-heuristics	Optimize through interaction and experience
Objective type	Single-objective	Multi-objective	Multi-objective
Objective criterion	Minimize the total latency	Minimize total communication and relocations costs	Optimize resource utilization
KPIs	Latency; Running time	Total communication and migration costs; Relocation overhead; Scalability intelligence; Running time; Generalization Aptitude	Model loss; Resouce utilization; Running time
Time Complexity	Linear: $O(h)$; h being the number of hidden layers including softmax (Inference time)	Linear: $O(ch)$; c is a constant number of ensemble models (Inference time)	Linear: $O(w)$; w being the DQN weights (Inference time)

We conclude this thesis by putting forward some details on future research directions and potential challenges involved with AI enabled service management IOT framework. Federated learning (FL) can enhance service management's efficiency by taking it to the next level for automated provisioning as a future research direction to this thesis work. FL is a collaborative machine learning approach for which the edge devices conduct the training phase locally rather than the cloud [57]. FL has already been implemented in the Google keyboard (Gboard), which locally stores the contemporary context information and suggestions accepted on the device [57]. Then, the historical data are being utilized to improve the local training by integrating the small focus updates to the trained model for succeeding suggestion scheme. This is a distributed machine learning approach, where generic models from the cloud are downloaded and customized at different edge devices. Then, the sub-models keep training on each edge device locally. Finally, the sub-models of the edge devices collaborate to enhance the performance of the generic shared model in the cloud. This is done by updating the locally trained and improved sub-models from all the edge devices to the cloud.

This approach eliminates the need for centralized training data in one machine or cloud data center by bringing the training and prediction in local sub-models present at the edge devices unlike conventional machine learning methods [57]. Even, FL is not simply a re-branded distributed learning. FL is different from the concept of on device automated reply suggestions and mobile vision API (facial features detection) that only use local sub-models of hosting devices for prediction purposes, and not for training [57]. FL can highly benefit three major aspects of IoT industry:

- *Security*: The decentralized FL breed of AI removes the obligation to send over data, rather only updates model weights to the cloud. This way or learning with model increases the chances of IoT end devices to be secured from personal data breach.

- *Advanced fault tolerance*: In case of communication failures (e.g., dropped updates) and byzantine updates, there has been several research works that uphold the ability of FL to trim the faulty bits. Various robust server update policies can manage the service orchestration in more efficient manner.

- *Mobile Edge Learning (MEL)*: MEL is an emerging parallel learning framework to learn the attributes of users to improve customized experience. This can be achieved by the federated version on-device training to provision the VNF requirements and chaining at different IoT edge devices. For the training purpose to learn VNF orchestration and management strategies, data concerning the users' personalized interaction with the devices can be utilized. Such VNF placement and service chaining schemes can level up the game for new age IoT devices to create more intuitive experiences for users, while increasing the sales for vendors and service providers as well.

Figure 6.1 illustrates the overview workflow of VNF placement and service chaining at IoT environment based on the FL mechanism.

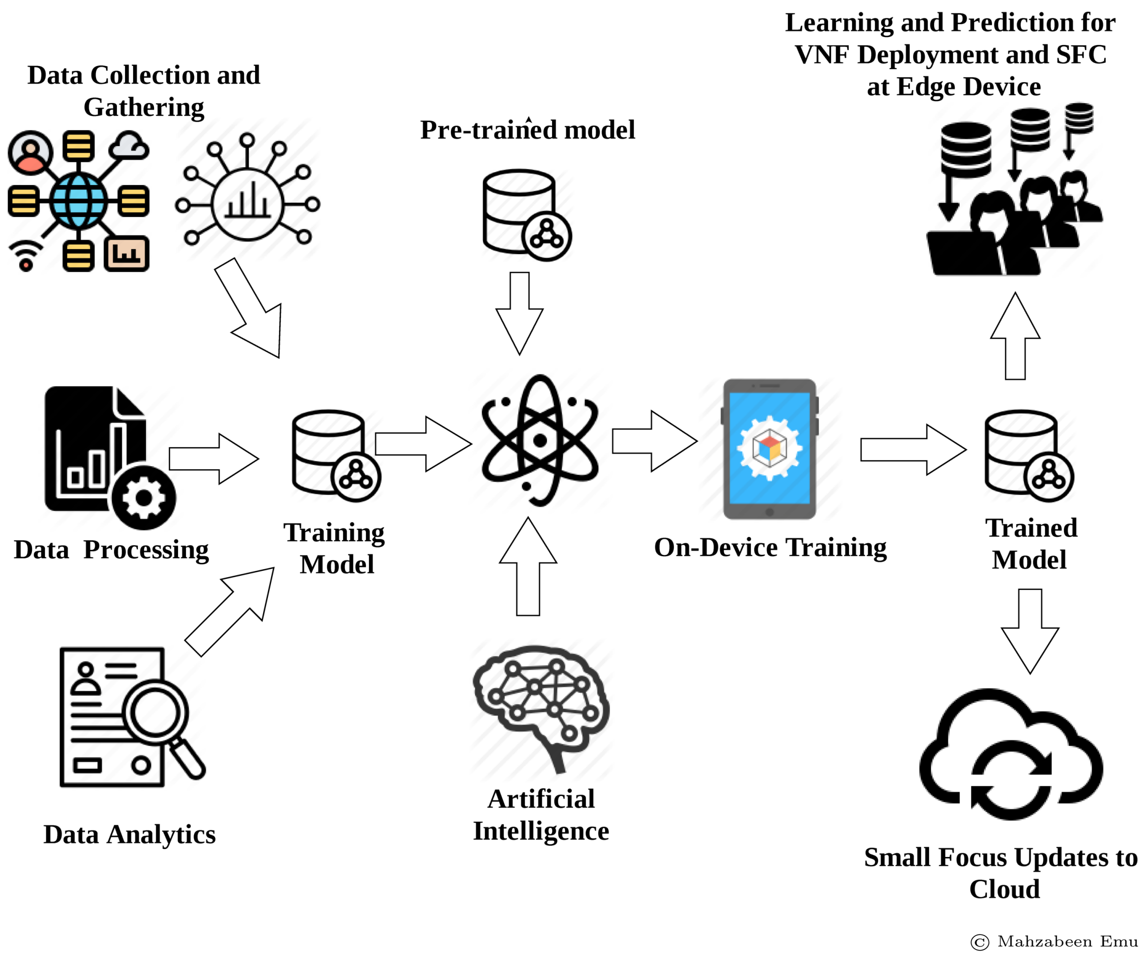


Figure 6.1: VNF and SFC deployment in context of various IoT domains

Neural networks, DQN, and FL can be promising approaches to manipulate the automation of VNF orchestration and service chaining for the next generation IoT services. However, there are some challenges associated with these advanced learning techniques that are required to be addressed. One of them is the demand for extensive GPU resources. GPU with Compute Unified Device Architecture (CUDA) programming framework are ideal for neural networks and deep learning methods. To train VNF orchestrator and SFC models at the edge IoT devices, extensive hardware design equipped with proper GPU resources is one of the most anticipated breakthroughs in the following years to come [54]. While increasing the computational efficiency, it is also necessary to keep track of the added hardware expenses to balance the overall performance-cost ratio, particularly for progressively growing competition in the IoT marketplace.

Another issue is with the ongoing debate regarding centralized and distributed learning. There is no specific answer that justifies the use of either a centralized (DQN, Neural Networks) or distributed learning approach (FL). While distributed training at edge IoT devices offer ultra-low latency and reduced privacy concerns, it comes with the cost of employing sophisticated hardware for vendors with raising the concerns of profitability for providers and reasonable pricing for customers. However, centralized learning increases the possibilities of the unnecessary signaling overhead, placement delay, and relatively less appreciated user experience. The trade-off here requires to be addressed with extensive research and testing efforts on different VNF deployment learning models for various practical IoT driven scenarios. Another direction of future work can be using LSTM so that once new data points arrive, the pre-trained model can be updated based on the new arrival data points. Certainly, deep learning can emerge as a promising approach to solve such combinatorial optimization problems, when integrated with the research expertise of such areas.

Appendix A

List of Abbreviations

5G: The fifth generation
6G: The sixth generation

A

API: Application programming interface
AR: Augmented reality
AI: Artificial intelligence
ANN: Artificial neural network
ACO: Ant colony optimization

B

B5G: Beyond 5G
BITW: Bump-in-the-wire
BP: Backpropagation

C

CNN: Convolutional neural network
CPU: Central processing unit

D

DC: Data center
DL: Deep learning
DQN: Deep Q-network
DQL: Deep Q-learning
DSO: Deep Q-network sharing based orchestration

E

E2E: End-to-end
ETSI: European Telecommunications Standards Institute
E-ConvNets: Ensemble convolutional neural network
E-ANN: Ensemble artificial neural network
eMBB: Enhanced mobile broadband

F

FF: First fit
FL: Federated learning

G

GPU: Graphics processing unit
Gboard: Google keyboard

I

ILP: Integer linear programming
IoT: Internet of things

K

KPI: Key performance indicator

L

LP: Linear programming
LAN: Local area network

M

MEC: Multi-access edge computing
ML: Machine learning
MDP: Markov decision process
MSE: Mean squared error
mMTC: Mobile machine type communication
MEL: Mobile edge learning
MILP: Mixed integer linear programming

N

NFV: Network Function Virtualization
NREN: National research and education network
NB-IoT: Narrow band IoT

O

OVN: Open virtual networking

P

PDN-GW: Packet data network gateway

Q

QoE: Quality of experience
QoS: Quality of service

R

ReLU: Rectified linear units
RAN: Radio access network
RAM: Random access memory

S

SI: Swarm intelligence
SDN: Software defined networking
SFC: Service function chaining
SLA: Service level agreement
S-GW: Serving gateway

T

TSP: Telecommunications service provider
t-ACO: Tuned ant colony optimization

U

UAV: Unmanned aerial vehicle
UUV: Unmanned underwater vehicle
URLLC: Ultra reliable low-latency communication

V

VR: Virtual reality
VM: Virtual machine
VNF: Virtual network function
VAS: Value added services
VNE: Virtual network embedding

W

WAN: Wide area network

Bibliography

[1] J. Navarro-Ortiz, P. Romero-Diaz, S. Sendra, P. Ameigeiras, J. J. Ramos-Munoz, and J. M. Lopez-Soler, “A survey on 5g usage scenarios and traffic models,” *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 905–929, 2020.

[2] N. N. Dao, Q. V. Pham, N. H. Tu, T. T. Thanh, V. N. Q. Bao, D. S. Lakew, and S. Cho, “Survey on aerial radio access networks: Toward a comprehensive 6g access infrastructure,” *IEEE Communications Surveys Tutorials*, pp. 1–1, 2021.

[3] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, “The roadmap to 6g: Ai empowered wireless networks,” *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[4] A. M. Zarca, J. B. Bernabe, A. Skarmeta, and J. M. Alcaraz Calero, “Virtual iot honeynets to mitigate cyberattacks in sdn/nfv-enabled iot networks,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1262–1277, 2020.

[5] W. Dong, Z. Xu, X. Li, and S. Xiao, “Low-cost subarrayed sensor array design strategy for iot and future 6g applications,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4816–4826, 2020.

[6] H. Yang, A. Alphones, Z. Xiong, D. Niyato, J. Zhao, and K. Wu, “Artificial-intelligence-enabled intelligent 6g networks,” *IEEE Network*, vol. 34, no. 6, pp. 272–280, 2020.

[7] B. Mao, Y. Kawamoto, and N. Kato, “Ai-based joint optimization of qos and security for 6g energy harvesting internet of things,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7032–7042, 2020.

[8] G. Liu, Y. Huang, N. Li, J. Dong, J. Jin, Q. Wang, and N. Li, “Vision, requirements and network architecture of 6g mobile network beyond 2030,” *China Communications*, vol. 17, no. 9, pp. 92–104, 2020.

[9] M. Emu and S. Choudhury, “Iot ecosystem on exploiting dynamic vnf orchestration and service chaining: Ai to the rescue?” *IEEE Internet of Things Magazine*, vol. 3, no. 4, pp. 30–35, 2020.

[10] A. Muhammad, L. Qu, and C. Assi, “Delay-aware multi-source multicast resource optimization in nfv-enabled network,” in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[11] M. Emu, P. Yan, and S. Choudhury, “Latency aware vnf deployment at edge devices for iot services: An artificial neural network based approach,” in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020, pp. 1–6.

[12] M. Emu and S. Choudhury, “Ensemble deep learning aided vnf deployment for iot services,” in *2020 16th International Conference on Network and Service Management (CNSM)*, Izmir, Turkey, 2020, pp. 1–7.

[13] B. Jaeger, “Security orchestrator: Introducing a security orchestrator in the context of the etsi nfv reference architecture,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 1255–1260.

[14] R. Amin, M. Reisslein, and N. Shah, “Hybrid sdn networks: A survey of existing approaches,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[15] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, “Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey,” *IEEE Access*, vol. 7, pp. 166 079–166 108, 2019.

[16] S. Agarwal, F. Malandrino, C. C. Fabiana, and S. De, “Vnf placement and resource allocation for the support of vertical services in 5g networks,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, p. 433–446, Feb. 2019.

[17] L. Fallon, J. Keeney, and R. K. Verma, "Autonomic closed control loops for management, an idea whose time has come?" in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1–5.

[18] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. pp. 2595–2621, Fourthquarter 2018.

[19] A. Hirwe, I. Dalal, and K. Kataoka, "Predicting dynamic network state of sfc operation with uncertainties," *IEEE Communications Letters*, vol. 24, no. 11, pp. 2564–2568, 2020.

[20] W. Ren, Y. Sun, H. Luo, and M. S. Obaidat, "A new scheme for iot service function chains orchestration in sdn-iot network systems," *IEEE Systems Journal*, vol. 13, no. 4, pp. 4081–4092, 2019.

[21] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network: integrating software defined networking and network function virtualization," *IEEE Network*, vol. 29, no. 3, pp. 36–41, May 2015.

[22] X. Cheng, Y. Wu, G. Min, and A. Y. Zomaya, "Network function virtualization in dynamic networks: A stochastic perspective," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2218–2232, Oct 2018.

[23] L. Ma, X. Wen, L. Wang, Z. Lu, and R. Knopp, "An sdn/nfv based framework for management and deployment of service based 5g core network," *China Communications*, vol. 15, no. 10, pp. 86–98, 2018.

[24] T. Subramanya and R. Riggio, "Machine learning-driven scaling and placement of virtual network functions at the network edges," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, June 2019, pp. 414–422.

[25] Y. Liao, L. Shou, Q. Yu, Q. Ai, and Q. Liu, "An intelligent computation demand response framework for iiot-mec interactive networks," *IEEE Networking Letters*, vol. 2, no. 3, pp. 154–158, 2020.

[26] H. Tang, D. Zhou, and D. Chen, "Dynamic network function instance scaling based on traffic forecasting and vnf placement in operator data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 530–543, March 2019.

[27] B. Farkiani, B. Bakhshi, and S. A. Mirhassani, "A fast near-optimal approach for energy-aware sfc deployment," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1360–1373, Dec 2019.

[28] K. Mebarkia and Z. Zsóka, "Service traffic engineering: Avoiding link overloads in service chains," *Journal of Communications and Networks*, vol. 21, no. 1, pp. 69–80, 2019.

[29] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah, "Generic ilp versus specialized 0-1 ilp: an update," in *IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002.*, San Jose, CA, USA, 2002, pp. 450–457.

[30] J. A. Momoh, M. E. El-Hawary, and R. Adapa, "A review of selected optimal power flow literature to 1993. ii. newton, linear programming and interior point methods," *IEEE Transactions on Power Systems*, vol. 14, no. 1, pp. 105–111, 1999.

[31] R. lima and E. Seminar, "Ibm ilog cplex-what is inside of the box," in *Proc. 2010 EWO Seminar*, 2010, pp. 1–72.

[32] A. S. Minkoff, "A systematic approach to osl application programming," *IBM systems journal*, vol. 31, no. 1, pp. 49–61, 1992.

[33] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

[34] R. Anand, D. Aggarwal, and V. Kumar, "A comparative analysis of optimization solvers," *Journal of Statistics and Management Systems*, vol. 20, no. 4, pp. 623–635, 2017.

[35] X. S. Yang, *Nature-inspired metaheuristic algorithms.* Luniver press, 2010.

[36] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[37] G. Beni, "Swarm intelligence," *Complex Social and Behavioral Systems: Game Theory and Agent-Based Models*, pp. 791–818, 2020.

[38] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[39] P. Roy, A. Tahsin, S. Sarker, T. Adhikary, M. A. Razzaque, and M. Hassan, “User mobility and quality-of-experience aware placement of virtual network functions in 5g,” *Elsevier Computer Communications*, vol. 150, pp. 367–377, 12 2019.

[40] C. Zhang, P. Patras, and H. Haddadi, “Deep learning in mobile and wireless networking: A survey,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, thirdquarter 2019.

[41] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, “Artificial neural networks-based machine learning for wireless networks: A tutorial,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. pp.3039–3071, Fourthquarter 2019.

[42] K. Braiki and H. Yousef, “Resource management in cloud data centers: A survey,” in *IEEE 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, Tangier, Morocco, June 2019, pp. 1007–1012.

[43] X. Huang, S. H. Hong, M. Yu, Y. Ding, and J. Jiang, “Demand response management for industrial facilities: A deep reinforcement learning approach,” *IEEE Access*, vol. 7, pp. 82 194–82 205, 2019.

[44] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, “Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520–5530, June 2019.

[45] Y. You, Z. Zhang, C. Hsieh, J. Demmel, and K. Keutzer, “Fast deep neural network training on distributed systems and cloud tpus,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2449–2462, Nov 2019.

[46] X. Fu, F. R. Yu, J. Wang, Q. Qi, and J. Liao, “Service function chain embedding for nfv-enabled iot based on deep reinforcement learning,” *IEEE Communications Magazine*, vol. 57, no. 11, pp. 102–108, November 2019.

[47] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, “Vnf placement and resource allocation for the support of vertical services in 5g networks,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 433–446, Feb 2019.

[48] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *arXiv preprint arXiv:1811.00866*, 2018.

[49] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[50] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, 1992.

[51] B. Liu, X. Yu, P. Zhang, A. Yu, Q. Fu, and X. Wei, "Supervised deep feature extraction for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 1909–1921, 2018.

[52] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, Fourthquarter 2017.

[53] S. Sakib, M. M. Fouda, Z. M. Fadlullah, N. Nasser, and W. Alasmary, "A proof-of-concept of ultra-edge smart iot sensor: A continuous and lightweight arrhythmia monitoring approach," *IEEE Access*, vol. 9, pp. 26 093–26 106, 2021.

[54] N. Kato, Z. M. Fadlullah, F. Tang, B. Mao, S. Tani, A. Okamura, and J. Liu, "Optimizing space-air-ground integrated networks by artificial intelligence," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 140–147, August 2019.

[55] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[56] T. Vallée, K. Sedki, S. Despres, M. . Jaulant, K. Tabia, and A. Ugon, "On personalization in iot," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, 2016, pp. 186–191.

[57] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning based on over-the-air computation," in *IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[58] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot-privacy: To be private or not to be private," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2014, pp. 123–124.

[59] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 172–185, 2020.

[60] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, "Energy efficient algorithm for vnf placement and chaining," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017, pp. 579–588.

[61] C. Tseng and F. J. Lin, "Extending scalability of iot/m2m platforms with fog computing," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, 2018, pp. 825–830.

[62] G. Zhong, K. Zhang, H. Wei, Y. Zheng, and J. Dong, "Marginal deep architecture: Stacking feature learning modules to build deep learning models," *IEEE Access*, vol. 7, pp. 30 220–30 233, 2019.

[63] T. H. Luan, L. X. Cai, and X. Shen, "Impact of network dynamics on user's video quality: Analytical framework and qos provision," *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 64–78, 2010.

[64] S. Sailik, C. Ankur, S. Abdulhakim, H. Dijiang, A. Adel, and K. Subbarao, "A survey of moving target defenses for network security," *CoRR*, vol. abs/1905.00964, 2019.

[65] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-computing-enabled cognitive network function virtualization for an information-centric future internet," *IEEE Communications Magazine*, vol. 57, no. 7, pp. pp. 48–54, July 2019.

[66] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2961–2991, Fourthquarter 2018.

[67] R. Cziva and D. P. Pezaros, "Container network functions: Bringing nfv to the network edge," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 24–31, June 2017.

[68] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, "Wifi can be the weakest link of round trip network latency in the wild," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, April 2016, pp. 1–9.

[69] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 693–701, 2018.

[70] M. Abu-Lebdeh, D. Naboulsi, R. Glitho, and C. W. Tchouati, "On the placement of vnf managers in large-scale and distributed nfv systems," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 875–889, Dec 2017.

[71] W. Lu, L. Liang, and Z. Zhu, "On vnf-sc deployment and task scheduling for bulk-data transfers in inter-dc eons," in *2017 IEEE/CIC International Conference on Communications in (ICCC)*, Qingdao, China, Oct 2017, pp. 1–4.

[72] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio, "Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 98–112, March 2019.

[73] H. Nguyen, T. Do, A. Hegyi, and C. Rotter, "An approach to apply reinforcement learning for a vnf scaling problem," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 02 2019, pp. 94–99.

[74] X. Zhu and Y. Wang, "Research on virtual network function (vnf) migration," *Nanjing Youdian Daxue Xuebao (Ziran Kexue Ban)/Journal of Nanjing University of Posts and Telecommunications (Natural Science)*, vol. 38, pp. 45–53, 02 2018.

[75] Z. A. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, Y. Vyas, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, Aug. 2013.

[76] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 19–24.

[77] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

[78] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[79] S. Ayoubi, S. Sebbah, and C. Assi, "A cut-and-solve based approach for the vnf assignment problem," *IEEE Transactions on Cloud Computing*, pp. 1–1, 06 2017.

[80] R. Cziva and D. P. Pezaros, "On the latency benefits of edge nfv," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, May 2017, pp. 105–106.

[81] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware vnf placement and chaining based on a flexible resource allocation approach," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov 2017, pp. 1–7.

[82] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive vnf scaling and flow routing with proactive demand prediction," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 486–494.

[83] K. S. Ghai, S. Choudhury, and A. Yassine, "A stable matching based algorithm to minimize the end-to-end latency of edge nfv," *Procedia Computer Science*, vol. 151, pp. pp. 377 – 384, 2019.

[84] M. R. Garey. and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co., 1990.

[85] R. Livni, S. Shalev, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in Neural Information Processing Systems 27.* Curran Associates, Inc., 2014, pp. 855–863.

[86] K. S. Ghai, S. Choudhury, and A. Yassine, "Efficient algorithms to minimize the end-to-end latency of edge network function virtualization," *Journal of Ambient Intelligence and Humanized Computing*, 01 2020.

[87] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[88] M. Gharbaoui, C. Contoli, G. Davoli, G. Cuffaro, B. Martini, F. Paganelli, W. Cerroni, P. Cappanera, and P. Castoldi, "Demonstration of latency-aware and self-adaptive service chaining in 5g/sdn/nfv infrastructures," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Verona, Italy, Nov 2018, pp. 1–2.

[89] H. Ben-Ammar, Y. Hadjadj-Aoul, and S. Ait-Chellouche, "Efficiently allocating distributed caching resources in future smart networks," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2019, pp. 1–4.

[90] H. Ge, F. Jiang, and Z. Zhang, "A hybrid localization algorithm of rss and toa based on an ensembled neural network," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, May 2019, pp. 1280–1284.

[91] J. Fu and G. Li, "An efficient vnf deployment scheme for cloud networks," in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, Chongqing, China, June 2019, pp. 497–502.

[92] V. Quintuna Rodriguez and F. Guillemin, "Cloud-ran modeling based on parallel processing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. pp. 457–468, March 2018.

[93] T. Hirayama, T. Miyazawa, M. Jibiki, and V. P. Kafle, “Service function migration scheduling based on encoder-decoder recurrent neural network,” in *2019 IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, June 2019, pp. 193–197.

[94] C. Zhang, H. P. Joshi, G. F. Riley, and S. A. Wright, “Towards a virtual network function research agenda: A systematic literature review of vnf design considerations,” *Journal of Network and Computer Applications*, vol. 146, p. 102417, 2019.

[95] J. Gil Herrera and J. F. Botero, “Resource allocation in nfv: A comprehensive survey,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[96] J. Fu and G. Li, “An efficient vnf deployment scheme for cloud networks,” pp. pp. 497–502, June 2019.

[97] X. Chen, Z. Zhu, J. Guo, S. Kang, R. Proietti, A. Castro, and S. J. B. Yoo, “Leveraging mixed-strategy gaming to realize incentive-driven vnf service chain provisioning in broker-based elastic optical inter-datacenter networks,” *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 2, pp. pp. 232–240, Feb 2018.

[98] N. T. Khai, A. Baumgartner, and T. Bauschert, “Optimising virtual network functions migrations: A flexible multi-step approach,” in *2019 IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, June 2019, pp. 188–192.

[99] X. Zhou, B. Yi, X. Wang, and M. Huang, “Approach for minimising network effect of vnf migration,” *IET Communications*, vol. 12, no. 20, pp. pp. 2574–2581, 2018.

[100] K. S. Ghai, S. Choudhury, and A. Yassine, “Efficient algorithms to minimize the end-to-end latency of edge network function virtualization,” *Journal of Ambient Intelligence and Humanized Computing*, 01 2020.

[101] Y. Qian, L. Hu, J. Chen, X. Guan, M. Hassan, and A. Alelaiwi, “Privacy-aware service placement for mobile edge computing via federated learning,” *Information Sciences*, vol. 505, pp. 562 – 570, 2019.

[102] H. Ghafoor and Y. Noh, "An overview of next-generation underwater target detection and tracking: An integrated underwater architecture," *IEEE Access*, vol. 7, pp. pp. 98 841–98 853, 2019.

[103] V. Sanchez-Aguero, F. Valera, B. Nogales, L. F. Gonzalez, and I. Vidal, "Venue: Virtualized environment for multi-uav network emulation," *IEEE Access*, vol. 7, pp. 154 659–154 671, 2019.

[104] M. Alasaly, M. Hassan, and A. Alsanad, "A cognitive/intelligent resource provisioning for cloud computing services: opportunities and challenges," *Soft Computing*, p. pp. 9069–9081, 05 2019.

[105] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *Australasian joint conference on artificial intelligence.* Springer, 2008, pp. 372–378.

[106] P. Sequeira and M. Gervasio, "Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations," *Artificial Intelligence*, vol. 288, p. 103367, 2020.

[107] M. Amirabadi, M. Kahaei, and S. Nezamalhosseini, "Novel suboptimal approaches for hyperparameter tuning of deep neural network [under the shelf of optical communication]," *Physical Communication*, vol. 41, p. 101057, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874490719306652

[108] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[109] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[110] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1409–1434, 2019.

[111] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 487–507, 2019.

[112] A. Mohamad and H. S. Hassanein, "On demonstrating the gain of sfc placement with vnf sharing at the edge," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA, 2019, pp. 1–6.

[113] J. Wang, H. Qi, K. Li, and X. Zhou, "Prsfc-iot: A performance and resource aware orchestration system of service function chaining for internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1400–1410, 2018.

[114] G. Wang, S. Zhou, S. Zhang, Z. Niu, and X. Shen, "Sfc-based service provisioning for reconfigurable space-air-ground integrated networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1478–1489, 2020.