IEEE ComSoc IEEE Communications Society · IEEE COMPUTER SOCIETY · IEEE Signal Processing Society · VTS · IEEE Transactions on Machine Learning in Communications and Networking

Received 19 May 2024; revised 17 September 2024; accepted 16 October 2024.
Date of publication 24 October 2024; date of current version 1 November 2024.

The associate editor coordinating the review of this article and approving it for publication was Y. Zhang.

*Digital Object Identifier 10.1109/TMLCN.2024.3485520*

# Information Bottleneck-Based Domain Adaptation for Hybrid Deep Learning in Scalable Network Slicing

**TIANLUN HU[1,2] (Student Member, IEEE), QI LIAO[1] (Member, IEEE), QIANG LIU[3] (Member, IEEE), AND GEORG CARLE[2]**

[1]Nokia Bell Labs, 70469 Stuttgart, Germany
[2]School of Computation, Information and Technology, Technical University of Munich, 80333 Munich, Germany
[3]School of Computing, University of Nebraska–Lincoln, Lincoln, NE 68588 USA

CORRESPONDING AUTHOR: T. HU (tianlun.hu@tum.de)

This work was supported by German Federal Ministry of Education and Research (BMBF) under Project 6G-ANNA. The work of Qiang Liu was supported by the US National Science Foundation under Grant 2333164.

**ABSTRACT** Network slicing enables operators to efficiently support diverse applications on a shared infrastructure. However, the evolving complexity of networks, compounded by inter-cell interference, necessitates agile and adaptable resource management. While deep learning offers solutions for coping with complexity, its adaptability to dynamic configurations remains limited. In this paper, we propose a novel hybrid deep learning algorithm called IDLA (integrated deep learning with the Lagrangian method). This integrated approach aims to enhance the scalability, flexibility, and robustness of slicing resource allocation solutions by harnessing the high approximation capability of deep learning and the strong generalization of classical non-linear optimization methods. Then, we introduce a variational information bottleneck (VIB)-assisted domain adaptation (DA) approach to enhance integrated deep learning and Lagrangian method (IDLA)'s adaptability across diverse network environments and conditions. We propose pre-training a variational information bottleneck (VIB)-based Quality of Service (QoS) estimator, using slice-specific inputs shared across all source domain slices. Each target domain slice can deploy this estimator to predict its QoS and optimize slice resource allocation using the IDLA algorithm. This VIB-based estimator is continuously fine-tuned with a mixture of samples from both the source and target domains until convergence. Evaluating on a multi-cell network with time-varying slice configurations, the VIB-enhanced IDLA algorithm outperforms baselines such as heuristic and deep reinforcement learning-based solutions, achieving twice the convergence speed and $\mathbf{16.52\%}$ higher asymptotic performance after slicing configuration changes. Transferability assessment demonstrates a $\mathbf{25.66\%}$ improvement in estimation accuracy with VIB, especially in scenarios with significant domain gaps, highlighting its robustness and effectiveness across diverse domains.

**INDEX TERMS** Domain adaptation, deep learning, non-linear optimization, network slicing, resource allocation.

## I. INTRODUCTION

NETWORK slicing has been widely investigated in 5G and beyond to support diverse services with cost-efficiency, flexibility, and performance assurance [2]. It enables the creation of virtualized slices tailored to specific service needs, optimizing resource utilization and enhancing overall service quality. However, the ever-increasing complexity and variability in network dynamics, highlighted by frequent changes in slicing configurations, significantly complicate the task of resource management [3]. In static network environments, the problem of resource allocation has been widely explored, focusing on optimizing slice resource allocation for specific slice configurations. Prior studies [4], [5] have tackled the challenges of highly intertwined and constrained slice resource allocation by formulating analytical closed-form network models and employing constrained non-linear optimization methods. Research findings in [6] and [7] indicate that these approximated models cannot accurately represent the complex demands and performances of slices and advocate the use of reinforcement learning (RL) methods for resource management under dynamic network optimization. Recent studies [8], [9] utilizing multi-agent RL frameworks further simplify the complexity with low-dimensional state and action spaces. Despite these advantages, deep reinforcement learning (DRL)-based approaches face significant challenges in reproducibility and generalization across diverse network scenarios. These models are typically highly specialized to the environments in which they are trained, often requiring extensive retraining and fine-tuning for new conditions.

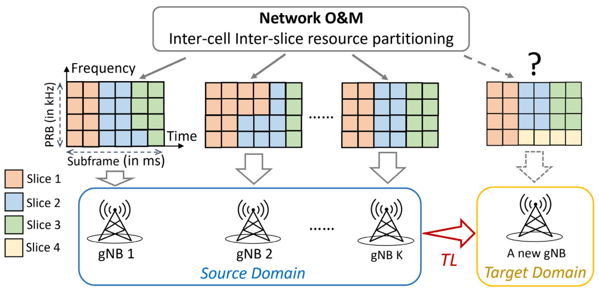

**FIGURE 1. Transfer learning (TL) of network slice resource allocation. The figure illustrates the process of dynamic slicing resource allocation in network operation and maintenance (O&M), and TL is applied to address challenges in new deployments.**

To address poor model reproducibility and limited sample efficiency, recent advancements in transfer learning (TL) [10] rapidly adapt pre-solved solutions to new tasks, reducing both processing time and data demand. By leveraging prior knowledge from relevant tasks, TL improves training efficiency, inspiring its use in wireless communication for network optimization [11], [12], [13]. Several studies [14], [15], [16] have shown TL's potential to reduce resource consumption and improve efficiency. However, TL methods struggle with challenges like data imbalance, domain shift, and negative knowledge transfer, limiting their generalization capabilities. Those drawbacks highlight the importance of studying domain adaptation (DA) [17], which mitigates domain disparities and enhances transferability. In wireless communication, DA boosts robustness and generalization [18], [19], [20], making it critical for dynamic scenarios like slice resource allocation. Additionally, the information bottleneck (IB) method [21] offers a promising path for generating flexible and generalizable solutions across varying network environments by maximizing the transfer of relevant information between source and target domains. The authors in [22] further discussed IB implications in machine learning models with practical applications in classification and generative modeling tasks, which paves an inspirable path in exploring IB-based approaches in the wireless communication field for achieving flexible and generalizable solutions from one network environment (source domain) to another (target domain), even when characteristics differ.

Fig. 1 illustrates the process of slice resource allocation in network operation and maintenance (O&M), dynamically allocating per-slice resource budgets as ratios of network resource in each cell at medium time scale, i.e., minutes or a quarter. Subsequently, the radio access network (RAN) resource scheduler periodically allocates physical resource blocks (PRBs) to each service based on the resource portions assigned by O&M as per-slice resource upper bounds. As Fig. 1 indicates, in fifth generation (5G) and beyond networks, deploying new base stations with dynamic slice configurations and sparse samples causes challenges for slice resource schedulers using existing distributing strategies. To address this issue, this work proposes a TL-aided deep learning approach with DA to optimize slice resource allocation for O&M, ensuring adaptability to diverse slice configurations and network conditions with high flexibility and generality. Our objective is to optimize slice-wise resource allocation strategies for both existing (source domain) and newly deployed (target domain) network slices, accommodating dynamic configurations regarding the number of slices and their respective quality of service (QoS) requirements. We aim to derive common solutions based on limited domain samples, ensuring efficient resource allocation across varying slice conditions.

In this paper, we first introduce the IDLA algorithm, which integrates deep learning with the Lagrangian method for highly scalable slice-wise resource allocation. This approach derives a general slice-wise network performance estimator shared across all slices using a neural network regression model and subsequently addresses the resource allocation problem for each cell as a constrained non-linear optimization problem. Then, based on the IDLA framework, we further address domain gaps across network environments by improving the slice-wise estimator's generality through TL techniques. Specifically, we implemented a VIB-based regression model [23], which consists of an encoder for invariant feature extraction and a network performance estimator, to replace the multi-layer perceptron (MLP)-based regression. This VIB model adapts to a broader range of network scenarios with its high representative ability [24], allowing the derivation of slice resource allocation solutions following the IDLA approach. Compared to previous TL methods, our approach integrates DA with a VIB-based slice QoS model, significantly improving transferability and accuracy within the IDLA framework. Unlike traditional TL approaches that only apply to static slicing configurations, our approach can flexibly adapt to dynamic network slicing environments with varying number of slices and their respective requirements, enhancing generalization and scalability. To evaluate the VIB model's performance in mitigating domain discrepancies, we compared its estimation accuracy in source and target domains across various sample combinations and network setups against two baseline methods: one employing domain sample re-weighting [25], a widely adopted technique for addressing sample imbalance, and the other using a conventional MLP regression model. The numerical results indicate that the VIB approach significantly enhances target domain accuracy, even with fewer samples, while maintaining robust performance in source domains, particularly when incorporating more target samples during training. In DA scenario comparisons, the VIB method outperforms both MLP and loss-reweighting baseline methods [26]. We also implement our IDLA method with proposed VIB-based estimators in a system-level network simulator to evaluate their practical efficiency in diverse slicing scenarios. In particular, we demonstrate a dynamic network slicing scenario, which deploys varying slicing configurations in terms of slice types and combinations in real-time, illustrating the high flexibility and scalability of the TL-aided IDLA algorithm compared with a DRL slicing approach [9] and IDLA without TL. The main contributions of this paper are summarized as follows:

- *Improving scalability and robustness with integrated deep learning and Lagrangian method*: We introduce IDLA, a novel method for slicing resource allocation that incorporates deep learning models, known for their strong approximation capabilities with constrained non-linear optimization method for robustness to data distribution and diverse utility models. We derive a general deep neural network (DNN) model to approximate network utility across slices and use this model to address the slicing resource allocation problem as a constrained non-linear optimization task with the Lagrangian method. By leveraging the computational efficiency of deep learning's partial derivatives, which facilitates fast computation of gradients in Lagrangian methods, we offer scalable and rapidly deployable solutions for resource allocation in real-world scenarios.
- *Enhancing generalizability with information bottleneck-based domain adaptation*: Given the challenges of imbalanced data and domain shift, we formulated the generalization of the slice QoS estimator as a domain adaptation problem in the context of transfer learning. To broaden the generality of the slice performance estimator, we develop a VIB-based DNN architecture incorporating the domain adaptation features. The model consists of an encoder extracting common patterns across domains and an estimator handling domain conditions.
- *Reducing complexity and improving flexibility with shared slice-wise models*: Unlike other works, which include features of all slices jointly, resulting in high-dimensional state and action spaces [27], our approach follows a multi-agent centralized training and distributed execution (CTDE) concept [28] and derives a lightweight general slice-wise DNN model, to be shared across diverse slices and domains. We train a single shared slice-wise model using samples from all slices, allowing it to generalize across various slice conditions. After training, the model can be deployed for inference across a large batch of distinct slices. In addition, we handle the inter-slice constraints with classical non-linear optimization methods. This approach significantly reduces the model complexity, improves sample efficiency, and enhances adaptability to dynamic slice configurations.
- *Numerical studies and advantages*: We evaluated the domain adaptation ability of the proposed VIB model under various cross-domain scenarios. Through experimentation across diverse training data combinations from various domains, our model exhibits twice convergence speed and 16.52% higher asymptotic compared to the conventional DNN model and a sample reweighting technique tackling data imbalance issues. Concerning domain shifts, the VIB-based model demonstrates a 25.66% improvement in estimation accuracy, especially in scenarios with significant domain gaps, highlighting its robustness and effectiveness across diverse domains.

The rest of the paper is organized as follows. In Section III, we first define the network slicing model, then formulate the slice resource allocation and DA problems. We introduce the IDLA algorithm for scalable slice-wise resource allocation by integrating deep learning with non-linear optimization in Section IV. In Section V, we further introduce VIB-aided IDLA as DA process for higher solution generality. We demonstrated the numerical results in Section VI, and we conclude this paper in Section VII.

## II. RELATED WORKS

This work relates to slice resource management in networking, transfer learning methods, and information bottleneck theory.

### A. NETWORK SLICE RESOURCE MANAGEMENT

Numerous studies have focused on addressing a critical challenge in network slicing: the allocation of RAN slice resources. Traditional methods, like in [29], [30], assume static slice demands to allocate resource budgets, utilizing dynamic programming and stochastic optimization. Anousheh et al. [31] adopted a two-time-scale approach, considering slice demand distributions on different scales and minimizing costs through stochastic mixed-integer programming. Others, such as [32] and [33], tackled inter-cell interference by formulating resource allocation as constrained optimization problems in dense networks, acknowledging inter-cell dependencies.

Recent advancements in DRL have led to innovative slice resource allocation solutions. Guo et al. [34] developed a DRL approach for admission control and resource allocation in network slicing. Similarly, Abiko et al. [35] proposed a DRL strategy for flexible PRB allocation in RAN slicing, enhancing profits and efficiency. In single-cell environments, [36] applied DRL to align resource allocation with user patterns, while Liu et al. [37] decomposed RAN slicing optimization into a primary problem with dedicated DRL agents for each slice. Yu and Gu [38] examined multi-cell scenarios and implemented DRL algorithms for complex multi-cell management. Notably, in [39], the authors introduced a constrained RL-based resource allocation framework for network slicing by optimizing midterm minimal network QoS or user quality of experience (QoE), demonstrating promising performance in satisfying service demand guarantees. However, it only addressed slicing scenarios in single-cell networks, neglecting the effects of inter-cell dependencies. Moreover, these prior studies mainly focused on optimizing slicing performance, with limited attention to the solutions' generalization capabilities across diverse network conditions.

### B. TRANSFER LEARNING AND DOMAIN ADAPTATION APPLICATIONS

The method of TL has been well explored in various fields, while its application on wireless communication remains relatively scarce but is gradually gaining attention. Yang et al. [40] applied TL to adapt a pre-trained beamforming model in a massive MIMO system, mitigating hardware limitations and requiring fewer channel data for derivation. TL was also recommended in [41] for radio map estimation, where the authors fine-tuned a source model trained on a specific network for other environments, highlighting TL's efficacy with limited training data. Tailored for DRL, Janiar et al. proposed a TL approach [42] to accelerate DRL training in wireless networks with an integrated feature extractor, which quantifies the disparity between source and target domains. Experimental results demonstrated significant reductions in training time, outperforming conventional DRL approaches.

Among TL techniques, DA has attracted significant attention in practical applications [43], [44], [45]. In [46], the authors explored the use of TL and DA in the context of Artificial Intelligence (AI)-native telecommunication networks, discussing their application to specific challenges, such as service performance prediction, end-to-end latency estimation, and block call rate prediction. The study highlighted the significant potential of DA techniques in improving performance within the communication field. Guan et al. [47] introduced an uncertainty-aware domain adaptation network (UaDAN) for object detection, applying conditional adversarial learning to align samples with varying degrees of similarity and leveraging uncertainty metrics for adaptive learning. Chen et al. [48] introduced CrossTrainier, a system for DA that leverages loss reweighting to improve model reproducibility across different sample sources, yet the sensitivity of loss reweighting hyperparameter requires expensive tuning and retraining procedures. In End-to-End communication, Raghuram et al. [49] advocated for DA via autoencoders to mitigate the need for frequent retraining amidst changing channel conditions, proposing a method for adapting a Gaussian mixture density network with minimal target distribution samples and validating its effectiveness through simulation. In the realm of wireless communication, Zhou et al. [20] presented SemiAMR, a semi-supervised network utilizing adversarial learning for cross-domain modulation recognition without the need for pre-training on labeled target domain data, improving classification accuracy. Shi et al. [50] tackled the simulation-to-reality gap in network configuration prediction through a teacher-student DNN approach for DA, merging simulation and real-world data to enhance training outcomes.

### C. TRANSFER LEARNING WITH INFORMATION BOTTLENECK

In the context of TL, one significant application of IB in RL is introduced by Goyal et al. [51], where IB was applied to enhance exploration by learning a default policy across multiple goals. This approach showcases how IB can improve knowledge transfer to new environments, offering advantages over traditional task-specific RL methods. In the field of medical imaging, Chen et al. [52] applied IB theory to a generative adversarial network (GAN) for cross-domain image translation by preserving relevant features and discarding extraneous information.

Building on the foundations of IB, the VIB emerged as a powerful tool for extracting transferable knowledge across domains. In [53], Fu et al. introduced Pluvio, a search engine that applies the VIB to handle out-of-domain architectures and libraries. By leveraging pre-trained language models with VIB, Pluvio demonstrates improved resilience to unseen architectures, outperforming state-of-the-art search engines in robustness and accuracy. Similarly, Song et al. [54] proposed a VIB-aided domain adaptation method to improve feature transferability by focusing on task-relevant information while filtering out irrelevant factors. The approach achieves balanced representations across source and target domains by combining VIB with conditional entropy minimization. These experiments show that VIB significantly reduces generalization error, addressing the limitations of traditional domain adaptation methods that only match marginal distributions. Recent works extended the benefits of VIB to TL. In [55], the authors propose using VIB to suppress irrelevant features in pre-trained sentence representations, improving the ability of task-specific classifiers to generalize from small datasets. The proposed model maps sentence embeddings to a latent space, outperforming conventional fine-tuning and other regularization techniques on low-resource datasets. However, no existing works have integrated the VIB approach into the cross-domain network slicing problem. This gap presents an opportunity to improve the robustness of feature extraction and decision-making, enabling more resilient performance across diverse network domains.

## III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, first, we define the dynamic network slicing system model in Section III-A. Then, in Section III-B, we formulate the scalable slice-wise resource allocation as a nonlinear optimization problem. The DA problem formulation under the context of TL is given later in Section V-A. Table 1 lists the notations used in this paper.

**TABLE 1.** Table of notations.

| Symbol | Meaning |
|---|---|
| $s$ | Network slice index $s \in \mathcal{S}$ |
| $c$ | Network cell index $c \in \mathcal{C}$ |
| $\phi_{c,s}$ | Average user throughput of slice $s$ and cell $c$ |
| $d_{c,s}$ | Average user delay of slice $s$ and cell $c$ |
| $x_{c,s}$ | Per-slice resource portion of slice $s$ and cell $c$ |
| $\mathbf{x}_c$ | Per-cell portions of all slices in cell $c$ |
| $\mathbf{x}$ | Global slice resource portions over all cells |
| $y_{c,s}$ | QoS metric of slice $s$ and cell $c$ |
| $\mathbf{o}_{c,s}$ | Network observation of slice $s$ and cell $c$ |
| $v_{c,s}$ | Number of active users of slice $s$ and cell $c$ |
| $q_{c,s}$ | Average user CQI of slice $s$ and cell $c$ |
| $\mathcal{X}$ | Input space |
| $\mathcal{Y}$ | Output space |
| $P_{c,s}(X,Y)$ | Joint sample distribution of slice $s$ and cell $c$ |
| $\mathcal{D}_{c,s}$ | Domain comprised by $\mathcal{D}_{c,s} := \{\mathcal{X}, \mathcal{Y}, P_{c,s}(X,Y)\}$ |
| $\mathcal{T}_{c,s}$ | Task $\mathcal{T}_{c,s} := \{\mathcal{Y}, f_{c,s}(\cdot)\}$ with $f_{c,s} : \mathcal{X} \to \mathcal{Y}$. |
| $\mathcal{D}_\mathsf{S}$ | Source domain $\mathcal{D}_\mathsf{S} := \{\mathcal{D}_{c,s} : c \in \mathcal{C}, s \in \mathcal{S}_c\}$ |
| $\mathcal{D}_\mathsf{T}$ | Target domain $\mathcal{D}_\mathsf{T} := \{\mathcal{X}, \mathcal{Y}, P_\mathsf{T}(X,Y)\}$ |
| $I_c$ | Information bottleneck |

## A. SYSTEM MODEL

In this work, we consider multi-cell slicing scenarios within a network system comprising a set of cells $\mathcal{C} := \{1, 2, \dots, C\}$ with dynamically configured slices from a collection denoted as set $\mathcal{S} := \{1, 2, \dots, S\}$. For each cell $c \in \mathcal{C}$, the set of slicing can be time-varying, represented as $\mathcal{S}_c(t) \subset \mathcal{S}$ with cardinality $|\mathcal{S}_c(t)| = S_c(t)$, where $S_c(t)$ is the number of slices in cell $c$ at time slot $t \in \mathbb{N}_0$. At each time slot $t$, the instantaneous slice performance is measured by the network QoS in terms of per-slice user throughput $\phi_{c,s}(t)$ and delay $d_{c,s}(t)$, referring to pre-defined requirements $\phi_s^*$ and $d_s^*$, respectively. It is worth noting that the slice types in this work are defined with network QoS referencing throughput and delay requirements. The problem formulation and proposed solutions in the following sections can be easily generalized to broader requirements.

For each cell $c \in \mathcal{C}$ at time slot $t$, O&M optimizes inter-slice resource portions $\boldsymbol{x}_c(t)$, composed of ratios of resource allocation to all slices, given by

$$\boldsymbol{x}_c(t) := \left[x_{c,1}(t), \dots, x_{c,S_c(t)}(t)\right] \in \mathcal{X}_c(t),\ \forall c \in \mathcal{C}, \quad (1)$$

$$\text{where } \mathcal{X}_c(t) := \left\{ [0,1]^{S_c(t)} \middle| \sum_{s \in \mathcal{S}_c(t)} x_{c,s}(t) \le 1 \right\}. \quad (2)$$

Here, $\boldsymbol{x}_c(t)$ represents the resource portions allocated to slices in cell $c$, with each element $x_{c,1}(t), x_{c,2}(t), \dots, x_{c,S_c(t)}(t)$ corresponding to the resource of a slice, bounded between 0 and 1. $S_c(t)$ denotes the set of slices in cell $c$ at time $t$. The slice resource allocation space $\mathcal{X}_c(t)$ consists of $\boldsymbol{x}_c(t)$, where the sum constraint $\sum_{s \in \mathcal{S}_c(t)} x_{c,s}(t) \le 1$ ensures that the total resource allocation across all slices in cell $c$ at time $t$ does not exceed 1, reflecting finite resource availability.

Then, the overall collection of network slice resource portions is denoted as $\boldsymbol{x}(t) := [\boldsymbol{x}_1(t), \dots, \boldsymbol{x}_C(t)]$. The slice-wise performance of slice $s \in \mathcal{S}_c(t)$ in cell $c \in \mathcal{C}$ is measured by the QoS satisfaction level $y_{c,s}(\boldsymbol{x}(t))$, which is a metric that assesses the effectiveness of the allocated resources determined by the proposed methods, defined as:

$$y_{c,s}(\boldsymbol{x}(t)) := \min\left\{ \frac{\phi_{c,s}(\boldsymbol{x}(t))}{\phi_s^*}, \frac{d_s^*}{d_{c,s}(\boldsymbol{x}(t))}, 1 \right\}, \quad (3)$$

where $\phi_{c,s}(\boldsymbol{x}(t))$ and $d_{c,s}(\boldsymbol{x}(t))$ are the instantaneous slice throughput and delay at time slot $t$, respectively. As (3) illustrates, slice performance is determined as the minimum satisfaction level among slice throughput and delay, which is upper-bounded by 1. In other words, the slice performance $y_{c,s}$ is measured as 1 only if both throughput $\phi_{c,s}$ and delay $d_{c,s}$ meet the requirements; otherwise, it is measured as the minimum between $\phi_{c,s}/\phi_s^*$ and $d_s^*/d_{c,s}$.

*Remark 1:* Note that at each time slot $t$, due to inter-cell dependencies, the slice throughput $\phi_{c,s}(t)$ and delay $d_{c,s}(t)$ not only depend on the local slice resource portion $x_{c,s}(t)$ but also on the allocated resources in other slices of interfered cells. Therefore, in (3), the QoS metric $y_{c,s}(t)$ is defined as a function of the global slice resource allocation $\boldsymbol{x}(t)$.

## B. SLICE RESOURCE ALLOCATION PROBLEM FORMULATION

In this work, our primary objective is to provide efficient and scalable solutions for slice resource allocation to optimize the utility of network QoS across all slices and cells at each time slot $t$. In Problem 1, we formulate the general problem of optimizing global network resource allocation concerning the QoS performance at each time slot.

*Problem 1 Global Slicing Problem:*

$$\begin{aligned} \max_{\boldsymbol{x}(t)}.\quad & U(\boldsymbol{y}(\boldsymbol{x}(t))) \\ \text{subject to } & \boldsymbol{y}(\boldsymbol{x}(t)) := \left[y_{c,s}(\boldsymbol{x}(t)) : c \in \mathcal{C}, s \in \mathcal{S}_c(t)\right], \\ & (1),\ (2),\ (3),\ \forall t. \end{aligned} \quad (4)$$

We define the network utility function as the sum of the logarithms of the local network performance metrics:

$$U(\boldsymbol{y}(\boldsymbol{x}(t))) := \sum_{c \in \mathcal{C}, s \in \mathcal{S}_c(t)} \log\left(y_{c,s}(\boldsymbol{x}(t)) + 1\right). \quad (5)$$

Here, we define the commonly used log utility function [56], which promotes fair resource distribution among slices. This approach encourages more efficient resource utilization while ensuring a balanced performance across various slices. Please note that leveraging the superior approximation capability of deep learning, our proposed approach can be extended to accommodate a wide range of utility functions according to different system designs and requirements.

Solving Problem 1 presents multifaceted difficulties. Firstly, the complexity of the utility function poses challenges for function approximation, mainly due to limited measurements in O&M. In contrast to RAN, where user and channel feedback can be collected with fine time granularity (e.g., in milliseconds), O&M only collects averaged cell and slice-level key performance indicators (KPIs) with a coarse granularity (e.g., in minutes). Consequently, deriving closed-form expressions becomes exceptionally challenging. Secondly, the flexible slice configurations and inter-slice constraints further complicate the problem, resulting in slow convergence and poor adaptability of deep learning-based approaches. Finally, O&M's high scalability demand makes it challenging to use large global models or collaborative multi-agent local models that require extensive exploration to learn from scratch.

Moreover, the challenges of deriving the solution for Problem 1 also raise another concern: since deriving the slice resource allocation solution for one specific network scenario can be time-consuming and complicated, deploying the same approach when the network scenario changes or under a new environment can incur significant costs in both time and resources. Hence, we pose the question: Can we enhance the adaptability of this solution to accommodate various network environments or setups more effectively?

## IV. THE IDLA ALGORITHM

In this section, we introduce IDLA, a novel algorithm that integrates deep learning with a non-linear optimization method to tackle the challenges posed by the complex utility function, slice flexibility, and algorithm scalability. Initially, we outline the design of a DNN model for estimating per-slice network utility, allowing the decomposition measurement of global network performance through distributed performance estimators on a per-slice basis. Then, we address the slice resource allocation problem by formulating it as a non-linear optimization task, leveraging the insights provided by the derived estimation model. Finally, we present an efficient Lagrangian method that capitalizes on the strengths of both the deep learning model and the optimization process.

### A. QoS ESTIMATOR ON PER-SLICE BASIS

For flexible slice-wise solutions, we decompose the global optimization problem defined in Problem 1 into multiple local problems, enabling distributed solutions to derive optimal local slice resource portions independently. However, the challenge of solving function (5) arises from the inter-cell dependency of each local utility $U_{c,s}(y_{c,s}(\boldsymbol{x}(t)))$ on the global slice resource allocation $\boldsymbol{x}(t)$. In this work, we explore the feasibility of approximating each local per-slice QoS satisfaction level $y_{c,s}(t)$ using a general estimator $f_{\boldsymbol{\theta}}(x_{c,s}(t), \boldsymbol{o}_{c,s}(t))$ for all cells and slices, based solely on the local observations $\boldsymbol{o}_{c,s}(t)$ and slice resource portion $x_{c,s}(t)$. This approximation is expressed as

$$f_{\boldsymbol{\theta}}\left(x_{c,s}(t), \boldsymbol{o}_{c,s}(t)\right) \approx y_{c,s}\left(\boldsymbol{x}(t)\right), \quad \forall s \in \mathcal{S}_c(t), \forall c \in \mathcal{C}, \tag{6}$$

where $\boldsymbol{\theta}$ represents the parameters that characterize the estimator.

Conventional QoS estimation models, based on assumed closed-form expressions, struggle with complex utility functions and sparse data from O&M. Yet O&M gathers averaged performance indicators at cell and slice levels in coarser intervals, like minutes, complicating the derivation of precise formulations. Unlike the conventional approach, our proposed estimator model is data-driven, utilizing samples collected from network KPIs. Theoretically, deep learning models can effectively address the mapping between allocated slice resource ratios and slice-wise network performance if the training samples adequately represent network behaviors. We propose the following data collection process to ensure the representativeness of training samples.

#### 1) DATA COLLECTION

The data for training the general network performance estimator are collected periodically in network O&M, typically every 15 minutes, Practically, collecting data too frequently (e.g., every minute) increases the load on both network resources and monitoring systems, while longer intervals (e.g., hourly) may overlook critical performance fluctuations. The 15 minutes interval balances the trade-off between timely issue detection and minimizing overhead, offering sufficient granularity for network KPIs analysis, which also aligns with practical settings and performance measurement guidelines suggested by [57]. Based on prior expert knowledge of network behavior, to evaluate the defined slice utility $y_{c,s}(t)$, computed by achieved throughput $\phi_{c,s}(t)$ and latency $d_{c,s}(t)$, for all $c$ and $s$, we consider the following highly correlated per-slice network KPIs as samples for DNN training:

- **Per-slice resource portion:** $\hat{x}_{c,s}(t)$ is the actual ratio of the PRBs occupied by the slice, resulting in achieved network throughput $\phi_{c,s}(t)$ and delay $d_{c,s}(t)$. It can be seen as the surrogate of the optimizing target $x_{c,s}(t)$, meaning that if resource portion $x_{c,s}(t) := \hat{x}_{c,s}(t)$ were allocated to the slice, the corresponding achieved throughput and delay would be $\phi_{c,s}(t)$ and $d_{c,s}(t)$, respectively. To avoid ambiguity, in the rest of the paper, we only use $x_{c,s}(t)$ to represent the slice resource portion;
- **Number of active users:** This terms composes the previous $H$ steps of historical per-slice average number of active users as $\boldsymbol{v}_{c,s}(t) := [v_{c,s}(t-H), \ldots, v_{c,s}(t-1)]$;
- **Channel quality indicator (CQI):** Similarly, the slice CQI samples are taken with the previous $H$ states of per-slice average CQI as $\boldsymbol{q}_{c,s} := [q_{c,s}(t-H), \ldots, q_{c,s}(t-1)]$;
- **Slice QoS requirements**: The QoS requirements are considered in terms of slice throughput and delay requirements as $\phi^*_{c,s}$ and $d^*_{c,s}$ respectively.

Here, we propose to collect multiple historical states of the number of users and CQI based on the hypothesis that the traffic demands do not change rapidly among successive time steps. We hope that the historical slice states can capture temporal correlation and reflect some hidden information extracted from the missing global states, such as inter-cell and inter-slice interference. Also, since in real-time $v_{c,s}(t)$ and $q_{c,s}(t)$ are unknown for estimating network performance at time step $t$, we can predict the succeeding readings from previous records within $[t-H, t-1]$. Then, the set of local observations as the training input of slice QoS estimator collected during the period $[1, T]$ is given by:

$$\mathcal{X}_{t=1}^{T} := \{(x_{c,s}(t), \boldsymbol{o}_{c,s}(t)) : \text{ for } t = 1, \ldots, T, \forall c, s\}, \tag{7}$$
$$\text{where } \boldsymbol{o}_{c,s}(t) := [\boldsymbol{v}_{c,s}(t), \boldsymbol{q}_{c,s}(t), \phi_s^*, d_s^*] \in \mathbb{R}^{2H+2}. \tag{8}$$

The set of output samples is denoted by:

$$\mathcal{Y}_{t=1}^{T} := \left\{y_{c,s}(t) : \text{ for } t = 1, \ldots, T, \forall c, s\right\}, \tag{9}$$

where the slice-wise QoS satisfaction level $y_{c,s}(t)$ is computed by (3) based on the observed throughput $\phi_{c,s}(t)$ and delay $d_{c,s}(t)$.

### 2) LOCAL UTILITY APPROXIMATION

Assume we have derived the general slice QoS estimator $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Y}$ defined in (6) based on the proposed data collection approach, where the interference on each local utility is extracted by collected network observations. Then, local utility (3) is approximated with:

$$U_{c,s}(y_{c,s}(\boldsymbol{x}(t))) \approx \log(f_{\boldsymbol{\theta}}(x_{c,s}(t), \boldsymbol{o}_{c,s}(t)) + 1). \tag{10}$$

In this manner, the local general estimator model has much lower complexity and smaller sample dimensions with input $x_{c,s}(t) \in \Re$ than the global model where $\boldsymbol{x} \in \Re^C$, and the training process gains higher sample efficiency than individual training of independent models. By introducing slice-specific requirements in terms of throughput $\phi_{c,s}^*$ and delay $d_{c,s}^*$ in the training input features, the derived model $f_{\boldsymbol{\theta}}$ can handle various network slicing setups, even unseen slice types. Additionally, to extend the diversity and explore a broader spectrum of the unseen sample space, we employed the following data augmentation methods on domain samples:

(1) For samples where observed QoS falls below the requirements, i.e., $y_{c,s}(t) < 1$, we augmented samples by substituting QoS targets $(\phi_s^*, d_s^*)$ with the achieved QoS $(\phi_{c,s}(t), d_{c,s}(t))$, and setting $y_{c,s}(t)$ to 1. This adjustment implies that when the achieved QoS matches the set, the requirement satisfaction level should be 1;

(2) For samples meeting or exceeding QoS requirements i.e., $y_{c,s}(t) = 1$, we augmented samples by increasing the allocated resource portion $x_{c,s}(t)$ to a random, higher value within $[x_{c,s}(t), 1]$. Since $y_{c,s}(t)$ is upper bounded by 1, enhancing resource allocation ensures that the satisfaction level remains at its maximum according to the monotonicity of $y_{c,s}(t)$ over $x_{c,s}(t)$.

*Remark 2:* By simulating variations within the defined resource constraints and reward bounds, the applied data augmentation methods enable the exploration of unseen sample spaces, enhancing the model's generality and ensuring robust performance across various scenarios.

## B. LAGRANGIAN METHOD FOR RESOURCE ALLOCATION OPTIMIZATION

With local utility approximation (10), we can decompose the global optimization Problem 1 into distributed per-cell local optimization problems respecting inter-slice resource constraints. For each cell $c \in \mathcal{C}$ at time $t$, the local optimization problem for slice resource allocation is given by:

*Problem 2 Decomposed Local Problem:*

$$\max_{\boldsymbol{x}_c(t)} . \; F(\boldsymbol{x}_c(t)) := \sum_{s \in \mathcal{S}_c(t)} \log\left(f_{\boldsymbol{\theta}}\left(x_{c,s}(t), \boldsymbol{o}_{c,s}(t)\right) + 1\right),$$
$$\text{subject to (1), (2), } \forall t, \forall c \in \mathcal{C}, \tag{11}$$

*where $\boldsymbol{o}_{c,s}(t)$ is the local observations defined in (8).*

Problem 2 is a classical constrained non-linear optimization problem. Note that the objective in (11) is a monotonic non-decreasing function over $\boldsymbol{x} \in \mathbb{R}_+$, i.e., we have $F(\boldsymbol{x}'_c(t)) \geq F(\boldsymbol{x}_c(t))$ if $\boldsymbol{x}'_c(t) \geq \boldsymbol{x}_c(t)$ (entrywise greater). Therefore, the optimal solution to the problem with the equality constraint is also an optimal solution to the original problem, and we can solve it using the Lagrange multiplier method. Since the problem is independently formulated for each time slot $t$ and cell $c \in \mathcal{C}$, hereafter in this subsection, we omit the index of $t$ for brevity. Referring to [58], for each cell $c \in \mathcal{C}$, the Lagrangian is given by:

$$\mathcal{L}(\boldsymbol{x}_c, \lambda_c) := \sum_{s \in \mathcal{S}_c} \log\left(f_{\boldsymbol{\theta}}(\bar{\boldsymbol{x}}_{c,s}) + 1\right) + \lambda_c\Big(1 - \sum_{s \in \mathcal{S}_c} x_{c,s}\Big), \tag{12}$$

where $\bar{\boldsymbol{x}}_{c,s} := \left(x_{c,s}, \boldsymbol{o}_{c,s}\right)$, $f_{\boldsymbol{\theta}}$ is the learned estimator in (11) defined by $\boldsymbol{\theta} \in \Theta$, and $\lambda_c \in \mathbb{R}_{\geq 0}$ is the real non-negative Lagrangian multiplier. The primal and dual problems are given by:

$$\boldsymbol{x}_c^*(\lambda_c) = \arg\max_{\boldsymbol{x}_c \in \mathbb{R}_{\geq 0}} \mathcal{L}(\boldsymbol{x}_c, \lambda_c), \tag{13}$$
$$\lambda_c^* = \operatorname*{argmin}_{\lambda_c \geq 0} \mathcal{L}(\boldsymbol{x}_c^*(\lambda_c), \lambda_c). \tag{14}$$

The problems can be solved by iteratively computing the partial derivatives with respect to each variable, applying gradient descent (GD), and Lagrange multipliers accordingly [58]:

$$\begin{cases} x_{c,s}^{(i+1)} := \left[ x_{c,s}^{(i)} + \delta_x^{(i)} \cdot \dfrac{\partial \mathcal{L}\left(\boldsymbol{x}_c^{(i)}, \lambda_c^{(i)}\right)}{\partial x_{c,s}^{(i)}} \right]_+, \forall s \in \mathcal{S}_c, \\ \lambda_c^{(i+1)} := \left[ \lambda_c^{(i)} - \delta_\lambda^{(i)} \cdot \left(1 - \displaystyle\sum_{s \in \mathcal{S}_c} x_{c,s}^{(i+1)}\right) \right]_+, \end{cases} \tag{15}$$

where $i$ is the index of iteration, $\delta_x$ and $\delta_\lambda$ are the positive updating rates of $x_{c,s}, \forall s \in \mathcal{S}_c$ and $\lambda_c$, respectively, and $[x]_+$ is equivalent to $\max\{x, 0\}$. The partial derivative of $\mathcal{L}$ with respect to $x_{c,s}, \forall s \in \mathcal{S}_c$, is given by:

$$\frac{\partial \mathcal{L}\left(\boldsymbol{x}_c^{(i)}, \lambda_c^{(i)}\right)}{\partial x_{c,s}^{(i)}} = \frac{1}{f_{\boldsymbol{\theta}}\left(x_{c,s}^{(i)}\right) + 1} \cdot \frac{\partial f_{\boldsymbol{\theta}}\left(x_{c,s}^{(i)}\right)}{\partial x_{c,s}^{(i)}} - \lambda_c^{(i)}. \tag{16}$$

A significant limitation of Lagrangian methods is their difficulty in handling non-linear and non-convex functions, which can lead to multiple sub-optimums on the functional surface, potentially trapping the search in a local optimum. To overcome this challenge, we utilize the automatic differentiation capabilities of deep learning and develop the following strategies:

(1) **Transitional initialization:** Assuming that network states between two successive time steps change smoothly, we suggest initializing the starting points for the optimization of each time slot $t$ with the optimized solution of the previous time slot $t-1$, i.e., $\boldsymbol{x}_c^{(0)}(t) := \boldsymbol{x}_c^*(t-1)$;

(2) **Parallel exploration:** To find a superior among a possibly local optimum, we generate $K$ neighboring points near $\boldsymbol{x}_c^{(0)}$ and run the GD optimizations from all initial points in parallel, i.e., $\boldsymbol{x}_{c_k}^{(i)} := \boldsymbol{x}_c^{(i)} + \boldsymbol{\epsilon}$, $k \in \{1, \dots, K\}$, where $\boldsymbol{\epsilon}$ is the shifting variable for taking neighboring points with $\boldsymbol{\epsilon} \in \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and is clipped to ensure $\boldsymbol{x}_{c_k}^* \in \mathcal{X}_c$. After GD optimizations have finished, we select the best solution among them with:

$$\boldsymbol{x}_c^* := \underset{\boldsymbol{x}_{c_k}^*}{\arg\max} \sum_{s \in \mathcal{S}_c} \log\left(f_{\boldsymbol{\theta}}(x_{c_k,s}^*, \boldsymbol{o}_{c,s}) + 1\right). \tag{17}$$

Leveraging the automatic differentiation module *torch.autograd* in PyTorch [59], we can efficiently calculate the partial derivatives of the slice estimation models with respect to any input variables on tensors, i.e., the partial derivative $\partial f_{\boldsymbol{\theta}}\left(x_{c,s}^{(i)}\right)/\partial x_{c,s}^{(i)}$ as in (16) which facilitates rapid parallel computation across multiple search paths.

## V. TRANSFER LEARNING FOR IDLA ALGORITHM

In Section IV, we introduced the slice-wise resource allocation solution IDLA for scalable slice configurations within the same network scenario. However, the transferability of IDLA from one specific network scenario to another raises concerns about its performance. This is because even a minor configuration change in the network system can introduce a drift in the slice sample space, which the derived slice-based model may not have encountered before, leading to a possible degradation in the accuracy of the QoS estimator. Consequently, the Lagrangian method loses the gradient guidance necessary to find optimal solutions.

To address these challenges, in this section, we propose to enhance the generality of the estimator with a DA model structure. We first provide the domain definitions under the context of the proposed IDLA framework and formulate the DA problem of slice-wise QoS estimation model in Section V-A. Then, in Section V-B, we introduce a VIB-based estimator. The derivation of this model involves finding a common latent sample space, which extracts representative features across different slice configurations. The underlying hypothesis is that certain common hidden patterns exist in samples across diverse cells and slices. By leveraging TL, we aim to identify these representative sample space features, deriving a general slice QoS estimator model. Lastly, we evolve VIB-based estimator into IDLA approach.

### A. TRANSFER LEARNING PROBLEM FORMULATION

Before we formulate the DA problem of the IDLA solution, let us introduce two general definitions related to TL:

- **Domain**: A domain $\mathcal{D} := \{\mathcal{X}, \mathcal{Y}, P(X, Y)\}$ comprises an input feature space $\mathcal{X}$, a label space $\mathcal{Y}$, and the joint probability distribution of random variables $X$ and $Y$ with the sample space $\mathcal{X}$ and $\mathcal{Y}$, respectively.
- **Task**: A task $\mathcal{T} := \{\mathcal{Y}, f(\cdot)\}$ includes the label space $\mathcal{Y}$ and the mapping function $f : \mathcal{X} \to \mathcal{Y}$.

In the following, we use upper case letters for random variables, e.g., $X, Y$, and lower case letters for the particular realizations (measured samples) of the random variables, e.g., $\boldsymbol{x}, y$. Formally, we extend the general definition of the TL problem in [17] as follows:

*Definition 1 Transfer Learning: Given a source domain $\mathcal{D}_\mathsf{S} := \{\mathcal{X}_\mathsf{S}, \mathcal{Y}_\mathsf{S}, P_\mathsf{S}(X, Y)\}$ with a set of samples $\Omega^{(\mathsf{S})} := \left\{\left(\boldsymbol{x}_k^{(\mathsf{S})}, y_k^{(\mathsf{S})}\right) : k = 1, \dots, N_\mathsf{S}\right\}$ for solving the source learning task $\mathcal{T}_\mathsf{S} := \{\mathcal{Y}_\mathsf{S}, f(\cdot)\}$ by minimizing the training loss $l\left(f(\boldsymbol{x}), y\right)$ over all samples, where $f(\cdot)$ refers to the model being trained. The expected loss in the source domain is given by [17]:*

$$R_\mathsf{S}(f) := \mathbb{E}_{(\boldsymbol{x},y)\sim P_\mathsf{S}(\boldsymbol{x},y)}\left[l\left(f(\boldsymbol{x}), y\right)\right] \tag{18}$$

$$= \int l\left(f(\boldsymbol{x}), y\right) \cdot P_\mathsf{S}\left(\boldsymbol{x}, y\right) d\boldsymbol{x}dy. \tag{19}$$

*Define a target domain $\mathcal{D}_\mathsf{T} := \{\mathcal{X}_\mathsf{T}, \mathcal{Y}_\mathsf{T}, P_\mathsf{T}(X, Y)\}$ which only has a limited number of samples $\Omega^{(\mathsf{T})} := \left\{\left(\boldsymbol{x}_k^{(\mathsf{T})}, y_k^{(\mathsf{T})}\right) : k = 1, \dots, N_\mathsf{T}\right\}$ with $N_\mathsf{T} \ll N_\mathsf{S}$. The objective of TL is to minimize the expectation of estimation loss in the target domain with $f(\cdot)$:*

$$\begin{aligned} R_\mathsf{T}(f) &:= \mathbb{E}_{(\boldsymbol{x},y)\sim P_\mathsf{T}(\boldsymbol{x},y)}\left[l(f(\boldsymbol{x}), y)\right] \\ &= \int l\left(f(\boldsymbol{x}), y\right) \cdot \frac{P_\mathsf{T}\left(\boldsymbol{x}, y\right)}{P_\mathsf{S}\left(\boldsymbol{x}, y\right)} P_\mathsf{S}\left(\boldsymbol{x}, y\right) d\boldsymbol{x}dy \\ &= \mathbb{E}_{(\boldsymbol{x},y)\sim P_\mathsf{S}(\boldsymbol{x},y)}\left[\frac{P_\mathsf{T}\left(\boldsymbol{x}, y\right)}{P_\mathsf{S}\left(\boldsymbol{x}, y\right)} \cdot l\left(f(\boldsymbol{x}), y\right)\right]. \end{aligned} \tag{20}$$

Note that in this work, the source and target tasks are identical with the same label space $\mathcal{Y}_\mathsf{T} = \mathcal{Y}_\mathsf{S}$, the objective of TL in this work is to leverage the source data and limited target data to learn a slice QoS estimator model $f(\cdot)$ capable of performing well in the target domain. Under the context of the slice-based QoS estimator in the IDLA approach, each slice $s \in \mathcal{S}_c$ of cell $c \in \mathcal{C}$ can be regarded as a domain $\mathcal{D}_{c,s}$. While all $\mathcal{D}_{c,s}$ and $\mathcal{T}_{c,s}$, $\forall s \in \mathcal{S}_c$, $c \in \mathcal{C}$ share the same sample spaces, such as $x_{c,s} \in [0, 1]$ in (2), $\boldsymbol{o}_{c,s} \in \Re^{2H+2}$ in (8), and $y_{c,s} \in [0, 1]$ in (9), respectively. For brevity, in this section we denote $\bar{\boldsymbol{x}}_{c,s}(t) := (x_{c,s}(t), \boldsymbol{o}_{c,s}(t))$ with $\bar{\boldsymbol{x}}_{c,s}(t) \in \bar{\mathcal{X}}_{c,s}$, where $\bar{\mathcal{X}}$ stands as an extension of $\mathcal{X}$ because of adding $\boldsymbol{o}(t)$. For each domain, the set of input samples for the QoS estimator is given by:

$$\bar{\mathcal{X}}_{c,s} := \{\bar{\boldsymbol{x}}_{c,s}(t) : \text{ for } t = 1, \dots, T\} \subset \bar{\mathcal{X}}. \tag{21}$$

The set of output samples is denoted by:

$$\mathcal{Y}_{c,s} := \left\{y_{c,s}(t) : \text{ for } t = 1, \dots, T\right\} \subset \mathcal{Y}, \tag{22}$$

where $\bar{\mathcal{X}} := \bigcup_{c\in\mathcal{C},s\in\mathcal{S}_c(t)} \bar{\mathcal{X}}_{c,s}$ and $\mathcal{Y} := \bigcup_{c\in\mathcal{C},s\in\mathcal{S}_c(t)} \mathcal{Y}_{c,s}$ are the collections of input samples and output samples, respectively. Then, the local domain can be denoted as $\mathcal{D}_{c,s} := \{\bar{\mathcal{X}}, \mathcal{Y}, P_{c,s}(\bar{X}, Y)\}$, and the local task of solving local utility estimation is $\mathcal{T}_{c,s} := \{\mathcal{Y}, f_{c,s}(\cdot)\}$, where $f_{c,s} : \bar{\mathcal{X}} \to \mathcal{Y}$.

Based on TL formulation, the problem of DA for training a slice-based QoS estimator can be formulated as Problem 3. We omit the time index $t$ for brevity starting from this section.

*Problem 3 DA Problem for Slice QoS Estimator: Given a set of source domains* $\mathcal{D}_{\mathrm{S}} := \left\{\mathcal{D}_{c,s} : c \in \bar{\mathcal{C}}, s \in \bar{\mathcal{S}}_c\right\}$, *where each domain comprises collected samples from source cells* $\bar{\mathcal{C}} \subset \mathcal{C}$ *and slices* $\bar{\mathcal{S}}_c \subset \mathcal{S}_c$, *and a target domain* $\mathcal{D}_{\mathrm{T}} := \left\{\bar{\mathcal{X}}, \mathcal{Y}, P_{\mathrm{T}}(\bar{X}, Y)\right\}$ *representing the slice* $s' \in \mathcal{S}_{c'}$ *of cell* $c' \notin \bar{\mathcal{C}}$, *we aims to derive a slice-based QoS estimator* $f^{(\mathrm{TL})}(\cdot)$ *based on the samples from* $\mathcal{D}_{\mathrm{S}}$, *such that the expectation of QoS estimation loss over samples in* $\mathcal{D}_{\mathrm{T}}$ *is minimized. The problem is given by*

$$\begin{aligned}\min_{f^{(\mathrm{TL})}}.\ R_{\mathrm{T}}\left(f^{(\mathrm{TL})}\right) &:= \mathbb{E}_{(\bar{\mathbf{x}},y)\sim P_{\mathrm{T}}(\bar{\mathbf{x}},y)}\left[l\left(f^{(\mathrm{TL})}(\bar{\mathbf{x}}), y\right)\right]\\ &= \mathbb{E}_{(\bar{\mathbf{x}},y)\sim P_{\mathrm{S}}(\bar{\mathbf{x}},y)}\left[\frac{P_{\mathrm{T}}(\bar{\mathbf{x}}, y)}{P_{\mathrm{S}}(\bar{\mathbf{x}}, y)} l\left(f^{(\mathrm{TL})}(\bar{\mathbf{x}}), y\right)\right],\\ &\text{subject to (21), (22).}\end{aligned} \tag{23}$$

$P_{\mathrm{S}}(\bar{\mathbf{x}}, y) := \sum_{s\in\bar{\mathcal{S}}_c,c\in\bar{\mathcal{C}}} \omega_{c,s}P_{c,s}(\bar{\mathbf{x}}, y)$ *is the mixture distribution of samples from all source domains used to derive the model* $f^{(\mathrm{TL})} : \bar{\mathcal{X}} \to \mathcal{Y}$, *where* $\omega_{c,s}$ *are the weights assigned to each local distribution with* $\sum_{s\in\bar{\mathcal{S}}_c,c\in\bar{\mathcal{C}}} \omega_{c,s} = 1$. *In this work, we assume the domain weights are equally distributed as* $\omega_{c,s} := 1/(|\bar{\mathcal{C}}| \cdot |\bar{\mathcal{S}}_c|), \forall s \in \bar{\mathcal{S}}_c, c \in \bar{\mathcal{C}}$. *The rationale behind using equal weights across source domains is to ensure that each domain contributes uniformly since there is no prior knowledge about the relative importance of specific domains, allowing the model to generalize more effectively across all domains. Equal weighting also simplifies the problem by normalizing the influence of each domain, promoting the learning of transferable features to the target domain.* $P_{\mathrm{T}}(\bar{\mathbf{x}}, y)$ *denotes the distribution of samples collected in slice* $s'$ *of cell* $c'$.

The challenges in developing an adaptive QoS estimator $f^{(\mathrm{TL})}(\cdot)$, that performs well on both source and target domains, arise primarily from the differences between $P_{\mathrm{S}}(\bar{\mathbf{x}}, y)$ and $P_{\mathrm{T}}(\bar{\mathbf{x}}, y)$. In our problem, the source and target task functions are the same, i.e., $P_{c_i,s_p}(y|\bar{\mathbf{x}}) = P_{c_j,s_q}(y|\bar{\mathbf{x}})$ with respect to a common slice-based estimator $f^{(\mathrm{TL})}(\bar{\mathbf{x}}) = y$ in (6). Hence, we can express the relationship as:

$$\frac{P_{\mathrm{T}}(\bar{\mathbf{x}}, y)}{P_{\mathrm{S}}(\bar{\mathbf{x}}, y)} = \frac{P_{\mathrm{T}}(\bar{\mathbf{x}})\cancel{P_{\mathrm{T}}(y|\bar{\mathbf{x}})}}{P_{\mathrm{S}}(\bar{\mathbf{x}})\cancel{P_{\mathrm{S}}(y|\bar{\mathbf{x}})}} = \frac{P_{\mathrm{T}}(\bar{\mathbf{x}})}{P_{\mathrm{S}}(\bar{\mathbf{x}})}. \tag{24}$$

The difference between domains depends solely on the distributions of input samples. This allows us to rewrite (23) as:

$$\min_{f^{(\mathrm{TL})}}.\ \mathbb{E}_{(\bar{\mathbf{x}},y)\sim P_{\mathrm{S}}(\bar{\mathbf{x}},y)}\left[\frac{P_{\mathrm{T}}(\bar{\mathbf{x}})}{P_{\mathrm{S}}(\bar{\mathbf{x}})} \cdot l\left(f^{(\mathrm{TL})}(\bar{\mathbf{x}}), y\right)\right]. \tag{25}$$

## *B. A VIB-BASED QoS ESTIMATION*

From (25), it is evident that the TL problem becomes an authentic machine learning problem if $P_{\mathrm{T}}(\bar{\mathbf{x}}) = P_{\mathrm{S}}(\bar{\mathbf{x}})$, i.e., training and validating model $f^{(\mathrm{TL})}$ on the same sample distribution. We aim to map samples from different domains into a common representative latent feature space. This common space should extract features informative enough to distinguish and represent the patterns of various domains, allowing the derivation of a general QoS estimator capable of handling samples from different distributions.

### 1) INFORMATION BOTTLENECK

In this work, we propose to derive an adaptive slice-based network QoS estimator $f^{(\mathrm{TL})}$ based on source domain samples from $P_{\mathrm{S}}(\bar{\mathbf{x}}, y)$ that performs well on target samples from $P_{\mathrm{T}}(\bar{\mathbf{x}}, y)$ using a domain adaptation approach inspired by the VIB method [23]. Specifically, we aim to find intermediate representations of $\bar{X}$ as latent variables $Z$ by solving an encoder $g_{\boldsymbol{\epsilon}}$ parameterized by $\boldsymbol{\epsilon}$. We aim for $Z \in \mathcal{Z}$ to be maximally informative about the training output $Y \in \mathcal{Y}$. The informative level is measured by the mutual information referring to [23]:

$$I(Z, Y; \boldsymbol{\epsilon}) := \int P(z, y|\boldsymbol{\epsilon}) \log \frac{P(z, y|\boldsymbol{\epsilon})}{P(z|\boldsymbol{\epsilon})P(y|\boldsymbol{\epsilon})} dzdy. \tag{26}$$

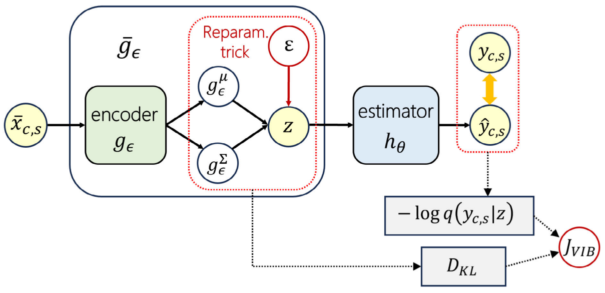

**FIGURE 2. VIB-based slice QoS estimator. The proposed VIB model consists of two sub-models: $g_\epsilon$, which acts as the encoder, and $h_\theta$, which serves as the QoS estimator.**

We aim for a representative space $\mathcal{Z}$ that is informative enough for capturing $\mathcal{Y}$ while maintaining a low dependency on $\bar{\mathcal{X}}$. To restrict the complexity of the representation space, we introduce a constraint on the mutual information between $\bar{X}$ and $Z$ as $I(Z, \bar{X}; \boldsymbol{\epsilon}) \le I_c$. Here, $I_c$ represents the information bottleneck, which sets the limit of information about $\bar{X}$ that $Z$ can obtain. Referring to [21], the problem of finding the encoder $g_{\boldsymbol{\epsilon}}$ is given by:

$$\max_{\boldsymbol{\epsilon}}.\ I(Z, Y; \boldsymbol{\epsilon}),\ \ \text{subject to}\ I(Z, \bar{X}; \boldsymbol{\epsilon}) \le I_c. \tag{27}$$

And by introducing a constant $\beta \ge 0$, the objective function for solving (27) is:

$$\mathcal{L}_{IB}(\boldsymbol{\epsilon}) := I(Z, Y; \boldsymbol{\epsilon}) - \beta I(Z, \bar{X}; \boldsymbol{\epsilon}). \tag{28}$$

By maximizing the first term of (28), the function encourages the representation $Z$ to be predictive of output $Y$, while minimizing the second term encourages $Z$ to "forget" $\bar{X}$. This means $Z$ is forced to be a minimal sufficient statistic of $\bar{X}$ for predicting $Y$. This objective function aligns with

Shannon's rate-distortion theory [60], where $I(Z, \bar{X})$ represents the "rate" of information about $\bar{X}$ encoded in $Z$, and $I(Z, Y)$ inversely relates to "distortion," reflecting how well the representation $Z$ predicts $Y$. The weight factor $\beta$ adjusts the trade-off between the information compression rate and prediction distortion.

### 2) VIB-BASED MODEL

To derive the solution for (28), we propose using the VIB approach, approximating the problem with variational inference. We assume $P(z|\bar{\boldsymbol{x}}, y) = P(z|\bar{\boldsymbol{x}})$ corresponding to the Markov chain $\mathcal{Y} \rightarrow \mathcal{X} \rightarrow \mathcal{Z} \rightarrow \hat{\mathcal{Y}}$, i.e., the representation variables $Z$ are not directly dependent on output labels $Y$, while $\hat{Y}$ denoted the prediction of $Y$ generated from $\mathcal{Z}$ without direct dependence on $Y$. Under this assumption, the joint distribution of $P(\bar{\boldsymbol{x}}, z, y)$ can be factorized as:

$$\begin{aligned} P(\bar{\boldsymbol{x}}, z, y) &= P(z|\bar{\boldsymbol{x}}, y)P(y|\bar{\boldsymbol{x}})P(\bar{\boldsymbol{x}}) \\ &= P(z|\bar{\boldsymbol{x}})P(y|\bar{\boldsymbol{x}})P(\bar{\boldsymbol{x}}). \end{aligned} \tag{29}$$

Recalling the objective function (27), besides $P(\bar{\boldsymbol{x}}, y)$ which is determined by the sample distribution, the only content we need now is the encoding function $P(z|\boldsymbol{x})$, while other distributions can be derived based on the Markov chain.

Following the derivation process of VIB, we can simplify two terms in (28) respectively:

$$I(Z, Y) \geq \int P(\bar{\boldsymbol{x}})P(z|\bar{\boldsymbol{x}})P(y|\bar{\boldsymbol{x}}) \log Q(y|z) d\bar{\boldsymbol{x}} dz dy, \tag{30}$$

$$I(Z, \bar{X}) \leq \int P(\bar{\boldsymbol{x}})P(z|\bar{\boldsymbol{x}}) \log \frac{P(z|\bar{\boldsymbol{x}})}{R(z)} d\bar{\boldsymbol{x}} dz, \tag{31}$$

where $Q(y|z)$ and $R(z)$ are the variational approximations of $P(y|z)$ and $P(z)$ respectively. It is necessary to address that our proposed VIB-based estimator aims to extract latent variables $Z$ as sufficient representatives of the original slice samples from different domains and thereby capture common latent features in between to conduct robust slice QoS estimations.

In this way, solving the Lagrangian (28) boils down to maximizing $\mathcal{L}_{IB}$ to find the optimal $g_{\boldsymbol{\epsilon}}$ for encoding $P(z|\bar{\boldsymbol{x}}; \boldsymbol{\epsilon})$. In the context of slice-based QoS estimation presented in this work, the joint sample distribution $P(\bar{\boldsymbol{x}}, y)$ can be approximated based on collected samples from domains, as indicated by (21) and (22), using the empirical distribution. Therefore, in practice, we can compute $\mathcal{L}_{IB}$ by:

$$\begin{aligned} \mathcal{L}_{IB} \approx \frac{1}{N_{\mathrm{S}}} \sum_{c \in \bar{\mathcal{C}}, s \in \bar{\mathcal{S}}_c} \Big[ &\sum_{z} P(z|\bar{\boldsymbol{x}}_{c,s}) \log Q(y_{c,s}|z) \\ &- \beta\, P(z|\bar{\boldsymbol{x}}_{c,s}) \log \frac{P(z|\bar{\boldsymbol{x}}_{c,s})}{R(z)} \Big], \end{aligned} \tag{32}$$

where $N_{\mathrm{S}}$ is the number of collected samples from source domains. Assuming $P(z|\bar{\boldsymbol{x}}_{c,s})$ follows a Gaussian distribution, we define the encoding $P(z|\bar{\boldsymbol{x}}_{c,s}) := \mathcal{N}(z|g^{\mu}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}), g^{\Sigma}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}))$ with respect to the encoder $g_{\boldsymbol{\epsilon}}$. To enable differentiability during backpropagation, we sample $z$ from $P(z|\bar{\boldsymbol{x}}_{c,s})$ in the first term of (32) using the reparameterization trick [61], where $\varepsilon \sim \mathbb{N}(0, I)$ is a Gaussian random variable. For simplicity, we represent this as the deterministic function $\bar{g}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}, \varepsilon)$, combining the encoder $g_{\boldsymbol{\epsilon}}$ and reparameterization. In the second term of (32), $\beta\ P(z|\bar{\boldsymbol{x}}) \log \left(P(z|\bar{\boldsymbol{x}})/R(z)\right)$ corresponds to the Kullback-Leibler (KL) divergence $D_{KL}$ between the learned posterior $P(z|\bar{\boldsymbol{x}})$ and the prior $R(z)$, which measures how much the learned distribution $P(z|\bar{\boldsymbol{x}})$ deviates from the reference prior $R(z)$.

In summary, we can derive the VIB model for slice-based QoS estimation with domain adaptation by minimizing the following objective function:

$$\begin{aligned} J_{VIB} = \frac{1}{N_{\mathrm{S}}} \sum_{c \in \bar{\mathcal{C}}, s \in \bar{\mathcal{S}}_c} \mathbb{E}\Big[ &- \log Q(y_{c,s}|\bar{g}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}, \varepsilon)) \\ &+ \beta\, D_{KL}\big[P(z|\bar{\boldsymbol{x}}_{c,s}) \| R(z)\big]\Big]. \end{aligned} \tag{33}$$

The variational estimation $Q(y_{c,s}|\bar{g}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}, \varepsilon))$ can be defined as a DNN model $h_{\boldsymbol{\theta}} : \mathcal{Z} \rightarrow \mathcal{Y}$ for QoS estimation, considering the derived representations $z = \bar{g}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}, \varepsilon)$. Therefore, the proposed VIB model for QoS estimation comprises two sub-models: the encoder $g_{\boldsymbol{\epsilon}}$, parameterized by $\boldsymbol{\theta} \in \Theta$, and the QoS estimator $h_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\epsilon} \in \mathcal{E}$. And the loss function comprises two terms: the negative log-likelihood and the KL divergence. The architecture of the model is illustrated in Fig. 2.

## C. VIB-AIDED IDLA ALGORITHM

In Section IV-A, we propose to collect slice-wise network observations as in (8) to capture common features from different slices for learning a general slice QoS estimator $f_{\boldsymbol{\theta}}$. For DA problems, the model is required to be generalizable to different source domains but also "predictable" for adapting to unseen domains. This implies that the DA model should be capable of distinguishing domains and learning hidden patterns through the collected source samples. Following the approach proposed in Section V-B, we can derive a VIB-based model as an adaptive QoS estimator.

Based on the VIB model, in the IDLA approach, the Lagrangian function for solving local Problem 2 of each cell $c \in \mathcal{C}$ is reformulated as:

$$\begin{aligned} \mathcal{L}_{VIB}(\boldsymbol{x}_c, \lambda_c; \varepsilon) := &\sum_{s \in \mathcal{S}_c} \log \left(h_{\boldsymbol{\theta}} \circ \bar{g}_{\boldsymbol{\epsilon}}(\bar{\boldsymbol{x}}_{c,s}, \varepsilon) + 1\right) \\ &+ \lambda_c \Big(1 - \sum_{s \in \mathcal{S}_c} x_{c,s}\Big). \end{aligned} \tag{34}$$

Similarly, we can solve the primal and dual problems and solve each local optimization with GD iteratively:

$$\begin{cases} x_{c,s}^{(i+1)} := \left[ x_{c,s}^{(i)} + \delta_x^{(i)} \cdot \dfrac{\partial \mathcal{L}_{VIB}\left(\boldsymbol{x}_c^{(i)}, \lambda_c^{(i)}\right)}{\partial x_{c,s}^{(i)}} \right]_+, \forall s \in \mathcal{S}_c, \\ \lambda_c^{(i+1)} := \left[ \lambda_c^{(i)} - \delta_\lambda^{(i)} \cdot \left(1 - \displaystyle\sum_{s \in \mathcal{S}_c} x_{c,s}^{(i+1)}\right) \right]_+. \end{cases} \tag{35}$$

The VIB model is composed of models $g_{\epsilon}$ and $h_{\theta}$, and their differentiability is guaranteed by implementing the reparameterization trick for sampling latent representation $z$, the derivatives of $\mathcal{L}_{VIB}$ are derivable with respect to $x_{c,s}$ as:

$$\frac{\partial \mathcal{L}_{VIB}}{\partial x_{c,s}^{(i)}} = \frac{1}{h_{\theta} \circ \bar{g}_{\epsilon} + 1} \cdot \frac{\partial h_{\theta} \circ \bar{g}_{\epsilon}}{\partial \bar{g}_{\epsilon}} \cdot \frac{\partial \bar{g}_{\epsilon}}{\partial x_{c,s}^{(i)}} - \lambda_c^{(i)}. \tag{36}$$

With these changes, the VIB can further proceed with the slice resource allocation following the pipeline of the IDLA algorithm. Similarly, to ensure a robust and efficient optimization process, for VIB-aided IDLA algorithm, we also implement the searching strategies presented in Section IV-B. Specifically, for parallel exploration in $K$ different paths, we select the optimal among them after convergence with:

$$\boldsymbol{x}_c^* := \arg\max_{\boldsymbol{x}_{c_k}^*} \sum_{s \in \mathcal{S}_c} \log \left( h_{\theta} \circ \bar{g}_{\epsilon} \left( x_{c_k,s}^*, \boldsymbol{o}_{c,s}, \varepsilon \right) + 1 \right). \tag{37}$$

In Algorithm 1, we demonstrate the complete process of IDLA algorithm with VIB-based estimator, where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $I^{(\max)}$, and $\xi$ denote the normal distribution for taking neighboring points with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, the maximum iteration steps, and criterion for stopping iteration, respectively.

**Algorithm 1** VIB-Aided IDLA Algorithm

```
1:  for $t \in T$ and $c \in C$ do
2:      $i \leftarrow 0$
3:      $x_c^{(i)}(t) \leftarrow \begin{cases} \textit{default action}, & \text{if } t = 0 \\ x_c^*(t-1), & \text{otherwise} \end{cases}$
4:      Take $K$ neighboring points as:
5:      $x_{c_k}^{(i)}(t) := x_c^{(i)}(t) + \epsilon$, $k \in [1, \ldots, K]$, with $\epsilon \in \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
6:      Parallelly compute for all $k \in [1, \ldots, K]$:
7:      Initialize Lagrangian multiplier $\lambda_{c_k}^{(i)}$
8:      Initialize update rates $\delta_x^{(i)} > 0$, $\delta_\lambda^{(i)} > 0$
9:      while $i \leq I^{(\max)}$ and $\|x_{c_k}^{(i)}(t) - x_{c_k}^{(i-1)}(t)\| \geq \xi$ do
10:         Compute partial derivatives with respect to (36)
11:         Update optimization variables multipliers with (35)
12:         Decrease update rates $\delta_x^{(i)}$, $\delta_\lambda^{(i)}$
13:         $i \mathrel{+}= 1$
14:     end while
15:     $x_{c_k}^*(t) \leftarrow x_{c_k}^{(i)}(t)$
16:     Choose the optimal among $K$ points with (37)
17: end for
```

*Remark 3: [Complexity Analysis of Algorithm 1] For each cell $c \in \mathcal{C}$, we solve the decomposed constrained nonlinear problem in (11) using the Lagrangian method, where a neural network approximates the nonlinear function. We parallelly compute the candidate solutions starting from $K$ initial starting points by performing (35) for maximum $I^{(\max)}$ iterations and choose the best solution among them. The computational complexity of each iteration consists of the following components*:

- *Computation of the gradient of Lagrangian (36) for all slices $s \in \mathcal{S}_c$ in each cell. The complexity of computing (36) is dominated by two computations: computing the forward pass through $h_{\theta} \circ \bar{g}_{\epsilon}(\cdot)$ with complexity $\mathcal{O}(|\Theta| + |\mathcal{E}|)$, where $\Theta$ and $\mathcal{E}$ are the parameter spaces of the neural networks $h_{\theta}$ (the QoS estimator) and $\bar{g}_{\epsilon}$ (the encoder) respectively; and computing the gradient $(\partial h_{\theta} \circ \bar{g}_{\epsilon} / \partial \bar{g}_{\epsilon}) \cdot \left( \partial \bar{g}_{\epsilon} / \partial x_{c,s}^{(i)} \right)$ using back propagation, also with complexity $\mathcal{O}(|\Theta| + |\mathcal{E}|)$. The overall complexity to compute the Lagrangian gradient for all slices in cell $c$ is then $\mathcal{O}((|\Theta| + |\mathcal{E}|) \cdot |\mathcal{S}_c|)$;*
- *Updating the multipliers $\lambda_c$ with complexity $\mathcal{O}(|\mathcal{S}_c|)$.*

*For each cell $c \in \mathcal{C}$, for solution selection, we perform the above two steps for maximum $I^{(\max)}$ iterations from $K$ starting points parallelly, and find the best solution with (37). Computing (37) also involves complexity of $\mathcal{O}((|\Theta| + |\mathcal{E}|) \cdot |\mathcal{S}_c|)$. Therefore, summing up over all cells in $\mathcal{C}$, the overall complexity of Algorithm 1 is $\mathcal{O}\left( H\mathcal{I}^{(\max)} (|\Theta| + |\mathcal{E}|) \sum_{c \in \mathcal{C}} |\mathcal{S}_c| \right)$.*

## VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed VIB-based model and IDLA algorithm under a system-level network simulator Season II [62], where we can define various types of slices regarding slice service requirements and build up network environments with arbitrary scales. First, we introduce the network environments for assessing our proposed methods, focusing on analyzing sample-based domain discrepancies. We derive slice-wise network QoS estimators using samples gathered from corresponding domains. Subsequently, to evaluate the slice resource allocation performance of the VIB-aided IDLA algorithm in real-time, we deployed the model in an evolving network slicing scenario within Season II, benchmarking its performance against other state-of-the-art solutions, including a IDLA approach with DNN-based estimator, a DRL approach and a traffic-aware resource allocation method. Finally, we assess the adaptability and scalability of the VIB-based models by comparing their DA efficacy against traditional DNN models and a sample re-weighting baseline to address their DA performance under data imbalance scenarios.

We deployed the model in an evolving network slicing scenario within Season II, benchmarking its performance against other state-of-the-art solutions.

### A. NETWORK SETTING

To implement the proposed methods and evaluate their transferability and scalability, we aimed to build a network system that can imitate a realistic scenario where slicing settings, such as slice requirements or served user behaviors, may vary within a consistent network architecture. Therefore, to induce domain disparities, we constructed a virtual network system consisting of 4 three-sector base stations operating at 2.6 GHz located in a small area of Helsinki city, represented by a network environment with $C = 12$ cells under the buildings map of Helsinki in Season II. All cells are provided with the same bandwidth of 20 MHz.

We defined a set of network slices with distinct service requirements with varying user throughput $\phi^*$ varying from 0.5 to 4 Mbit/s. The slice combination of any cell $c$ can be configurable and time-varying, while each slice $s \in \mathcal{S}_c(t)$ is correspondingly assigned to a specific service. Besides, to manually create diverse user behaviors across cells, we categorized users into groups with varying moving radius $U^{(\mathrm{R})}$ from 200 to 100 meters, and the maximum user number per service $U^{(\mathrm{N})}$ changing between [12], [36].

These deliberate discrepancies among slices and user groups aim to simulate diverse domain sample characteristics under an identical network architecture. During experiments, we have the flexibility to construct slicing scenarios by selecting various combinations of slices and user groups under the structure of the pre-built network system. Specifically, to address the performance of the proposed DA method, we can acquire network data from a particular set of cells, denoted as source domains, to train slice-aware network performance estimation models. Subsequently, we evaluate these models' efficacy on the cell not included in the source domain, i.e., denoted as the target domain, to assess their transferability. For convenience, we omit the units in the rest of the section.

## B. DATA COLLECTION AND TRAINING

In this subsection, we first outline the process for collecting training samples from the network system we established above for training slice-wise QoS estimators, followed by introducing training setups for these estimators, including the specification of training hyperparameters. We explored various configurations of slices and user groups to generate diverse per-slice samples for DA assessment. The per-slice samples that exhibited common characteristics were aggregated as source domains for training slice-wise QoS estimators. Subsequently, we deployed these models on target domain samples after training convergence to evaluate the DA features of the proposed estimator model. These target domains, while distinct, retain relevance to their corresponding source domains. In Subsection VI-D, we introduce the selection of source and target domain pairs by identifying specific combinations of network slice setups as indicated in Table 2 to facilitate evaluation.

### 1) DOMAIN SAMPLE COLLECTION AND AUGMENTATION

Before implementing the training of proposed slice QoS estimators, we first demonstrate the data collection process within the Season II network system. Following the data collection pipeline presented in IV-A, we collect per-slice samples across all cells, incorporating $H = 5$ steps of historical data to enhance the capture of temporal correlations. Consenquently, this forms the training input $[x_{c,s}, \boldsymbol{o}_{c,s}] \in \Re^{17}$ refer to (8).

It is worth mentioning that consistent with DA assumption, we have limited data volumes in target domains, which results in a much smaller set of samples compared to those from source domains. Before initiating the training of QoS estimators with these per-slice samples, we analyzed the discrepancy between source and target domains. This preliminary step was essential for assessing the feasibility and potential challenges of implementing DA techniques across domains. Accordingly, we derived the domain discrepancy measurements utilizing the approach from our previous work [63]. This discrepancy metric is determined by averaging the KL divergence [64] across latent variables of the source and target samples extracted by a variational autoencoder (VAE) with:

$$\mathfrak{D}_{i,j} := \frac{1}{N_i N_j} \cdot \sum_{\substack{(\boldsymbol{\mu}_n, \boldsymbol{\sigma}_n); \\ (\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m)}} D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}_n, \mathrm{diag}(\boldsymbol{\sigma}_n)) \| \mathcal{N}(\boldsymbol{\mu}_m, \mathrm{diag}(\boldsymbol{\sigma}_m))\right), \quad (38)$$

where $i, j$ are the domain labels and $(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)k \in i, j$ are the parameters of latent variable distributions extracted with $P^{\mathrm{VAE}}(z_k|\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{\mu}_k, \mathrm{diag}(\boldsymbol{\sigma}_k))$ from $\boldsymbol{x}_n \in \mathcal{X}_i$ and $\boldsymbol{x}_m \in \mathcal{X}_j$. Please note that the encoder used here differs from the one in the VIB-based estimation model.

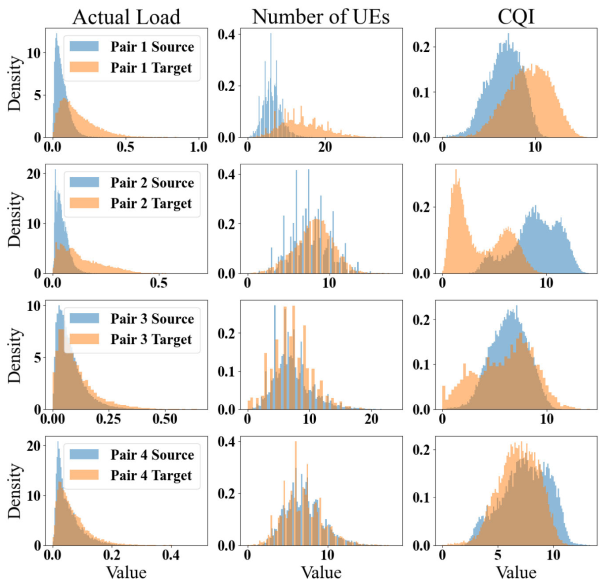

**FIGURE 3. Compare distributions of input between domains. Here, we compare sample distributions regarding four key parameters: Actual Load, Number of Users (UEs), and Channel Quality Indicator (CQI) between the source and target domains for each evaluated domain pair. This visualization highlights the domain differences for the considered parameters across different source-target pairs.**

### 2) TRAINING OF THE ESTIMATION MODEL

In this section, we evaluate the DA performance of proposed models by comparing slice QoS estimation accuracy across source and target domains in each defined comparison pair. Regarding the DA approach, in addition to our proposed VIB model, we implement a label distribution smoothing (LDS) [26] technique to address the data distribution imbalance issue across different slices. LDS decicates to smooth the domain label distribution by convolving the empirical distribution with a symmetric kernel. In this work, we obtained a smoothed version of the domain label distribution. This smoothed distribution was used to reweight the loss function during training, inversely proportional to each domain label's effective density, thereby emphasizing rarer domain samples and improving the model's performance on underrepresented data. The weighted estimation loss in the form of the mean absolute error (MAE) for LDS training can be expressed as:

$$L_{LDS} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot |\hat{y}_i - y_i|. \tag{39}$$

The weights for each sample are given as $w_i = 1/\tilde{p}(y_i)$, where $\tilde{p}(y_i)$ represents the effective presenting frequency of $y_i$ determined by its distribution, emphasizing domains with fewer samples to potentially mitigate the impact of label imbalance. Specifically, we compare the following estimation models with corresponding model configurations:

- **VIB**: The VIB-based model incorporated both an encoder and an estimator, both structured as MLPs with hidden layers (36, 24, 16) for the encoder and (24, 16, 16) for the estimator, respectively. To achieve a good trade-off between representative richness and estimation precision, we mapped the original input samples $[x_{c,s}, \boldsymbol{o}_{c,s}] \in \Re^{17}$ into latent variables $z_{c,s} \in \Re^{12}$. It is trained by targeting (33), where $\beta$ was set to 0.002 to balance the model's ability to generalize across domains while maintaining meaningful information in the latent representation. The distribution $r(z)$ of the latent variables is assumed as a standard Gaussian $\mathbb{N}(0, I)$.
- **DNN**: The conventional DNN-based estimator, constructing with a MLP architecture comprising 5 hidden layers with neurons (64, 36, 24, 16, 16), targeting MAE as estimatino loss metric $L_{DNN} = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i|$;
- **LDS**: The LDS technique, while not a model itself, is applied to the DNN estimator to address label imbalance through effective reweighting. This enhances the DA performance of the DNN estimator and serves as a baseline for assessing the performance of VIB models. The LDS approach is incorporated into the DNN setup by training with a weighted loss function, as described in (39).

For each domain pair, we trained all model types for 200 epochs with a learning rate of $5 \times 10^{-3}$, batch size of 32, using the Adam optimizer. Early stopping with a patience of 10 epochs was applied to prevent overfitting. The training was conducted on two Nvidia GeForce 2080Ti GPUs. Both the VIB and DNN-based IDLA approaches took less than 0.5 seconds in real time to complete all slice partitions in our experimental network systems at each time step. Additionally, each iteration of GD took about 2 milliseconds.

In our previous work [1], we validated the effectiveness of the DNN-based estimator on both simulated and real-world datasets. Additionally, we applied a realistic traffic mask (see Fig. 4) to closely replicate real-world network conditions, further ensuring the proposed methods' applicability and robustness in practical deployment environments. These efforts help to bridge the sim-to-real gap and demonstrate the real-world feasibility of our approach.

## C. PERFORMANCE COMPARISON OF RESOURCE ALLOCATION

In this subsection, we assess the online performance of slice QoS estimators by implementing the proposed IDLA procedure presented in IV. Our objectives were twofold: firstly, we intend to evaluate the online performance of the estimators within the framework of the proposed IDLA algorithm, and secondly, we aim to explore the adaptability of the IDLA approach in practical network dynamics. Specifically, we investigated how the IDLA would respond to domain shifts when facing network slicing changes. In this subsection, we implement the IDLA approaches based on derived models in a scenario with dynamically changing networks. For comparison, we also implemented a DRL-based approach and a traffic-aware slice resource allocation mechanism as baselines. In detail, the following schemes were implemented for online comparisons:

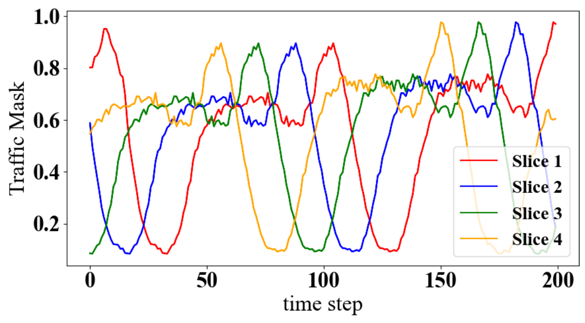

**FIGURE 4. Traffic masks for slices in $\mathcal{H}_3$. Varying traffic masks $\tau_s(t) \in [0, 1]$ were applied to each slice $s \in \mathcal{S}_c(t)$ to simulate realistic user traffic patterns based on data collected from a real network system. The first 192 steps of the traffic mask for $\mathcal{H}_3$ are shown here, aligned with the other phases.**

- **IDLA-VIB**: the proposed IDLA algorithm based on the VIB-based slice QoS estimator. The IDLA algorithm is implemented by setting the number of initial neighboring points $P = 5$ with offset $\varepsilon \sim \mathbb{N}(0, 0.05)$;
- **IDLA** scheme: the proposed IDLA algorithm with a DNN-based QoS estimator. The initial neighboring point setting is identical to the **IDLA-VIB**;
- **DRL**: a distributed cell-wise DRL approach using a twin delayed deep deterministic policy gradient (TD3) algorithm, similar to our prior work [9], which solves optimal slice resource portions in a cell-wise manner regarding defined DRL reward as the minimum QoS satisfaction level among all slices;

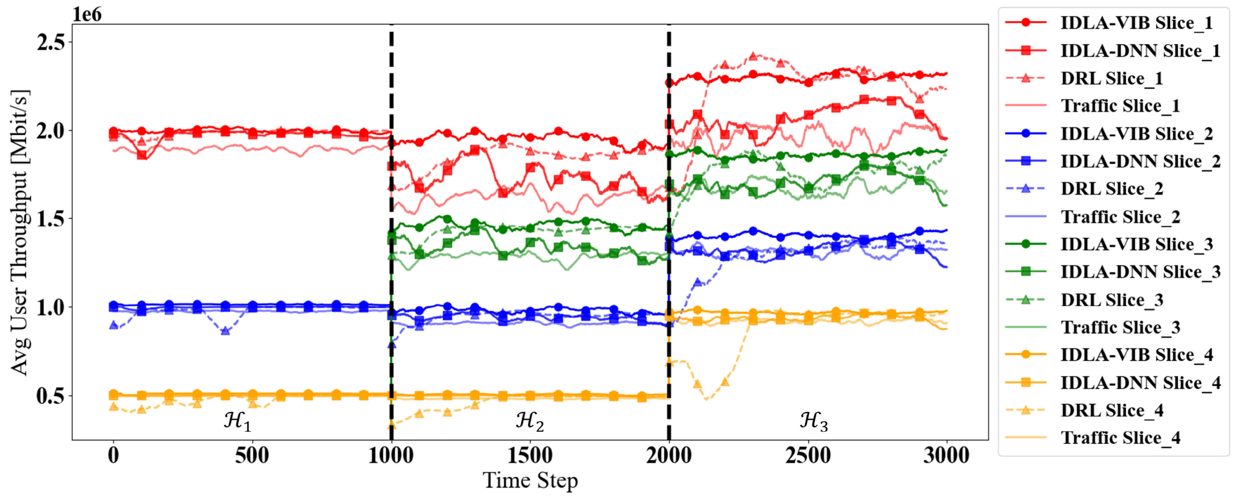

**FIGURE 5. Comparison of average user throughput for different dynamic slicing schemes across three phases ($\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$).**

- **Traffic**: a traffic-aware approach that dynamically adapts slice resource portions in each cell proportionally to the current per-slice user traffic demands, which assumes perfect knowledge of the traffic amount;

We applied the IDLA approaches with the help of derived slice performance estimators, i.e., VIB-based and DNN-based models, that were derived with source domain data from the corresponding network environments. To evaluate the adaptation performance of the IDLA-VIB scheme against the others, we implemented an online assessment in the network simulator under dynamic network slicing configurations and various user groups, i.e., during the network processing, we changed network slicing combinations or user behaviors. For fair comparisons, we divided the whole online process into 3 time periods, denoted by $H_1, H_2, H_3$, respectively:

- $\mathcal{H}_1$ ($t \in [0, 1000)$): First, in time phase $\mathcal{H}_1$ we set 3 slices with combination $\mathcal{S}_c(t) := [1, 2, 4], t \in \mathcal{H}_1, c \in \mathcal{C}$ in network, with slice throughput requirements as $\phi^*_{\mathcal{H}_1} \in \{2.0, 1.0, 0.5\}$ Mbit/s respectively.
- $\mathcal{H}_2$ ($t \in [1000, 2000)$): At time step 2000, the network scenario enters phase $\mathcal{H}_2$, and we changed the network slicing configuration by introducing a new slice with index 3, i.e., the new slice combination becomes $\mathcal{S}_c(t) := [1, 2, 3, 4]$. The new slice throughput requirements become $\phi^*_{\mathcal{H}_2} \in \{2.0, 1.0, 1.5, 0.5\}$ Mbit/s.
- $\mathcal{H}_3$ ($t \in [2000, 3000]$): We further changed the slicing configuration at the switching point between phase $\mathcal{H}_2$ and $\mathcal{H}_3$ by increasing the throughput requirement of each slice by 0.5 Mbit/s, i.e., the new slice requirements are $\phi^*_{\mathcal{H}_3} \in \{2.5, 1.5, 2.0, 1.0\}$ Mbit/s.

To assess transferability, during this process, we kept collecting new samples from the network simulators and finetuned the estimators periodically every 200 steps for 100 epochs for **IDLA-VIB** and **IDLA** schemes, while the **DRL** finetunes itself through online interaction when entering a new phase with new slicing setups. For fairness

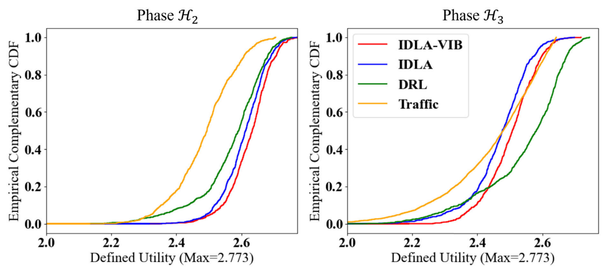

**FIGURE 6. Comparison of the CDF of network utility during phase $\mathcal{H}_2$ and phase $\mathcal{H}_3$ for different resource allocation schemes.**

comparison, the initial exploration phase of the **DRL** scheme was excluded at the beginning of $\mathcal{H}_1$. To simulate realistic user traffic patterns, we applied varying traffic masks $\tau_s(t) \in [0, 1]$ collected from a real network system to each slice $s \in \mathcal{S}_c(t)$, reflecting the daily periodic pattern of the per-slice user traffic, where each step represents 15 minutes in real time. To avoid overlapping of traffic peaks, on each slice, the traffic mask has 16 steps shift to the next. For instance, Fig. 4 illustrates the first 192 steps of the applied traffic mask for $\mathcal{H}_3$.

Fig. 5 illustrates the trends in average throughput across all cells for each slice during phase $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ by implementing different slice resource allocation schemes. Both the **IDLA-VIB** and **IDLA** schemes maintain consistent and stable throughput levels across all slices before encountering network condition changes, with the **IDLA-VIB** scheme offering slightly improved and more robust performance. In contrast, the **DRL** and **Traffic** schemes exhibit higher fluctuations, especially after each configuration change. At the network change points, a clear impact on throughput is observed for all schemes. However, the IDLA-based schemes show better adaptability and stabilize within approximately 300 time steps, with the **IDLA-VIB** scheme demonstrating even faster convergence due to its

**TABLE 2. Table of source-target domain pairs. Here, we define four domain pairs with varying network setups to assess the model adaptability across domains in the experimental network scenario. Each pair introduces specific differences between the source and target domains, such as the maximum number of users (pair 1), user movement radius (pair 2), slice combinations (pair 3), and the number of slices (pair 4). The final column presents the domain discrepancy metrics, with higher values indicating greater domain differences.**

| | Source Domain | Target Domain | Distance Metric |
|---|---|---|---|
| Pair 1 | $\phi^*_S \in \{2.0, 1.5, 1.0, 0.5\}$, $U^{(R)}_S = 500$, $U^{(N)}_S \in \{16, 24\}$ | $\phi^*_T \in \{2.0, 1.5, 1.0, 0.5\}$, $U^{(R)}_T = 500$, $U^{(N)}_T = 36$ | 0.311 |
| Pair 2 | $\phi^*_S \in \{2.0, 1.5, 1.0, 0.5\}$, $U^{(R)}_S = 250$, $U^{(N)}_S = 24$ | $\phi^*_T \in \{2.0, 1.5, 1.0, 0.5\}$, $U^{(R)}_T = 750$, $U^{(N)}_T = 24$ | 0.746 |
| Pair 3 | $\phi^*_S \in \{2.0, 1.5, 1.0, 0.5\}$, $U^{(R)}_S = 500$, $U^{(N)}_S = 24$ | $\phi^*_T \in \{2.5, 2.0, 1.5, 1.0\}$, $U^{(R)}_T = 500$, $U^{(N)}_T = 24$ | 0.383 |
| Pair 4 | $\phi^*_S \in \{2.0, 1.0, 0.5\}$, $U^{(R)}_S = 500$, $U^{(N)}_S = 24$ | $\phi^*_T \in \{2.0, 1.5, 1.0, 0.5\}$, $U^{(R)}_T = 500$, $U^{(N)}_T = 24$ | 0.958 |

online fine-tuning capabilities. On the other hand, the **DRL** scheme, which requires exploration and retraining after each network change, exhibits larger fluctuations in performance post-change, eventually converging after about 600 steps. While having initial knowledge of traffic demands, the **Traffic**-aware baseline shows varying adaptability after configuration changes. Across the slices, the **IDLA-VIB** consistently outperforms the other schemes in maintaining higher throughput levels, particularly under dynamic conditions. This highlights the robustness and efficiency of the VIB-based estimator in optimizing resource allocation for different slice requirements under changing network conditions. Overall, the **IDLA-VIB** scheme demonstrates superior performance over the other allocation strategies, successfully managing and optimizing resource allocation in a dynamic slicing environment.

In Fig. 6, we compare the empirical complementary cumulative distribution function (CDF) of the network utility (5) across different resource allocation schemes during phase $\mathcal{H}_2$ and $\mathcal{H}_3$, which represent target domains in this context. The defined utility (5) is calculated by taking the logistic of the minimum satisfaction level between slice throughput and delay, with a maximum possible utility of 2.773. The results highlight the superior performance of the **IDLA-VIB** scheme in both phases. In phase $\mathcal{H}_2$, the **IDLA-VIB** scheme achieves a 95% utility level for 54.72% of the time, compared to 38.91% for the **IDLA** scheme and 34.19% for the **DRL** scheme. Similarly, in phase $\mathcal{H}_3$, the **IDLA-VIB** continues to perform the best, with 42.56% of the time spent at the highest utility level, outperforming both the **DRL** scheme at 31.24% and the **IDLA** scheme at 11.05%. These results emphasize the robustness and adaptability of the **IDLA-VIB** approach, which consistently achieves the highest utility levels under dynamic slicing configurations. The high transferability of the VIB-based models further enhances the efficiency and scalability of the IDLA algorithm in optimizing resource allocation across diverse network conditions. This comparison underscores the overall effectiveness of IDLA combined with VIB models in ensuring optimized performance in target domains with varying slice demands.

## *D. PERFORMANCE COMPARISON OF ESTIMATORS*

To assess the models' adaptability to samples across domains, we set up 4 domain pairs with variant network living setups under the experimental network scenario. Specifically, our defined categories of domain setups are summarized as follows:

- **Pair 1: Different maximum number of users.** In the source domain, the maximum number of users ranges between $U^{(N)}_S \in \{16, 24\}$, whereas in the target domain, this expands to a larger group with $U^{(N)}_T = 36$;
- **Pair 2: Different user moving radius.** In the source domain, users move within a radius of $U^{(R)}_S = 250$ meter, compared to the target domain, where the moving radius increases to $U^{(R)}_T = 750$ meter, indicating a significantly wider range of movement;
- **Pair 3: Different slice combinations**. The source domain features slice types with throughput requirements of $\phi^*_S \in \{2.0, 1.5, 1.0, 0.5\}$ Mbit/s. In contrast, the target domain requires higher throughputs for each slice, with values at $\phi^*_T \in \{2.5, 2.0, 1.5, 1.0\}$ Mbit/s, making an increase of 0.5 Mbit/s per slice.
- **Pair 4: Different number of slices**. The source domain includes 4 slice types with throughput requirements of $\phi^*_S \in \{2.0, 1.0, 0.5\}$ Mbit/s. The target domain expands this set by introducing an additional slice, making the slice combination $\phi^*_T \in \{2.0, 1.5, 1.0, 0.5\}$ Mbit/s.

Table 2 specifies these domain configurations pairwisely. In the final column of Table 2, we provide the domain discrepancy measures derived with (38), higher metric values indicate greater domain differences, notably in pairs 2 and 4, attributed to varying slice types despite similar user counts and CQI distributions.

To demonstrate the domain difference, Fig. 3 illustrates a comparison of sample distributions of four key parameters - actual load, number of user equipments (UEs), and CQI - between the source and target domains for each evaluated domain pair. The distribution comparisons highlight distinct differences, such as the number of users in pair 1 and CQI in pair 2, leading to variations in per-slice resource occupations as depicted by the actual load. However, the differences between source and target domains are less pronounced for certain domain pairs due to their shared slice configurations. As illustrated in Table 2, domain pair 4 exhibits the largest domain distance measure, which is attributed to adding a new slice in the target domain.

### 1) ABILITY OF DOMAIN GENERALIZATION

To investigate the domain generalization (DG) ability of models, we first trained these estimators exclusively on source domain samples following the procedures outlined in Section VI-B.1 and subsequently evaluated their accuracy on both source and target domains.

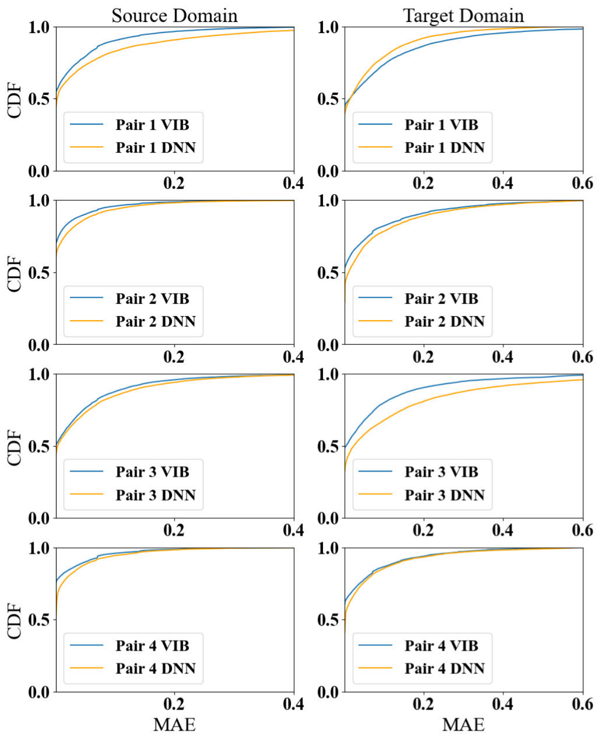

**FIGURE 7. Comparison of MAE CDF between VIB and DNN models on source and target domains.**

### 2) ABILITY OF DOMAIN ADAPTATION

In response to the diminished performance in target domains with substantial domain gaps identified in DG scenarios, we further explored the models' DA capabilities by incorporating varying proportions of limited target domain samples in the training data. Specifically, we implemented the training process of estimators under the DA scenarios, utilizing target domain sample inclusion ratio of [0%, 20%, 50%, 70%, 90%]. It is important to note that even with a 90% inclusion rate of target samples, the aggregate proportion of target domain data within the training set remains significantly lower than the portion of the source domain data, which reflects the basic assumption of DA scenarios. Table 3 demonstrates the average estimation error derived from 6 independent implements in MAE of each model, i.e., VIB, DNN, and LDS, in comparison under different usage ratios of target domain samples under all domain pairs.

As Table 3 illustrates, VIB models consistently outperform DNN estimators across all domain pairs, demonstrating notable reductions in estimation errors as the target sample ratio increases. Particularly in pairs 2 and 4, where the VIB model shows exceptional adaptability, significantly reducing the performance gap between the source and target domains. In contrast, the DNN model exhibited less flexibility and struggled to adapt to varying domain conditions.

When integrated with the LDS method, the DNN model showed improved performance, particularly in managing data imbalance, indicating LDS's effectiveness in adaptability. However, even with the enhancement provided by LDS, conventional models still exhibit lower estimation accuracy compared to VIB. Specifically, in pair 1, which has the smallest domain discrepancy, the VIB model offers improvements ranging from 24.01% to 41.62% under varying inclusion rates of target samples. This is compared to the 10.74% to 15.96% improvement range provided by the LDS model. This improvement becomes even more evident as domain discrepancy increases, as seen in domain pairs 2 and 4. The last column of each sub-table showcases the baseline estimation accuracy achieved by applying models exclusively on target domain samples. The estimation errors of both VIB and DNN models suggest that including target domain samples may slightly impact their performance on source domains. On the contrary, with the LDS method, the models' accuracy in the source domain is not adversely affected by including fewer samples. It even achieves higher accuracy in source domains with the reweighted loss. In Fig. 8, we visualize the models' estimation errors in target domains across various target sample ratios, where shadowed solid lines depict the baselines for each model. The comparison focuses on the impact of incorporating varying ratios of target domain samples into the training process under the DA scenarios.

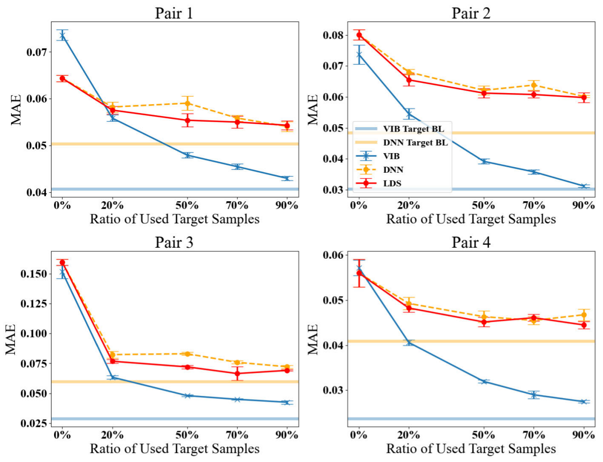

**FIGURE 8. Comparison of MAE on target domain across four domain pairs for VIB, DNN and LDS models by involving variant ratio of target samples.**

For domain pairs 1 and 3 with smaller domain discrepancies, the VIB model shows notable improvements in target domain accuracy with increased target sample inclusion. In pair 1, the VIB model reduces the MAE from $7.36 \times 10^{-2}$ to $4.3 \times 10^{-2}$, stabilizing around the 70% inclusion mark. In pair 3, the VIB model demonstrates even more substantial improvements, lowering MAE from $15.15 \times 10^{-2}$ to $4.26 \times 10^{-2}$. In contrast, the DNN model shows slight improvements in pair 1, where MAE drops from $6.44 \times 10^{-2}$ to $5.04 \times 10^{-2}$, and in pair 3, where it decreases from 15.96 to

**TABLE 3.** **Estimation error in domain pair 4 (in 0.01).**

(a): Estimation Error in Domain Pair 1 (in 0.01)

| Model | Domain | Target Sample Ratio (in %) | | | | | Target Only |
|---|---|---|---|---|---|---|---|
| | | 0 | 20 | 50 | 70 | 90 | |
| VIB | $\mathcal{D}_S$ | 3.17 | 3.26 | 3.29 | 3.36 | 3.47 | — |
| | $\mathcal{D}_T$ | 7.36 | **5.59** | **4.79** | **4.55** | **4.30** | 4.07 |
| DNN | $\mathcal{D}_S$ | 4.73 | 4.81 | 4.85 | 4.88 | 5.37 | — |
| | $\mathcal{D}_T$ | **6.44** | 5.82 | 5.90 | 5.59 | 5.41 | 5.04 |
| LDS | $\mathcal{D}_S$ | — | 5.34 | 5.27 | 5.26 | 5.17 | — |
| | $\mathcal{D}_T$ | — | 5.76 | 5.54 | 5.50 | 5.43 | — |

(b): Estimation Error in Domain Pair 2 (in 0.01)

| Model | Domain | Target Sample Ratio (in %) | | | | | Target Only |
|---|---|---|---|---|---|---|---|
| | | 0 | 20 | 50 | 70 | 90 | |
| VIB | $\mathcal{D}_S$ | 1.31 | 1.54 | 1.61 | 1.68 | 1.64 | — |
| | $\mathcal{D}_T$ | **7.37** | **5.45** | **3.92** | **3.58** | **3.12** | **3.02** |
| DNN | $\mathcal{D}_S$ | 2.63 | 2.54 | 2.88 | 2.83 | 2.89 | — |
| | $\mathcal{D}_T$ | 8.01 | 6.80 | 6.21 | 6.38 | 6.02 | 4.85 |
| LDS | $\mathcal{D}_S$ | — | 3.29 | 3.05 | 3.06 | 3.02 | — |
| | $\mathcal{D}_T$ | — | 6.55 | 6.12 | 6.08 | 5.98 | — |

(c): Estimation Error in Domain Pair 3 (in 0.01)

| Model | Domain | Target Sample Ratio (in %) | | | | | Target Only |
|---|---|---|---|---|---|---|---|
| | | 0 | 20 | 50 | 70 | 90 | |
| VIB | $\mathcal{D}_S$ | 3.77 | 3.87 | 3.99 | 4.06 | 4.14 | — |
| | $\mathcal{D}_T$ | **15.15** | **6.35** | **4.81** | **4.50** | **4.26** | **2.89** |
| DNN | $\mathcal{D}_S$ | 4.56 | 5.18 | 5.22 | 5.96 | 5.40 | — |
| | $\mathcal{D}_T$ | 15.96 | 8.24 | 8.31 | 7.59 | 7.22 | 5.98 |
| LDS | $\mathcal{D}_S$ | — | 5.98 | 5.82 | 6.06 | 5.68 | — |
| | $\mathcal{D}_T$ | — | 7.68 | 7.21 | 6.65 | 6.92 | — |

(d): Estimation Error in Domain Pair 4 (in 0.01)

| Model | Domain | Target Sample Ratio (in %) | | | | | Target Only |
|---|---|---|---|---|---|---|---|
| | | 0 | 20 | 50 | 70 | 90 | |
| VIB | $\mathcal{D}_S$ | 1.29 | 1.44 | 1.61 | 1.61 | 1.68 | — |
| | $\mathcal{D}_T$ | 5.72 | **4.06** | **3.19** | **2.90** | **2.75** | **2.37** |
| DNN | $\mathcal{D}_S$ | 2.41 | 2.63 | 2.74 | 2.73 | 2.80 | — |
| | $\mathcal{D}_T$ | **5.60** | 4.92 | 4.63 | 4.55 | 4.67 | 4.09 |
| LDS | $\mathcal{D}_S$ | — | 2.94 | 2.83 | 2.78 | 2.80 | — |
| | $\mathcal{D}_T$ | — | 4.82 | 4.51 | 4.61 | 4.45 | — |

7.57. The LDS-enhanced DNN model improves performance as well but does not achieve the same low MAE as the VIB model, particularly in domain pairs with larger gaps. In domain pairs 2 and 4, where domain gaps are larger, the VIB model exhibits even more significant reductions in MAE, improving from 7.37 $\times 10^{-2}$ to 3.12 $\times 10^{-2}$ in pair 2, and from 5.72 $\times 10^{-2}$ to 2.75 $\times 10^{-2}$ in pair 4. These results suggest the VIB model's robust adaptability in response to increased proportions of target domain samples. In contrast, the DNN model shows more restrained improvements in pair 2 and 4. And the LDS method, while improving DNN performance in managing data imbalance, still falls short of the VIB model's performance, with MAE reductions. In addition, the VIB model demonstrates superior robustness in maintaining high accuracy in the source domains, even as target domain samples are incorporated. For instance, in pair 1, the VIB model consistently maintains an MAE range between 3.17 $\times 10^{-2}$ and 3.47 $\times 10^{-2}$. We can observe similar patterns in other domain pairs, where the DNN and LDS models exhibit larger fluctuations. This ability to balance source domain accuracy while improving target domain performance underscores the robustness and reliability of the VIB model.

Overall, incorporating target domain samples into the training significantly improved the VIB model's accuracy in the target domain, with a little compromising of source domain performance in the cases of small domain discrepancies, while the DNN model does not exhibit clear enhancements from the inclusion of target domain samples. The VIB model consistently demonstrates superior generalization to the target domain across all pairs under both DG and DA contexts, particularly with higher ratios of target samples. The numerical results of estimation performance underscore VIB model's superior adaptability and robustness relative to the DNN model.

It is worth mentioning that, although we could not implement our method in real network environments, we assessed the effectiveness of the deep learning-based QoS estimator using real network data in our previous work [1]. In that study, we evaluated the deep learning-based QoS estimator not only on simulation data but also on a dataset collected from a commercial LTE network. As shown in [1], the results demonstrate the estimator's high effectiveness, with a strong correlation between predicted and actual performance metrics. The QoS estimator provides accurate predictions of network quality based on the allocated slice resources. The average MAE was 0.0639 for the simulation dataset and 0.0573 for the real dataset. These results underscore the reliability of our framework's core components, particularly the QoS estimation, which has numerically verified to be both robust and adaptable to real-world network conditions.

### *E. KEY TAKEAWAYS*

We summarize the following key takeaways based on the discussions of numerical results above:

- **Scalable Resource Allocation:** Implementing the IDLA algorithm in real-time network slicing scenarios revealed its high scalability in resource allocation. When compared to other approaches, such as a **DRL** approach and a traffic-aware allocation mechanism, the IDLA algorithm, especially when combined with VIB-based estimators, provides significant improvements in managing slice resources dynamically and efficiently with 2 times faster convergence.
- **Dynamic Adaptability:** The evaluation within dynamically changing network environments highlighted the adaptability of **IDLA-VIB** scheme. Despite the challenges posed by varying network conditions and slicing configurations, the VIB-based approach consistently outperformed other models in achieving higher throughput levels across different slices with the highest ratios of utility satisfaction, thereby illustrating its robustness in optimizing slice resource allocation under dynamic conditions.
- **Transferability:** Deploying the VIB-based model in dynamically changing network settings further emphasized its transferability. The model provided higher

adaptability over the original IDLA approach without TL. This underscores the flexibility of VIB, which optimizes resource allocation under varying slice configurations.

- **Domain Adaptation Performance:** The VIB-aided slice-wise network QoS estimators demonstrated higher DA capabilities in comparison with traditional DNN-based models. This is evident under varying TL tasks, where the VIB-based models exhibited lower MAE values across both source and target domains, signaling enhanced estimation accuracy and robust DA capacity.

## VII. CONCLUSION

In this paper, we introduced a novel framework that integrates the robust generalization capabilities of the Lagrangian method with the approximation potential of deep learning. Our proposed IDLA algorithm specifically addresses the resource allocation challenges inherent in network slicing, considering inter-slice resource constraints. The empirical results from a system-level network simulator indicate the efficacy of our proposed methods, demonstrating higher transferability, faster convergence, and better scalability compared to state-of-the-art solutions. More importantly, we applied the TL methods to IDLA by integrating the DA technique with a VIB-based slice QoS estimation model to further expand the transferability of IDLA algorithm. We investigated the DA capability of the proposed VIB models under four different TL scenarios in the context of network slicing, showcasing remarkable accuracy in QoS estimations across both source and target domains. This was particularly evident through the comparison of MAE estimation errors under different DA setups in comparison with traditional MLP-based model and a sample reweighting baseline, highlighting its enhanced estimation precision and robust DA capabilities.

In conclusion, our proposed method provides a highly scalable and transferable solution for network slicing. By ensuring near-optimal performance, fast convergence, and high generality, the IDLA algorithm with the VIB-based DA model sets a new pipeline for slicing resource allocation, paving the way for its scalable and generalized application in real-world network systems. Future research directions include expanding this work into a general domain adaptation and transfer learning framework for real-time dynamic network optimization beyond the scope of the slicing resource allocation problem. This involves integrating safe and efficient multi-agent deep reinforcement learning to enhance adaptability and exploring cross-domain transferability to broaden its applicability across diverse industries such as internet of things (IoT) and edge computing. These developments will further improve the robustness, scalability, and generalizability of the proposed solution.

## ACKNOWLEDGMENT

An earlier version of this paper was presented at the IEEE Global Communications Conference 2023 [DOI: 10.1109/GLOBECOM54140.2023.10436849].

## REFERENCES

[1] T. Hu, Q. Liao, Q. Liu, A. Massaro, and G. Carle, "Fast and scalable network slicing by integrating deep learning with Lagrangian methods," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2023, pp. 6346–6351.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[3] X. Li et al., "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 20–27, Sep. 2017.

[4] M. Leconte, G. S. Paschos, P. Mertikopoulos, and U. C. Kozat, "A resource allocation framework for network slicing," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 2177–2185.

[5] M. K. Motalleb, V. Shah-Mansouri, S. Parsaeefard, and O. L. A. López, "Resource allocation in an open RAN system using network slicing," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 1, pp. 471–485, Mar. 2023.

[6] Q. Liu, N. Choi, and T. Han, "Constraint-aware deep reinforcement learning for end-to-end resource orchestration in mobile networks," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–11.

[7] Q. Liu, N. Choi, and T. Han, "OnSlicing: Online end-to-end network slicing with reinforcement learning," in *Proc. ACM CoNEXT*, 2021, pp. 141–153.

[8] H. Zhou, M. Elsayed, and M. Erol-Kantarci, "RAN resource slicing in 5G using multi-agent correlated Q-learning," in *Proc. IEEE 32nd Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2021, pp. 1179–1184.

[9] T. Hu, Q. Liao, Q. Liu, D. Wellington, and G. Carle, "Inter-cell slicing resource partitioning via coordinated multi-agent deep reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2022, pp. 3202–3207.

[10] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[11] C. T. Nguyen et al., "Transfer learning for future wireless networks: A comprehensive survey," 2021, *arXiv:2102.07572*.

[12] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: Recent advances and future challenges," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 790–807, Jun. 2021.

[13] C. Parera, Q. Liao, I. Malanchini, C. Tatino, A. E. C. Redondi, and M. Cesana, "Transfer learning for tilt-dependent radio map prediction," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 829–843, Jun. 2020.

[14] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Transfer learning for mixed-integer resource allocation problems in wireless networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6. [Online]. Available: https://api.semanticscholar.org/CorpusID:70033937

[15] Q. Zeng, Q. Sun, G. Chen, H. Duan, C. Li, and G. Song, "Traffic prediction of wireless cellular networks based on deep transfer learning and cross-domain data," *IEEE Access*, vol. 8, pp. 172387–172397, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:222095907

[16] N. Van Huynh and G. Y. Li, "Transfer learning for signal detection in wireless networks," *IEEE Wireless Commun. Lett.*, vol. 11, no. 11, pp. 2325–2329, Nov. 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:251879154

[17] W. M. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning," 2018, *arXiv:1812.11806*.

[18] M. Akrout, A. Feriani, F. Bellili, A. Mezghani, and E. Hossain, "Domain generalization in machine learning models for wireless communications: Concepts, state-of-the-art, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 4, pp. 3014–3037, 4th Quart., 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257505534

[19] J. Shi, M. Sha, and X. Peng, "Adapting wireless mesh network configuration from simulation to reality via deep learning based domain adaptation," in *Proc. Symp. Networked Syst. Design Implement.*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:232127966

[20] Q. Zhou, R. Zhang, Z. Jing, and X. Jing, "Semi-supervised-based automatic modulation classification with domain adaptation for wireless IoT spectrum monitoring," *Frontiers Phys.*, vol. 11, Mar. 2023, Art. no. 1158577. [Online]. Available: https://api.semanticscholar.org/CorpusID:257838140

[21] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," 2000, *arXiv:physics/0004057*.

[22] Z. Goldfeld and Y. Polyanskiy, "The information bottleneck problem and its applications in machine learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 19–38, May 2020.

[23] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," 2016, *arXiv:1612.00410*.

[24] Y. Du et al., "Learning to learn with variational information bottleneck for domain generalization," in *Proc. ECCV*, 2020, pp. 200–216.

[25] Y. Yang, K. Zha, Y.-C. Chen, H. Wang, and D. Katabi, "Delving into deep imbalanced regression," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11842–11851.

[26] Y. Yang, K. Zha, Y.-C. Chen, H. Wang, and D. Katabi, "Delving into deep imbalanced regression," 2021, *arXiv:2102.09554*.

[27] F. Wei, G. Feng, Y. Sun, Y. Wang, S. Qin, and Y.-C. Liang, "Network slice reconfiguration by exploiting deep reinforcement learning with large action space," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2197–2211, Dec. 2020.

[28] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. NIPS*, 2016, pp. 1–9.

[29] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1311–1324, Jun. 2020.

[30] T. Ma, Y. Zhang, F. Wang, D. Wang, and D. Guo, "Slicing resource allocation for eMBB and URLLC in 5G RAN," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–11, Jan. 2020.

[31] A. Gholami, N. Torkzaban, and J. S. Baras, "Mobile network slicing under demand uncertainty: A stochastic programming approach," in *Proc. IEEE 9th Int. Conf. Netw. Softwarization (NetSoft)*, Jun. 2023, pp. 272–276. [Online]. Available: https://api.semanticscholar.org/CorpusID:258418083

[32] R. L. G. Cavalcante, Q. Liao, and S. Stańczak, "Connections between spectral properties of asymptotic mappings and solutions to wireless network problems," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2747–2760, May 2019.

[33] V. Sciancalepore, I. Filippini, V. Mancuso, A. Capone, and A. Banchs, "A multi-traffic inter-cell interference coordination scheme in dense cellular networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2361–2375, Oct. 2018.

[34] J. Guo, G. Zhu, D. Zhang, and C. Xu, "Resource management algorithm for slicing function in 5G network slicing," in *Proc. 5th Int. Conf. Natural Lang. Process. (ICNLP)*, vol. 38, Mar. 2023, pp. 367–372. [Online]. Available: https://api.semanticscholar.org/CorpusID:261563624

[35] Y. Abiko, T. Saito, D. Ikeda, K. Ohta, T. Mizuno, and H. Mineno, "Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 68183–68198, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:216042437

[36] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *Proc. IEEE 28th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2020, pp. 1–6.

[37] Q. Liu, T. Han, N. Zhang, and Y. Wang, "DeepSlicing: Deep reinforcement learning assisted resource allocation for network slicing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[38] Z. Yu and F. Gu, "Multi-agent reinforcement learning based 5G bi-level multi-slice resource allocation," in *Proc. 18th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2022, pp. 49–53. [Online]. Available: https://api.semanticscholar.org/CorpusID:258010998

[39] Y. Liu, J. Ding, Z.-L. Zhang, and X. Liu, "CLARA: A constrained reinforcement learning based resource allocation framework for network slicing," 2021, *arXiv:2111.08397*.

[40] H. Yang, J. Jee, G. Kwon, and H. Park, "Deep transfer learning-based adaptive beamforming for realistic communication channels," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2020, pp. 1373–1376. [Online]. Available: https://api.semanticscholar.org/CorpusID:229374594

[41] R. Jaiswal, M. Elnourani, S. Deshmukh, and B. Beferull-Lozano, "Deep transfer learning based radio map estimation for indoor wireless communications," in *Proc. IEEE 23rd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2022, pp. 1–5. [Online]. Available: https://api.semanticscholar.org/CorpusID:251169549

[42] S. B. Janiar and P. Wang, "A transfer learning approach based on integrated feature extractor for anti-jamming in wireless networks," in *Proc. IEEE 34th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, vol. 13, Sep. 2023, pp. 1–6. [Online]. Available: https://api.semanticscholar.org/CorpusID:264883315

[43] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, no. 1, pp. 151–175, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:8577357

[44] J. Wang et al., "Generalizing to unseen domains: A survey on domain generalization," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8052–8072, Aug. 2023.

[45] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," 2021, *arXiv:2103.02503*.

[46] K. Vandikas, F. Moradi, H. Larsson, and A. Johnsson, "Transfer learning and domain adaptation in telecommunications," in *Transfer Learning—Leveraging the Capability of Pre-Trained Models Across Different Domains*, D. A. P. A. Majeed, Ed., Rijeka, Croatia: InTech, 2024, ch. 12. [Online]. Available: https://doi.org/10.5772/intechopen.114932

[47] D. Guan, J. Huang, A. Xiao, S. Lu, and Y. Cao, "Uncertainty-aware unsupervised domain adaptation in object detection," *IEEE Trans. Multimedia*, vol. 24, pp. 2502–2514, 2022.

[48] J. Chen, E. Gan, K. Rong, S. Suri, and P. Bailis, "CrossTrainer: Practical domain adaptation with loss reweighting," in *Proc. 3rd Int. Workshop Data Manage. End End Mach. Learn.*, 2019, pp. 1–10.

[49] J. Raghuram et al., "Few-shot domain adaptation for end-to-end communication," 2023, *arXiv:2108.00874*.

[50] J. Shi, A. Ma, X. Cheng, M. Sha, and P. Xi, "Adapting wireless network configuration from simulation to reality via deep learning-based domain adaptation," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 1983–1998, Jun. 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:265433693

[51] A. Goyal et al., "InfoBot: Transfer and exploration via the information bottleneck," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–25.

[52] J. Chen et al., "Beyond mutual information: Generative adversarial network for domain adaptation using information bottleneck constraint," *IEEE Trans. Med. Imag.*, vol. 41, no. 3, pp. 595–607, Mar. 2022.

[53] Z. Fu, S. H. H. Ding, F. Alaca, B. C. M. Fung, and P. Charland, "Plvio: Assembly clone search for out-of-domain architectures and libraries through transfer learning and conditional variational information bottleneck," 2023, *arXiv:2307.10631*.

[54] Y. Song et al., "Improving unsupervised domain adaptation with variational information bottleneck," in *Proc. ECAI*, 2020, pp. 1499–1506.

[55] S. Si et al., "Variational information bottleneck for effective low-resource audio classification," in *Proc. Interspeech*, Aug. 2021, pp. 591–595. [Online]. Available: https://openreview.net/forum?id=kvhzKz-_DMF

[56] G. Tychogiorgos and K. K. Leung, "Optimization-based resource allocation in communication networks," *Comput. Netw.*, vol. 66, pp. 32–45, Jun. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128614001170

[57] *Telecommunication Management; Performance Management (PM); Performance Measurements—Definitions*, document TS 32.404 V17.1.0, Release 17, 3GPP, Jun. 2023. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1994

[58] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[59] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. NIPS Autodiff Workshop*, 2017, pp. 1–4.

[60] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Conv. Rec.*, vol. 7, pp. 325–350, Mar. 1959.

[61] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2022, *arXiv:1312.6114*.

[62] Nokia Siemens Networks, "Introducing the Nokia Siemens networks SON suite—An efficient, future-proof platform for SON," Self-Organizing Netw., Espoo, Finland, White Paper, Oct. 2009.

[63] T. Hu, Q. Liao, Q. Liu, and G. Carle, "Network slicing via transfer learning aided distributed deep reinforcement learning," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2022, pp. 2909–2914.

[64] J. M. Joyce, *Kullback–Leibler Divergence*. Berlin, Germany: Springer, 2011, pp. 720–722, doi: 10.1007/978-3-642-04898-2_327.