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

Received 28 February 2024; revised 11 July 2024; accepted 23 August 2024.
Date of publication 29 August 2024; date of current version 6 September 2024.

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

Digital Object Identifier 10.1109/TMLCN.2024.3452057

Node Cardinality Estimation in the Internet of Things Using Privileged Feature Distillation

PRANAV S. PAGE[1,2], ANAND S. SIYOTE[3], VIVEK S. BORKAR[2] (Life Fellow, IEEE), AND GAURAV S. KASBEKAR[2] (Member, IEEE)

[1]Carnot Technologies, Mumbai 400059, India
[2]Department of Electrical Engineering, IIT Bombay, Mumbai 400076, India
[3]TIH Foundation for IoT and IoE, Indian Institute of Technology (IIT) Bombay, Mumbai 400076, India
CORRESPONDING AUTHOR: G. S. KASBEKAR (gskasbekar@ee.iitb.ac.in)

The work of Vivek S. Borkar and Gaurav S. Kasbekar was supported by Grant RD/0222-EETIHBY-014.

ABSTRACT The Internet of Things (IoT) is emerging as a critical technology to connect resource-constrained devices such as sensors and actuators as well as appliances to the Internet. In this paper, a novel methodology for node cardinality estimation in wireless networks such as the IoT and Radio-Frequency Identification (RFID) systems is proposed, which uses the Privileged Feature Distillation (PFD) technique and works using a neural network with a teacher-student model. This paper is the first to use the powerful PFD technique for node cardinality estimation in wireless networks. The teacher is trained using both privileged and regular features, and the student is trained with predictions from the teacher and regular features. Node cardinality estimation algorithms based on the PFD technique are proposed for homogeneous wireless networks as well as heterogeneous wireless networks with $\boldsymbol{T} \geq \mathbf{2}$ types of nodes. Extensive simulations, using a synthetic dataset as well as a real dataset, are used to show that the proposed PFD based algorithms for homogeneous as well as heterogeneous networks achieve much lower mean squared errors (MSEs) in the computed node cardinality estimates than state-of-the-art protocols proposed in prior work. In particular, our simulation results for the real dataset show that our proposed PFD based technique for homogeneous (respectively, heterogeneous) networks achieves a MSE that is $\mathbf{92.35\%}$ (respectively, $\mathbf{94.08\%}$) lower on average than that achieved by the Simple RFID Counting (SRC_s) protocol (respectively, T-SRC_s protocol) proposed in prior work while taking the same number of time slots to execute.

INDEX TERMS Medium access control protocols, Internet of Things, node cardinality estimation, privileged feature distillation, neural network.

I. INTRODUCTION

THE Internet of Things (IoT) is emerging as a critical technology to connect a large number of resource-constrained devices such as sensors and actuators as well as appliances to the Internet [1]. Many industries, including smart grids, healthcare, vehicular telematics, smart cities, security and public safety, agriculture, and industrial automation, extensively use IoT networks [2]. Active research is being conducted on designing effective networking protocols to handle the growing number of IoT devices. The design of Medium Access Control (MAC) protocols for IoT networks is particularly challenging because of their unique characteristics [3]. For instance, (i) network access must be provided to a large number of IoT devices, (ii) most IoT devices are battery-powered and have limited power availability, and (iii) the Quality of Service (QoS) requirements in IoT applications differ from those in human-to-human (H2H) communications [3]. A key component of a MAC protocol for IoT networks is a node cardinality estimation protocol that rapidly estimates the number of active devices (i.e., the devices that currently have some data that needs to be transferred to the base station (BS)) in every time frame [3]. This is because these estimates can be used to determine the optimal values of various MAC protocol parameters such as contention probability, contention period duration, data transmission period duration, etc. [4], [5], [6].

Node cardinality estimation protocols also have a large number of applications apart from their use in MAC protocol design, as will now be explained. They are used in [7] to periodically estimate the numbers of vehicles moving on various congested routes; the estimated information can be used to dynamically adapt the ON/ OFF periods of traffic lights based on vehicle density. Consider a farm with sensors installed to track a number of variables such as temperature and soil moisture. Before gathering the actual data from the active sensors, a Mobile BS (MBS), e.g., one mounted on an Unmanned Aerial Vehicle (UAV), navigates over the agricultural area and stops at designated locations to estimate the number of active sensors [8]. This increases the effectiveness of data collection because the MBS can optimally determine the amount of time it needs to spend at each stop when it subsequently returns to the same locations to collect the actual data and because it can inform the active sensors when to be available for sending the data based on the estimates. During or after natural calamities such as floods and earthquakes, MBSs hover above the affected area to estimate the number of people who need help. These estimates are used to plan disaster relief efforts and efficiently distribute supplies [9]. Also, numerous Radio-Frequency Identification (RFID) systems use node cardinality estimation protocols for inventory management, tag identification, missing tag detection, etc. [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. Hence, the focus of this paper is on designing effective schemes for node cardinality estimation.

Extensive research has been conducted on the problem of node cardinality estimation in IoT networks and RFID systems. Most of this research is focused on node cardinality estimation in a *homogeneous* network, wherein the network consists of only one type of nodes [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41]. In addition, some work has been carried out on node cardinality estimation in a *heterogeneous* network, i.e., a network consisting of T types of nodes, where $T \geq 2$ is an integer [2], [4], [5], [42], [43], [44], [45], [46], [47], [48]. Different types of nodes in a heterogeneous network may have different hardware and software capabilities such as different processor speeds, transmission power, memory, battery life, etc.; also, different types of nodes may have different QoS requirements [43]. In a heterogeneous network with T types of nodes, execution of an estimation protocol designed for a homogeneous network T times to obtain separate estimates of the cardinality of each node type is inefficient and improved protocols that rapidly obtain these estimates have been proposed in [2], [4], [5], [42], [43], [44], [45], [46], [47], and [48]. However, all the protocols designed so far for node cardinality estimation in homogeneous and heterogeneous wireless networks use simple techniques, which are sub-optimal, for computing the node cardinality estimates. Very little research has been conducted so far on applying machine learning based techniques to compute node cardinality estimates; in particular, to the best of our knowledge, the powerful *Privileged Feature Distillation* (PFD) technique [49] has not been used in prior work for node cardinality estimation in wireless networks. A contribution in this space is made in this paper.

In this paper, a novel methodology for node cardinality estimation in wireless networks such as the IoT and RFID systems is proposed, which uses the PFD technique and works using a neural network with a *teacher-student model* [49]. The teacher is trained using both privileged and regular features, and the student is trained using predictions from the teacher and regular features [49].

The concept of a privileged feature [49] arises in scenarios where a particular feature z is available during the training phase but not during the testing or inference phase. The term ''privileged'' refers to the notion that this feature possesses additional information during the training process that can potentially aid in improving the prediction accuracy or performance. By identifying privileged features and incorporating them into the training process, it is possible to leverage the additional information they provide to potentially improve the model's predictive capabilities, when those features are not available during the testing phase. *Distillation* [50] refers to the standard practice of labeling the training dataset using teacher predictions, and using these as supervision targets in the training of the student model. PFD has been successfully applied in various machine learning problems including speech recognition [51], medical imaging [52], image super-resolution [53], and IoT traffic classification [54]. Some background concepts pertaining to PFD are reviewed in Section IV-C.

Another motivation for applying the PFD technique to the problem of node cardinality estimation is that the technique is well-suited to this problem since there are several features in the problem that are natural candidates for being used as privileged features. For example, in a practical network, when a collision occurs in a time slot, the BS does not know as to how many nodes transmitted in the slot. So the number of transmitting nodes in a slot in which a collision takes place can serve as a privileged feature to train the teacher network. Also, in a practical network, the BS does not know the true number of active nodes in the previous time frame; it only knows the estimated number of active nodes. So the true value of the number of active nodes in the previous time frame can serve as another privileged feature. The benefit of using PFD for node cardinality estimation is that the use of privileged information during the training phase can significantly improve the performance of the estimation algorithm relative to traditional techniques for estimation, as our results in Section VI demonstrate.

In summary, the motivation behind this work is as follows. Node cardinality estimation in wireless networks is an important problem with a large number of applications. However, so far, only simple and sub-optimal techniques have been designed to solve this problem. This motivates us to apply the powerful PFD technique, which is well-suited to

this problem as explained above, to design improved node cardinality estimation schemes.

The main contributions of this paper are as follows. The problem of estimating the number of active nodes in a homogeneous wireless network while minimizing the mean squared error (MSE) between the actual number of active nodes and the algorithm's estimate has been formulated and then generalized to estimating the number of active nodes of each type in a heterogeneous wireless network with T types of nodes, where $T \geq 2$ is an integer, while minimizing the MSE. A novel algorithm using the PFD technique and a neural network with a teacher-student model has been proposed for node cardinality estimation in a homogeneous network and then has been generalized to estimate the cardinality of each node type in a heterogeneous network with T types of nodes. This paper is the first to use the powerful PFD technique for node cardinality estimation in wireless networks. Extensive simulations, using a synthetic dataset [55] as well as a real dataset [56], show that the proposed PFD-based algorithms for homogeneous and heterogeneous networks achieve much lower MSEs than state-of-the-art protocols, despite taking the same number of time slots to execute. In particular, our simulation results for the real dataset show that our proposed PFD based technique for homogeneous (respectively, heterogeneous) networks achieves a MSE that is 92.35% (respectively, 94.08%) lower on average than that achieved by the Simple RFID Counting (SRC_s) protocol (respectively, T-SRC_s protocol) proposed in prior work while taking the same number of time slots to execute. Also, these algorithms are applicable to arbitrary wireless networks, including the IoT.

The rest of this paper is organized as follows. A review of related prior literature is provided in Section II. The system model and problem formulation are described in Section III. Some relevant background is given in Section IV. The proposed algorithms and other algorithms for comparison are described in Section V. Simulation results are provided in Section VI. Finally, conclusions and directions for future research are provided in Section VII.

II. RELATED WORK

In Section II-A (respectively, Section II-B), a review of related prior literature on node cardinality estimation in wireless networks (respectively, PFD) is provided.

A. NODE CARDINALITY ESTIMATION

The estimation of active node cardinalities is considered crucial in the design of a MAC protocol for IoT networks. This importance has led to extensive research being conducted on this topic [26], [27], [28], [29], [30], [31], [32], [33], [34], [35]. These studies focus not only on estimating the number of active devices in a homogeneous IoT network, but also on using these estimates to determine the contention probabilities that optimize the throughput of their respective MAC protocols for IoT networks. To estimate the number of active nodes in the current time frame, the estimation scheme proposed in [27] uses the estimates obtained in the previous frame as well as the sub-optimal Dynamic Access Class Barring (D-ACB) factors from the previous frame. In [28], a modified Carrier Sense Multiple Access/ Collision Avoidance (CSMA/CA) protocol intended for IoT networks was introduced. The size of the backoff window for the current time frame is chosen by this protocol by considering the size of the previous backoff window and the previously computed estimates of the active node cardinality. This procedure incorporates historical estimates to improve the effectiveness of the backoff mechanism. Note that both [27] and [28] relied on the estimates obtained in previous frames to compute their estimates in the current frame. This iterative approach allows the utilization of past information to improve the accuracy and effectiveness of the estimation process. A new technique for dynamic access control and resource allocation for random-access channels based on an estimation scheme was introduced in [29]. The only input used in the estimation procedure in [29] for computing the estimates was the number of open slots. The 6-Dimensional Markov Chain (6)-DMC) estimation method was introduced in [30]. The numbers of devices that are delay tolerant and delay sensitive are estimated using this approach. The estimation methods in [30], [31], and [32] are based on 6-DMC, Maximum Likelihood Estimation (MLE), and IoT-OSA (an extension of the opportunistic splitting technique).

The satellite Random Access (RA) MAC protocol is provided in [57]. In this protocol, throughput is maximized by computing an estimate of the number of Return Channel Satellite Terminals (RCSTs). The number of collisions observed in earlier frames affects the length of the current frame in the model described in [57]. The approach described in [33] estimates the number of nodes that cause collisions. In Long-Term Evolution (LTE) networks, this estimation enables an effective partitioning of nodes into a predetermined number of groups while minimizing intra-group collisions. Dynamic Backoff (DB), a new method for resolving channel contention, was first described in [34]. Based on the estimated number of competing active devices, this approach modifies the size of the backoff window used to manage channel contention during data transfer. The scheme proposed in [34] also dynamically modifies each frame size based on the projected number of devices, making it adaptable to shifting network circumstances and device activity.

The node cardinality estimation problem in IoT networks is similar to the tag cardinality estimation problem in the context of RFID technology. In the latter situation, an RFID reader estimates the number of tags, like a BS does when estimating the number of active nodes in an IoT network. Schemes for estimating the number of tags in an RFID system were proposed in [20], [21], [22], [23], [24], [25], [36], [37], [38], [39], [40], [41], and [58].

Node cardinality estimation schemes for heterogeneous IoT networks and RFID systems have been proposed in [2], [4], [5], [42], [43], [44], [46], [47], [48], and [45]. A specialized MAC protocol for a heterogeneous IoT network,

catering to three types of IoT devices, has been introduced in [4] and [5]. It incorporates a rapid estimation protocol to determine active node counts, and uses them to optimize the contention probabilities in the MAC protocol. An efficient node cardinality estimation solution with two components– snapshot collection and accurate estimation– has been given in [46]. It focuses on improving joint cardinality estimation in distributed RFID systems, allowing queries across multiple tag sets at different locations and times with controlled error. It has applications in tracking product flows in logistics. Simulations show a significant time cost reduction while maintaining accuracy. Enhancement of RFID technology's cardinality estimation function in two ways– joint estimation across tags at different locations and times and category-level tracking– was proposed in [47]. It introduces an anonymous protocol that efficiently estimates joint category-level information, preserving tag anonymity and enabling applications such as monitoring diverse products in distributed supply chains. Multi-category RFID tag estimation has been addressed in [48], aiming to swiftly and accurately count tags within each category. It introduces the "Simultaneous Estimation for Multi-category RFID Systems" (SEM) approach, leveraging Manchester coding to decode combined signals, allowing simultaneous estimation across categories while maintaining pre-defined accuracy. SEM significantly improves estimation speed compared to existing protocols. Rapid estimation of the cardinalities of active nodes of different types in heterogeneous IoT networks with T node types, where $T \geq 2$ is an arbitrary integer, has been addressed in [42], [43], [44], and [45].

However, the PFD technique has not been applied to the problem of node cardinality estimation in either a homogeneous or heterogeneous wireless network in any of the above prior works.

B. PFD

The concept of learning with privileged features was introduced in [59], and a framework called "Learning Using Privileged Information" (LUPI) was proposed. Privileged information is the primary approach used by LUPI to discriminate between simple and complex cases. These methods are closely related to Support Vector Machines (SVM); e.g., the "SVM+" algorithm, which creates slack variables from privileged features and learns an SVM based on regular features with those slack variables, is proposed in [59] and [60]. A pair-wise SVM algorithm for ranking that uses privileged features to differentiate between easy and hard pairs is proposed in [61]. The privileged features are employed in the version presented in [62] to produce importance weighting for various training samples.

A popular technique for knowledge transfer is model distillation [50], often from a large model to a smaller model [63], [64]. Recent studies [65], [66], [67], and even those where the teacher model and student model have the same structure [68], [69], have demonstrated remarkable empirical success in ranking tasks.

"Generalised Distillation" (GenD) is the term for the method first suggested in [70] for using distillation to learn from privileged features. This offers a comprehensive perspective on distillation and LUPI. GenD and its derivatives [51], [53], [71] train an expert model using just privileged features, after which the student model is trained to replicate the expert's predictions. Recently, PFD was presented in [72], where the teacher model accepts input from both regular and privileged features. Due to their emphasis on privileged feature exploitation rather than model size reduction, PFD and GenD are different from traditional model distillation. On a non-public data collection, the improved performance of PFD for recommendation systems is empirically demonstrated in [72].

Despite the aforementioned empirical accomplishment, there remains a lack of understanding of privileged characteristics distillation. Prior research [73] demonstrates that LUPI speeds up convergence under the strict premise that the best classifier can be realised using just privileged information. GenD has a quick convergence rate, as shown by [70]. This is different from PFD since it assumes that the function class complexity of the student model is significantly larger than that of the teacher model. The study by [74] on GenD under semi-supervised learning reveals that the benefits come from reducing the complexity of student function classes. However, it does not quantify this reduction, and the theory does not explain why exploiting privileged traits is advantageous.

Prior proposals include other uses of privileged features. To enhance picture classification performance, [75] learns a more varied representation using privileged information. Distillation strategies have been proposed by [53] and [76] for more effective feature extraction from regular features. A more recent study [77] examined the possibility of training a model using both regular and privileged features to improve the internal representation of regular features.

However, to the best of our knowledge, this paper is the first to use the technique of PFD to address the problem of node cardinality estimation in wireless networks.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In Section III-A (respectively, Section III-B), the system model and problem formulation for the case of a homogeneous network (respectively, heterogeneous network) are described.

A. HOMOGENEOUS NETWORK

1) SYSTEM MODEL

Consider a population of nodes such that each node is in the range of a single stationary BS. A node is considered as *active* when it has data to send to the BS. Time is divided into frames of equal durations. To effectively design MAC protocols for data upload to the BS, the number of active nodes in a time frame must be estimated. The case, which often arises in practice, when there exists some correlation between the number of nodes active in a time frame and the

number of nodes active in the next time frame, is studied. E.g., the number of active nodes may evolve as a Discrete-Time Markov Chain (DTMC).

The above choice of the system model, in which there is correlation between the numbers of active nodes in consecutive time frames, is influenced by examples from practical network deployments. For example, in a wireless sensor network deployed as a part of a precision agriculture system, a node equipped with a sensor may need to become active and send an alert whenever a particular sensed physical quantity (soil moisture, temperature, humidity, etc.) crosses a particular threshold. E.g., water may need to be sprayed whenever the soil moisture level goes below a threshold. These quantities are slowly varying with time (e.g., the soil moisture level slowly decreases due to evaporation), due to which the numbers of active nodes in consecutive time frames are correlated. As another example, in a wireless network, a node with data packets to send might have to wait for several time frames till it is allowed to transmit by the MAC protocol, which makes it likely that the numbers of active nodes in consecutive time frames are correlated.

2) PROBLEM FORMULATION

The aim of this work is to design algorithms that minimize the MSE between the actual number of active nodes in a time frame and the algorithm's estimate of this number, while also reducing the number of time slots needed to produce the estimate. The objective of minimizing the MSE for a homogeneous network of nodes is as follows:

$$alg' = \text{argmin}_{alg}\mathbb{E}\left[\lim_{\tau\to\infty}\frac{1}{\tau}\sum_{t=0}^{\tau-1}(\hat{n}_t^{alg} - n_t^{truth})^2\right]. \quad (1)$$

Here, n_t^{truth} is the true number of nodes active in time frame t, while $\hat{n}_t^{alg}$ is the estimate given by the algorithm alg in time frame t. The expectation is with respect to the different realizations of the random process $(n_t^{truth},\ t = 0, 1, 2, \ldots)$. The objective is to find the algorithm alg' that achieves the minimum in the RHS of (1).

B. HETEROGENEOUS NETWORK

1) SYSTEM MODEL

In the heterogeneous case, there exist T types of nodes, where $T \geq 2$ is an integer, in the range of a BS, as shown in Figure 1 for the case $T = 3$. Different types of nodes (e.g., nodes of types 1, 2, 3, etc.) in a heterogeneous wireless network may have different hardware and software capabilities, such as different processor speeds, transmission power, memory, battery life, etc. [43]. Also, different types of nodes may have different QoS requirements. E.g., one type of node may need to transmit small bursts of data periodically or randomly, another type may have stringent latency requirements or need priority access to communicate alarms (e.g., in healthcare and security applications), another type may require highly reliable communication (e.g., in remote payment systems) or high throughput (e.g., in a video surveillance application) [43]. As another example, nodes of different types may correspond to nodes that send emergency traffic such as fire alarms, nodes that contain moisture sensors, nodes that contain temperature sensors, etc. All the results in this paper hold regardless of how the set of all nodes is divided into nodes of different types. The numbers of active nodes of different types are represented by $\mathbf{n}_t^{truth}$, a $1 \times T$ vector, where $\mathbf{n}_t^{truth}[b]$, $b \in \{1, \cdots, T\}$, is the number of active nodes of type b in time frame t.

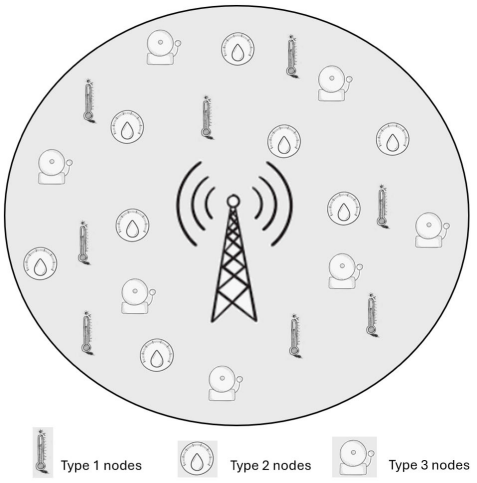


FIGURE 1. The figure shows an example heterogeneous network with $T = 3$ types of nodes in the range of a BS.

2) PROBLEM FORMULATION

Similar to the homogeneous case, for the heterogeneous case, the objective is to minimize the expected time-averaged squared Euclidean distance between the estimates computed by the algorithm and the time series, $\mathbf{n}_t^{truth}$, of true numbers of active nodes:

$$alg' = \text{argmin}_{alg}\mathbb{E}\left[\lim_{\tau\to\infty}\frac{1}{\tau}\sum_{t=0}^{\tau-1}\left\|\hat{\mathbf{n}}_t^{alg} - \mathbf{n}_t^{truth}\right\|_2^2\right]. \quad (2)$$

Here, $\hat{\mathbf{n}}_t^{alg}$ is a $1 \times T$ vector, where $\hat{\mathbf{n}}_t^{alg}[b]$, $b \in \{1, \cdots, T\}$, is the estimate of the number of active nodes of type b in time frame t found by the algorithm alg. The objective is to find the algorithm alg' that achieves the minimum in the RHS of (2).

IV. BACKGROUND

In this section, some concepts that are used in the rest of the paper are reviewed.

A. SRC$_S$ PROTOCOL

SRC$_s$ is a node cardinality estimation protocol for a homogeneous network [58], which finds an estimate, $\hat{n}$, of the number of active nodes, n, to within given accuracy requirements ϵ

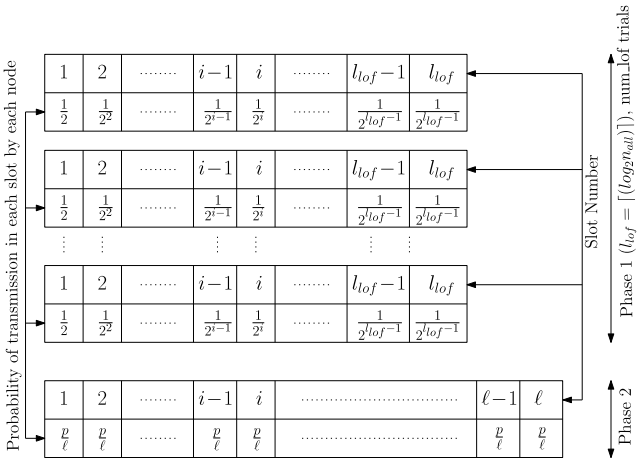


FIGURE 2. The figure shows the frame structure of the SRC$_s$ protocol.

and δ, i.e, the following relation is satisfied:

$$\mathbb{P}(|\hat{n} - n| \leq \epsilon n) \geq 1 - \delta.$$

The SRC$_s$ protocol [58] uses the Lottery Frame (LoF) protocol to generate a rough estimate of the number of active nodes, followed by a Balls and Bins (BB) trial that uses the rough estimate given by LoF. The LoF, BB, and SRC$_s$ protocols are described in Algorithm 1. The frame structure of SRC$_s$ is shown in Figure 2.

The LoF protocol uses a trial length of $l_{lof} = \lceil \log_2 n_{all} \rceil$ time slots, where n_{all} is the maximum number of active nodes in the network. Each active node independently transmits in a randomly chosen slot of the trial, and the rough estimate returned by LoF is computed as a function of the slot number, say j, of the first empty slot (slot in which no node transmits) of the trial.

A BB trial consists of l time slots, where l is chosen as explained in the next paragraph. Given a rough estimate n', each active node independently participates in the trial with probability $p = \min(1, 1.6l/n')$. Each participating node transmits in a slot chosen uniformly at random from the l slots, and the estimate returned by BB is computed as a function of the number of empty slots in the trial.

In step 1 of the SRC$_s$ protocol given in Algorithm 1, the protocol conducts multiple, say num_lof, LoF trials (see Figure 2) and computes an average of the rough estimates generated in the trials, n'. Subsequently, in step 2, the SRC$_s$ protocol conducts a BB trial of length l (see Figure 2), in which each node independently participates with probability $p = \min(1, 1.6l/n')$. The choice of the length of the BB trial, l, depends on the relative error tolerated, ϵ, and is taken as $l = \frac{65}{(1-0.04^{\epsilon})^2}$ [58]. The number of LoF trials, num_lof, in SRC$_s$ is taken to be of the order $O(\log \frac{1}{\delta})$. E.g., for $\delta = 10^{-3}$, num_lof $= 3$ is used. At the end of the BB trial, the final SRC$_s$ estimate, $\hat{n}$, is computed as a function of the number of empty slots in the trial.

Algorithm 1 The LoF, BB, and SRC$_s$ Protocols [58]

LoF Protocol
1: Choose trial length $l_{lof} = \lceil \log_2 n_{all} \rceil$
2: Each active node independently transmits in slot $i = 1, \ldots, l_{lof} - 1$ with probability 2^{-i} and in slot l_{lof} with probability $2^{-(l_{lof}-1)}$
3: Trial ends when first empty slot (slot in which no node transmits), say slot j, is seen
4: **return** $n' = 1.2897 \times 2^j$

BB Protocol
1: Given rough estimate n', each active node independently participates in the trial of length l slots with probability $p = \min(1, 1.6l/n')$
2: Each participating node transmits in a slot chosen uniformly at random from the l slots
3: z = number of empty slots in trial
4: **if** $z > 0$ **then**
5: **return** $\frac{\ln(z/l)}{\ln(1-p/l)}$
6: **else**
7: **return** arbitrary number
8: **end if**

SRC$_s$ Protocol
1: Conduct num_lof LoF trials and compute the following rough estimate: $n' = 1.2897 \times 2^{\sum_{m=1}^{\text{num\_lof}} (j(m)-1)/\text{num\_lof}}$, where $j(m)$ is the first empty slot in the m'th LoF trial
2: Run a BB trial of length l in which each node independently participates with probability $p = \min(1, 1.6l/n')$
3: Count the number of empty slots, say z, in the trial
4: **if** $z > 0$ **then**
5: **return** $\hat{n} = \frac{\ln(z/l)}{\ln(1-p/l)}$
6: **else**
7: **return** $\hat{n}$ = arbitrary number
8: **end if**

B. 3-STAGE SCHEME-BALLS AND BINS (3)-SS-BB) PROTOCOL

3-SS-BB is a protocol for finding an estimate, $\hat{n}_b$, of the number of active nodes, n_b, of each type $b \in \{1, \ldots, T\}$ in a heterogeneous network with T types of nodes (see Figure 1) [42], [43]. It is an extension of the BB trial (see Algorithm 1) to a heterogeneous network. It assumes that rough estimates, n'_b, $b \in \{1, \ldots, T\}$, of the numbers of active nodes of the T types are initially available; e.g., these estimates may be generated by separately conducting LoF trials for each node type as in the first phase of SRC$_s$. 3-SS-BB uses a trial with l blocks; within each block, there are $T - 1$ slots. Each active node of type b independently participates in the trial of l blocks with probability $p_b = \min(1, 1.6l/n'_b)$. Each participating node of type b chooses a block out of the l blocks uniformly at random and sends, in its chosen block, the symbol combination of length $T-1$ slots given in the row corresponding to type b in Figure 3. For example, each node of type 1 transmits the pattern $(\alpha, \alpha, \cdots, \alpha)$, while each node of type 2 transmits $(\beta, 0, \cdots, 0)$, where α and β are distinct symbols (bit patterns) and 0 indicates no transmission. If there are two or more transmissions in a slot, the result of the slot is c (collision). Thus, a slot can have four possible outcomes: $\{0, \alpha, \beta, c\}$. Estimates $\hat{n}_b$, $b \in \{1, \ldots, T\}$, are generated based on the outcomes of the $(T-1)l$ slots using the algorithm

Symbol Combinations

Types \ Slots	1	2	3	·	·	·	·	$T-2$	$T-1$
1	α	α	α	·	·	·	·	α	α
2	β	0	0	·	·	·	·	0	0
3	0	β	0	·	·	·	·	0	0
4	0	0	β	·	·	·	·	0	0
·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	·
$T-1$	0	0	0	·	·	·	·	β	0
T	0	0	0	·	·	·	·	0	β

FIGURE 3. The figure shows the symbol combinations used by different types of nodes under the 3-SS-BB protocol. 0 indicates no transmission.

provided in [42] and [43]. Under this algorithm, first, the sets of types of nodes that transmitted in each of the l blocks are inferred based on the outcomes of the $T-1$ slots in the respective blocks. For example, if the outcomes of the $T-1$ slots in a block are $(c, c, \alpha, \ldots, \alpha)$, then from Figure 3, it follows that nodes of types 1, 2, and 3 transmitted and nodes of the other types did not transmit in the block. If the outcomes of the $T-1$ slots in a block are $(0, \ldots, 0, \beta, c)$, then it follows that nodes of types $T-1$ and T transmitted in the block and nodes of the other types did not transmit in the block. Subsequently, for each node type b, the number of blocks, say z_b, in which no node of type b transmitted is deduced. Finally, the estimates $\hat{n}_b$, $b \in \{1, \ldots, T\}$, are computed using z_b, $b \in \{1, \ldots, T\}$, as in steps 4 to 8 of the BB protocol given in Algorithm 1 with z and p replaced with z_b and p_b, respectively. See [42] and [43] for details of the algorithm.

C. PFD

In some problem settings, there exist some features that are not available during testing, but are available offline for training. Instead of discarding these features, one approach is to train a 'teacher' model on the privileged features, say $\mathbf{x}_{privileged}$ [49]. The teacher model is then used in the training of a different 'student' model [49]. The student is trained only on the features, say $\mathbf{x}_{general}$, that are available during testing, but the loss of the student model is designed (see (5)) as a convex combination of the data loss and the teacher loss, as will now be explained. The teacher network is represented by g^{Tr}, and the teacher's prediction is $g^{Tr}(\mathbf{x}_{privileged})$. In (3), $\mathcal{L}^{Tr}$ refers to the teacher loss corresponding to the loss described by the loss function $L(\cdot, \cdot)$, operating on the prediction of the teacher and the target y. Similarly, in (4), the data loss $\mathcal{L}^{data}$ is the loss between the student's prediction $g^{Stu}(\mathbf{x}_{general})$ on the general features, $\mathbf{x}_{general}$, and the target y. In (5), the student loss, $\mathcal{L}^{Stu}$, is the convex combination of the data loss and the teacher loss; the mixing ratio between the data loss and the teacher loss is $\alpha \in [0, 1]$. The student does not interact with the privileged features, nor with the teacher's predictions, but only with the loss between the teacher's prediction and the target.

$$\mathcal{L}^{Tr} = L(g^{Tr}(\mathbf{x}_{privileged}), y) \tag{3}$$

$$\mathcal{L}^{data} = L(g^{Stu}(\mathbf{x}_{general}), y) \tag{4}$$

$$\mathcal{L}^{Stu} = \alpha\mathcal{L}^{data} + (1-\alpha)\mathcal{L}^{Tr} \tag{5}$$

V. ALGORITHMS

In Section V-A (respectively, Section V-B), the proposed PFD based algorithm, and other algorithms for comparison, for homogeneous (respectively, heterogeneous) wireless networks are described.

A. HOMOGENEOUS NETWORK

1) PROPOSED ALGORITHM

Consider the model and problem formulation described in Section III-A. The entire population of nodes in the range of the BS is of a single type. Recall from Section IV-A that the SRC$_s$ protocol consists of a LoF-based phase 1 and a BB-based phase 2. The LoF phase obtains a rough estimate of the number of nodes, n', which the BB phase uses to obtain a refined estimate.

In each time frame $t = 1, 2, 3, \ldots$, the proposed Neural Network (NN) based algorithm (see Algorithm 2) executes only phase 2 (BB) and obtains the trial result. If the trial consists of l_{BB} slots, then a vector of size l_{BB} is generated via BB. This vector consists of the outcome (no transmission, success (one transmission), or collision) in each of the l_{BB} slots of the BB trial. The trained model takes this vector as input, along with the estimate of the number of active nodes generated by the model in the previous time frame, and estimates the value of the number of active nodes in the current time frame. The NN is a student model trained using PFD as explained in Section V-A.2; so henceforth, the trained model will be represented by the notation *Stu*.

In time frame 0, the proposed NN method conducts a set of LoF trials to give the initial rough estimate $\hat{n}'_0$, which is then used as the rough estimate (see step 1 in the BB protocol given in Algorithm 1) in a BB trial. The NN then uses the result of the BB trial, which is a vector of length l_{BB}, say v_0, and the rough estimate $\hat{n}'_0$, to generate its own estimate in time frame 0, $\hat{n}_0^{Stu}$, using the student network g^{Stu}. Subsequently, in each time frame $t = 1, 2, \ldots,$ num_iters, where num_iters is the total number of time frames, a BB trial is conducted with rough estimate $\hat{n}_{t-1}^{Stu}$ (*Stu*'s estimate of the previous time frame), which generates a vector of length l_{BB}, say v_t, as a result. The NN g^{Stu} operates on $(v_t, \hat{n}_{t-1}^{Stu})$ to produce its estimate $\hat{n}_t^{Stu}$ in time frame t.

The motivation for using the estimate of the previous time frame, $\hat{n}_{t-1}^{Stu}$, as the rough estimate for the BB trial of the current time frame t arises from the fact that some correlation exists between the nodes active in the previous time frame and the nodes active in the current time frame as explained in Section III-A. This fact is exploited to reduce the number of

Algorithm 2 Proposed NN Method and its Training

Proposed NN Method

At $t = 0$, conduct LoF trials to give the initial rough estimate $\hat{n}'_0$; then conduct BB trial to generate v_0
$\hat{n}_0^{Stu} = g^{Stu}(v_0, \hat{n}'_0)$
for $t = 1, \cdots,$ num_iters **do**
  Conduct BB trial with rough estimate $\hat{n}_{t-1}^{Stu}$, giving result v_t
  $\hat{n}_t^{Stu} = g^{Stu}(v_t, \hat{n}_{t-1}^{Stu})$
end for

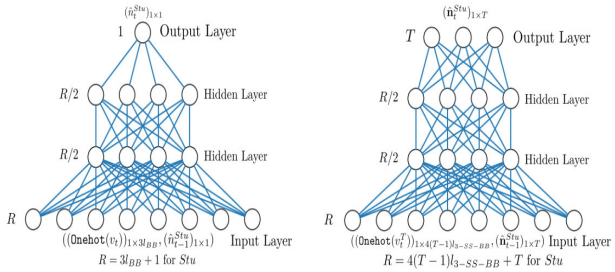
Teacher Training

function gen_teacher_training_data
  Randomly initialize gTr's weights
  At $t = 0$, conduct LoF trials to give the initial rough estimate $\hat{n}'_0$; then conduct a BB trial to generate V_0
  $\hat{n}_0^{gTr} = g^{gTr}(V_0, \hat{n}'_0)$
  for $t = 1, \ldots,$ num_iters **do**
    Run BB trial with rough estimate $n' = \hat{n}_{t-1}^{gTr}$ to generate V_t
    $\hat{n}_t^{gTr} = g^{gTr}(V_t, n_{t-1}^{truth})$
    Save $((V_t, n_{t-1}^{truth}), n_t^{truth})$ in Tr training data
    Fit $g^{gTr}(.)$ to $((V_t, n_{t-1}^{truth}), n_t^{truth})$ using loss $\mathcal{L}_t^{gTr} = (\hat{n}_t^{gTr} - n_t^{truth})^2$
  end for
  return Tr training data
end function
function train_teacher_offline(Tr training data)
  Randomly initialize teacher Tr
  Shuffle and split dataset into train and test
  Train Tr with loss $\mathcal{L}_t^{Tr} = (\hat{n}_t^{Tr} - n_t^{truth})^2$
  return Tr
end function

Student Training

function gen_student_training_data(α)
  Randomly initialize $gStu$'s weights
  At $t = 0$, conduct LoF trials to give the initial rough estimate $\hat{n}'_0$; then conduct a BB trial to generate v_0
  $\hat{n}_0^{gStu} = g^{gStu}(v_0, \hat{n}'_0)$
  for $t = 1, \ldots,$ num_iters **do**
    Run BB trial with rough estimate $n' = \hat{n}_{t-1}^{gStu}$
    $\hat{n}_t^{gStu} = g^{gStu}(v_t, \hat{n}_{t-1}^{gStu})$
    Save the tuple $((v_t, \hat{n}_{t-1}^{gStu}), n_t^{truth})$ in Stu training data; also save (V_t, n_{t-1}^{truth}) for Tr prediction
    Fit $g^{gStu}(.)$ to $((v_t, \hat{n}_{t-1}^{gStu}), n_t^{truth})$ with loss $\mathcal{L}_t^{gStu} = \alpha(\hat{n}_t^{gStu} - n_t^{truth})^2 + (1-\alpha)(\hat{n}_t^{Tr} - n_t^{truth})^2$
  end for
  return Stu training data
end function
function train_student_offline(Tr, Stu training data, α)
  Randomly initialize Stu's weights; load pre-trained teacher Tr
  Shuffle and split Stu training data into train and test
  For a particular BB trial with results (v_t, V_t) conducted at time t, calculate the Stu and Tr predictions $\hat{n}_t^{Stu} = g^{Stu}(v_t, \hat{n}_{t-1}^{Stu})$ and $\hat{n}_t^{Tr} = g^{Tr}(V_t, n_{t-1}^{truth})$
  Train Stu with loss $\mathcal{L}_t^{Stu} = \alpha(\hat{n}_t^{Stu} - n_t^{truth})^2 + (1-\alpha)(\hat{n}_t^{Tr} - n_t^{truth})^2$
  return Stu
end function

time slots used by *not* executing LoF trials to obtain the rough estimates for the BB trials of time frames $t = 1, 2, 3, \ldots$.



(a) Homogenous Network (b) Heterogenous Network

FIGURE 4. Figure (a) (respectively, Figure (b)) shows the neural network architecture of a student (*Stu*) model used in the case of a homogeneous (respectively, heterogeneous) network.

The architecture of the NN used is shown in Figure 4a. The input dense layer of length R has Rectified Linear Unit (ReLU) activation, while the other two $R/2$ dense layers (hidden layers) have sigmoid activation. The output layer of length 1 has linear activation. A description of the activation functions is provided in [78]. The architecture has been designed for regression specifically and for ease of training, according to insights from [79], as will now be explained. For regression tasks, it is desirable to have one or two hidden layers of neurons at the most. After training with one hidden layer, it was observed that adding an additional hidden layer improved prediction performance; hence, two hidden layers were used as shown in Figure 4a. Also, trying to predict the number of active nodes from the result of a BB trial is a non-linear problem as can be seen from the description of the BB protocol given in Algorithm 1. In particular, if the number of active nodes in a time frame exhibits correlation with the number of active nodes in the previous time frame as in the system model in Section III, this offers an opportunity for using the estimate of the previous time frame as a rough estimate for the BB trial of the current time frame. A *multi-layer perceptron model* [79] is used, which offers itself as a solution for modeling the non-linear relationship between the vector of results of the BB trial, the previous estimate, and the target current estimate of the number of active nodes.

2) TRAINING

The estimation of the number of active nodes in each time frame is to be done by a model that uses data recorded in a real online scenario. This implies that each element of the results of the trial v_t would comprise of three possibilities: {no transmission, success, collision}. This would be a difficult problem to tackle, and would need more information than the online model has to perform well. Thus, PFD is used, where a teacher model is trained on privileged data not available in the online scenario. In particular, in a practical network, when a collision occurs in a time slot, the BS does not know as to how many nodes transmitted in the slot. So the number of transmitting nodes in a slot can serve as a privileged feature to train the teacher model. Also, in a practical network, the BS does not know the true number of active nodes in the

previous time frame; it only knows the estimated number of active nodes. So the true value of the number of active nodes in the previous time frame can serve as another privileged feature. Thus, in our proposed algorithm, a teacher model is trained on the following privileged data: the number of nodes transmitting in each slot of the BB trial and the true value of the number of active nodes in the previous time frame. A student model is trained on the actual data seen during the online scenario.

In particular, the objective is to train a student model, which is the actual *Stu* NN used in the "Proposed NN Method" in Algorithm 2, to perform online estimation of the number of active nodes, $\hat{n}_t^{Stu}$, in each time frame t using the general feature vector v_t and the previous time frame's estimate $\hat{n}_{t-1}^{Stu}$. The feature vector v_t for a BB trial of length l_{BB} has dimensions $1 \times 3l_{BB}$ since the outcome of each of the l_{BB} slots is represented using *one-hot encoding* and there are 3 possibilities for a slot– no transmission, success, and collision. Also, $R = 3l_{BB} + 1$ in Figure 4a since the NN takes as input the vector v_t and the estimate of the previous time frame.

In the first step, the teacher model is to be trained. The teacher model is fed with privileged information. Let V_t be a $1 \times l_{BB}$ vector denoting the results of the BB trial in time frame t and containing privileged information. In particular, for $i \in \{1, \ldots, l_{BB}\}$, $V_t[i]$ is the number of active nodes transmitting in slot i of the BB trial, which is privileged information as explained above.

Due to the nature of the algorithm, where the NN's output in the previous time frame is part of the input vector for the current time frame, if a network is trained on each time frame one-by-one, it overfits the current sample. It is thus able to predict the next few steps accurately, but the same model will 'forget' the samples seen a few time frames ago. Hence, the procedure of getting a model to be a *genie* to 'track' the evolution of the time series, logging the dataset, shuffling it, and training a new model to mimic the genie is followed. The model trained offline is able to go through the data multiple times and out-of-order.

This calls for a step of data generation– creating a dataset with $((V_t, n_{t-1}^{truth}), n_t^{truth})$ as the (feature, target) tuple for time frame t. The dataset is then shuffled and a new uninitialized teacher network is trained on this dataset.

The teacher training procedure is now explained in detail (see the "Teacher Training" procedure in Algorithm 2). A *genie* teacher network is used, which is represented by gTr. In the function GEN_TEACHER_TRAINING_DATA in the "Teacher Training" procedure in Algorithm 2, a new network gTr is initialized by setting its weights randomly. In time frame 0, LoF trials are conducted to give an initial rough estimate n_0', and a BB trial is conducted with rough estimate n_0'. The resulting privileged information V_0 is used for prediction by gTr to generate $\hat{n}_0^{gTr} = g^{gTr}(V_0, \hat{n}_0')$, where g^{gTr} denotes the genie teacher neural network. Subsequently, in each time frame t, a BB trial is run with rough estimate $\hat{n}_{t-1}^{gTr}$ to generate the privileged vector V_t. The estimate by the *genie* teacher for the trial at time t is $\hat{n}_t^{gTr}$, and is given by the following equation:

$$\hat{n}_t^{gTr} = g^{gTr}(V_t, n_{t-1}^{truth}). \tag{6}$$

The prediction $\hat{n}_t^{gTr}$ is used as the initial rough estimate for the BB trial in time frame $t+1$. The feature vector (V_t, n_{t-1}^{truth}) and the target n_t^{truth} are stored in the teacher Tr training data. Then, the genie teacher gTr is trained on $((V_t, n_{t-1}^{truth}), n_t^{truth})$ with the loss $\mathcal{L}_t^{gTr}$ given by:

$$\mathcal{L}_t^{gTr} = (\hat{n}_t^{gTr} - n_t^{truth})^2. \tag{7}$$

The above steps are executed in each time frame $t = 1, \ldots,$ num_iters, where num_iters is the number of time frames (iterations).

Subsequently, in the function TRAIN_TEACHER_OFFLINE in the "Teacher Training" procedure in Algorithm 2, the generated Tr training data is shuffled to avoid overfitting. The dataset is split into train and test, and a new teacher network Tr is trained on the dataset using the MSE loss given by:

$$\mathcal{L}_t^{Tr} = L(\hat{n}_t^{Tr}, n_t^{truth}) = (\hat{n}_t^{Tr} - n_t^{truth})^2. \tag{8}$$

The teacher network Tr is trained on the train dataset and the test dataset is used to evaluate the test error. Training is done by making multiple passes through the dataset, i.e., by using multiple *epochs*, and to prevent overfitting, the test error is used to evaluate when to stop training. This concludes the training of the teacher network.

Recall that the student model does not see the privileged information (number of active nodes transmitting in each slot, true value of the number of active nodes in the previous time frame). It only sees the result of each slot (no transmission, success, or collision). To train the student model, similar to the teacher model's training, the results of the trials executed in different time frames are recorded offline, and a new model is trained on the recorded data with random shuffling.

The student training procedure is now explained in detail (see the "Student Training" procedure in Algorithm 2). In the function GEN_STUDENT_TRAINING_DATA, similar to the teacher training protocol, a genie student network $gStu$ is initialized by setting its weights randomly. In time frame $t = 0$, LoF trials are conducted and the initial estimate $\hat{n}_0'$ is generated. Next, a BB trial is conducted to generate v_0. Then the estimate $\hat{n}_0^{gStu}$ is computed using $\hat{n}_0^{gStu} = g^{gStu}(v_0, \hat{n}_0')$, where g^{gStu} denotes the genie student neural network.

Subsequently, in each time frame t, $gStu$ uses the result of the BB trial of time frame t, viz., v_t, and the estimate of the previous trial $\hat{n}_{t-1}^{gStu}$ to compute $\hat{n}_t^{gStu} = g^{gStu}(v_t, \hat{n}_{t-1}^{gStu})$. The feature vector for the student $(v_t, \hat{n}_{t-1}^{gStu})$, the target n_t^{truth}, and the feature vector for the teacher (V_t, n_{t-1}^{truth}) are stored in the Stu training data. $gStu$ is then fit on the current sample with the distillation loss, which is a combination of the data loss and the teacher loss (see step 9 in the "Student Training" procedure in Algorithm 2). The above steps are executed in each time frame $t = 1, \ldots,$ num_iters, where num_iters is the number of time frames (iterations).

Next, in the function TRAIN_STUDENT_OFFLINE in the "Student Training" procedure in Algorithm 2, a new student model *Stu* is initialized and trained on the recorded data, *Stu* training data, and with the pre-trained teacher *Tr*'s predictions as follows. *Stu* training data is first shuffled and split into train and test. Then, in each time frame t, the general information v_t and the privileged information V_t are used for *Stu* and *Tr* inference, giving $\hat{n}_t^{Stu} = g^{Stu}(v_t, \hat{n}_{t-1}^{Stu})$ and $\hat{n}_t^{Tr} = g^{Tr}(V_t, n_{t-1}^{truth})$, respectively, as the *Stu* and *Tr* estimates. Instead of minimizing the standard regression MSE loss, *Stu*'s weights are adjusted for minimizing the distillation loss, which includes the loss between the teacher's prediction and the truth, and is given by:

$$\mathcal{L}_t^{Stu} = (1-\alpha)L(\hat{n}_t^{Tr}, n_t^{truth}) + \alpha L(\hat{n}_t^{Stu}, n_t^{truth}), \quad (9)$$

where α is the mixing ratio. Intuitively, a high α gives less importance to the *Tr* loss and vice versa.

The time complexity of the above proposed algorithm is the time complexity of the inference (forward propagation) step of the student neural network. Recall that a fully connected hidden layer with m inputs and n outputs can be represented as an $m \times n$ matrix, and would thus involve $\mathcal{O}(mn)$ operations. Hence, considering the architecture of the student network (see Figure 4a), with the length of the input vector being $R = 3l_{BB} + 1$, the time complexity of inference is $\mathcal{O}(\frac{3R^2}{4} + \frac{R}{2}) = \mathcal{O}(l_{BB}^2)$.

3) OTHER ALGORITHMS FOR COMPARISON

As a benchmark for comparison with the student model, in each time frame, the SRC$_s$ protocol (see Algorithm 1) is executed, with the number of time slots used being the same as the length of the BB trial used by the NN (student) model. There is an inherent disadvantage to the SRC$_s$ protocol since it does not use knowledge (e.g., estimate of the number of active nodes) from the previous time frame, unlike the NN method. To analyse whether the NN method estimates the number of active nodes better than an SRC$_s$ based algorithm that uses knowledge of an estimate of the number of active nodes in the previous time frame, the former is compared against the algorithm *BB-Aware*, which operates as follows. BB-Aware executes SRC$_s$ in time frame $t = 0$ to generate a rough estimate $\hat{n}_0^{BB-Aware}$. In each subsequent time frame $t = 1, \ldots,$ num_iters, BB-Aware conducts a BB trial of length $l_{BB-Aware}$ and with rough estimate $\hat{n}_{t-1}^{BB-Aware}$ (estimate of number of active nodes in the previous time frame) and computes the estimate $\hat{n}_t^{BB-Aware}$ as a function of the number of empty slots in the trial (as in steps 3-8 of the BB protocol in Algorithm 1).

For a fair comparison, it is ensured that each algorithm takes the same number of time slots to execute in each time frame by ensuring that the following equation holds:

$$l_{BB-Aware} = l_{BB} = (l_{BB-SRC_s} + \text{num\_lof} \times l_{lof}), \quad (10)$$

where $l_{BB-Aware}$ is the length of the BB trial under BB-Aware, l_{BB} is the length of the BB trial under the NN method, l_{BB-SRC_s} is the length of the BB trial under SRC$_s$, l_{lof} is the length of each LoF trial, and num_lof is the number of LoF trials that are conducted as part of the execution of SRC$_s$. For example, if the total number of time slots in a time frame is to be 100, then the NN method performs a BB trial of length 100, the SRC$_s$ protocol performs 3 LoF trials of length 8 each, and a BB trial of length 76, while the BB-Aware method performs a BB trial of length 100.

B. HETEROGENEOUS NETWORK

1) PROPOSED ALGORITHM

Consider the model and problem formulation described in Section III-B. In this case, the coverage area of a BS contains T types of nodes. The problem is to estimate $\mathbf{n}_t^{truth}$, a $1 \times T$ vector.

The approach followed is largely similar to that described in Section V-A.1. Recall from Section V-A.1 that in the homogeneous case, a BB trial is conducted in each time frame; instead, in the heterogeneous case, in each time frame, 3-SS-BB (explained in Section IV-B) is conducted, and the results of all the slots are converted into a feature vector.

The architecture of the NN used is shown in Figure 4b. The input dense layer of length R has ReLU activation, while the other two $R/2$ dense layers have sigmoid activation. The output layer of length R has linear activation. A description of the activation functions is provided in [78].

2) TRAINING

The teacher is trained on the feature vector V_t^T, which contains the number of nodes of each type participating in each of the l blocks of 3-SS-BB, and $\mathbf{n}_{t-1}^{truth}$; note that the size of $\left(V_t^T, \mathbf{n}_{t-1}^{truth}\right)$ is $(l+1)T$. The student is trained on v_t^T, which contains the result of each slot (0, α, β or c (see Section IV-B)) in each of the l blocks of 3-SS-BB in one-hot encoding fo rmat, and $\hat{\mathbf{n}}_{t-1}^{Stu}$, which is the vector of estimates of the numbers of active nodes of the T types produced by the student in time frame $t-1$; note that the size of $\left(v_t^T, \hat{\mathbf{n}}_{t-1}^{Stu}\right)$ is $4(T-1)l+T$. The training of the teacher and student proceed as per the "Teacher Training" and "Student Training" procedures in Algorithm 2, respectively, with the feature vectors being as in the heterogeneous case.

The time complexity of the above proposed algorithm, which is the time complexity of the inference step of the student neural network, will now be analyzed. Considering the architecture of the student network (see Figure 4), with the length of the input vector being $R = 4(T-1)l_{3-SS-BB} + T$, the time complexity of inference is $\mathcal{O}(\frac{3R^2}{4} + \frac{TR}{2}) = \mathcal{O}(12(T-1)^2 l_{3-SS-BB}^2 + 10T(T-1)l_{3-SS-BB} + 7T^2/4)$, i.e., $\mathcal{O}(T^2 l_{3-SS-BB}^2)$.

3) OTHER ALGORITHMS FOR COMPARISON

As a benchmark for comparison with the student model, in each time frame, SRC$_s$ is independently run T times (henceforth referred to as T-SRC$_s$)– once for each type of node. Similarly, the BB-Aware algorithm described in

Section V-A.3 is adapted to the heterogeneous case to give the algorithm T-BB-Aware, wherein in each time frame, BB-Aware is independently run T times– once for each type of node.

For a fair comparison, each algorithm takes the same number of time slots to execute in each time frame. In particular, the lengths of the BB trials in 3-SS-BB and T-SRC$_s$ are related as in the following equation:

$$l_{3-SS-BB}(T-1) = (l_{BB-SRC_s} + \text{num\_lof} \times l_{lof})T, \quad (11)$$

where $l_{3-SS-BB}$ is the length of the BB trial (number of blocks) in 3-SS-BB, l_{BB-SRC_s} is the length of the BB trial for each node type under T-SRC$_s$, l_{lof} is the length of each LoF trial and num_lof is the number of LoF trials that are conducted as part of the execution of SRC$_s$ for each node type under T-SRC$_s$. The length of the BB trial in 3-SS-BB ($l_{3-SS-BB}$) is initially fixed and the length of each BB trial under T-SRC$_s$ (l_{BB-SRC_s}) is computed using (11) and rounded to the nearest integer. The lengths of the BB trials in 3-SS-BB and T-BB-Aware are related as in the following equation:

$$l_{3-SS-BB}(T-1) = l_{BB-Aware}T. \quad (12)$$

The length of a BB trial, $l_{3-SS-BB}$, in 3-SS-BB is fixed and the length of the BB trials in T-BB-Aware, $l_{BB-Aware}$, is computed using (12) and rounded to the nearest integer.

Remark 1: Under our proposed neural network based methods for node cardinality estimation in homogeneous and heterogeneous wireless networks, training of the student neural network is done only once– prior to its deployment in the wireless network– and the complexity of the estimation algorithms used during the testing phase is low. Moreover, our proposed estimation algorithms are designed for execution at a BS, gateway, etc., which has sufficient computational resources even in environments where the end devices are resource-constrained. Hence, our proposed methods are suitable for implementation even in resource-constrained environments such as IoT networks.

Remark 2: Note that energy or power detection can be potentially used for node cardinality estimation in a wireless network. In particular, the set of active nodes can simultaneously transmit signals at fixed power, and the total received energy at the BS can be used to estimate the number of active nodes– the higher the received energy, the larger the estimate of the number of active nodes. However, this method of estimating node cardinalities is prone to large errors because the qualities of the channels from different nodes to the BS are unknown and vary with time, in general, due to mobility of the nodes and/ or objects in the environment. So, e.g., the received energy from a node that is close to the BS and such that there are no obstacles between the node and the BS will be large and vice versa. Also, for a given received energy level at the BS, it is difficult to know whether the energy was received from a small number of nodes with good channel qualities to the BS or a large number of nodes with poor channel qualities to the BS. Hence, the errors in the node cardinality estimates computed using the above energy detection based method can be large. In this paper, our objective is to estimate node cardinalities with high accuracy. Hence, in this paper, energy or power detection is not used to estimate the node cardinalities. Note that there is an extensive research literature on node cardinality estimation in wireless networks, in which the proposed algorithms do not use energy or power detection, e.g., [20], [21], [37], [38], [39], [43], [45], [46], and [58], etc.

VI. PERFORMANCE EVALUATION

Simulation results for a synthetic dataset as well as for a real dataset are provided in this section.

The simulation setup for the synthetic dataset case is described in Section VI-A. In Section VI-B (respectively, Section VI-C), our simulation results for the synthetic dataset are provided for the case of a homogeneous (respectively, heterogeneous) network. Our synthetic dataset is available for download at [55], so that it can be used for future work by the research community.

Our simulation results for a real dataset– the Intel Lab Data dataset [56]– including those for homogeneous as well as heterogeneous networks, are provided in Section VI-D.

A. SIMULATION SETUP: SYNTHETIC DATASET CASE

Since the numbers of active nodes in successive time frames are highly correlated as explained in Section III-A, the evolution of the number of active nodes of a given type over different time frames is modeled by a DTMC with N states $\{0, 1, \ldots, N-1\}$, with a Transition Probability Matrix (TPM) $P = [p_{i,j}]$, where the transition probabilities are given by the following equation:

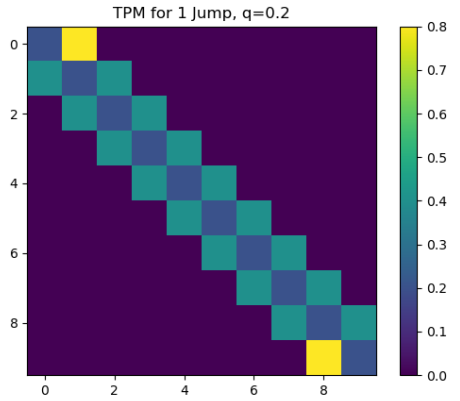
$$p_{i,j} = \begin{cases} q, & \text{if } i = j, \\ 1-q, & \text{if } i = 0, j = 1, \\ 1-q, & \text{if } i = N-1, j = N-2, \\ 1-p-q, & \text{if } i \neq 0, N-1 \text{ and } j = i-1, \\ p, & \text{if } i \neq 0, N-1 \text{ and } j = i+1. \end{cases} \quad (13)$$

The case when $p = (1-q)/2$, which indicates that it is equally likely to go from a state i to states $i+1$ and $i-1$, is considered. A visual representation of P is shown in Figure 5a. In order to model more sudden changes, the k-step TPM (P^k), which allows changes in the number of active nodes from one time frame to the next one by more than 1, is considered. An example is shown in Figure 5b.

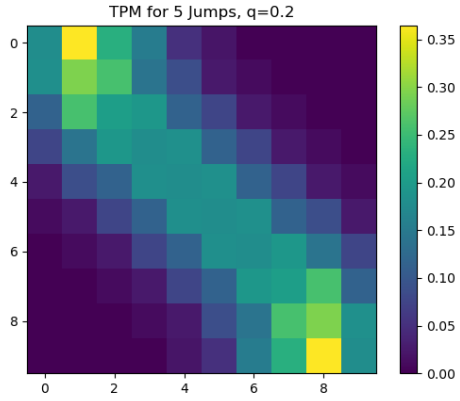
In the homogeneous case, let n_t^{truth} be the number of active nodes in time frame t. The system evolves as in the following equation:

$$\mathbb{P}[n_t^{truth} = j | n_{t-1}^{truth} = i] = P^k[i,j], \quad (14)$$

where $P^k[i,j]$ is the (i,j)'th element of the matrix P^k. Also, in the heterogeneous case, the number of active nodes of each type evolves as in (14), with the DTMCs for different types of nodes being independent.



(a) P for $q = 0.2$



(b) P^5 for $q = 0.2$

FIGURE 5. Figures (a) and (b) show the TPMs *P* and P^5, respectively, for $q = 0.2$.

Throughout the simulations for the synthetic dataset case, for the homogeneous case, the maximum number of active nodes in a time frame is taken to be 64 and for the heterogeneous case, the maximum number of active nodes of each type in a time frame is taken to be $\lfloor 192/T \rfloor$, where T is the number of types. Table 1 summarizes the meanings of different simulation parameters.

B. HOMOGENEOUS NETWORK: SYNTHETIC DATASET CASE

Throughout the paper, an ''epoch'' refers to one complete pass through the entire training dataset during training. The evolution of the training and test loss over 2500 epochs is plotted, when the student is trained using the teacher. It was observed that around 500 epochs, the test loss increases significantly, while having insignificant variation in the training loss. The training of the student was thus stopped at that point. Figure 6 shows the evolution of the training and test loss of the student g^{Stu} over 500 epochs, when the student is trained using the teacher g^{Tr}. The mixing ratio used in the training is $\alpha = 0.1$. The training loss decreases faster than the test loss, which stops decreasing around 500 epochs, which is why training is stopped at that point to avoid overfitting.

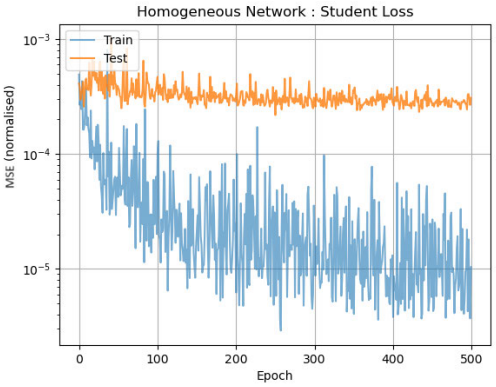


FIGURE 6. The figure shows the training of the student using the teacher over epochs with $l_{BB} = 100$, $\alpha = 0.1$, $k = 5$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames.

TABLE 1. The table summarizes the meanings of different simulation parameters.

Simulation Parameters	
Parameter	Meaning
T	Number of types of nodes
α	Mixing ratio
l	Length of trial
N	Number of states in the DTMC of number of active nodes of a given type
P	TPM of the DTMC of number of active nodes of a given type
k	Number of steps in the TPM of the DTMC of number of active nodes of a given type
l_{BB}	Length of a BB trial under NN
l_{lof}	Length of an LoF trial under SRC_s
l_{BB-SRC_s}	Length of a BB trial under SRC_s
$l_{3-SS-BB}$	Length of a 3-SS-BB trial
num_lof	Number of LoF trials conducted
ϵ	Relative error tolerated in node cardinality estimation
δ	Node cardinality estimation confidence parameter

Figure 7 shows the normalized MSEs in the active node cardinality estimates computed by the trained student NN, the SRC_s protocol, and the BB-Aware method, in different time frames in a homogeneous network. The average MSEs achieved by different methods are given above the plot. Recall that the number of active nodes varies as per the DTMC in (14), with $k = 5$. Thus, the number of active nodes in a time frame is correlated with the number of active nodes in the previous time frame. In this scenario, it is seen that the student NN achieves much lower normalized MSEs than both the other methods.

Experiment 1: Changing the Length of the BB Trial: To study the variation of the error on changing the length of

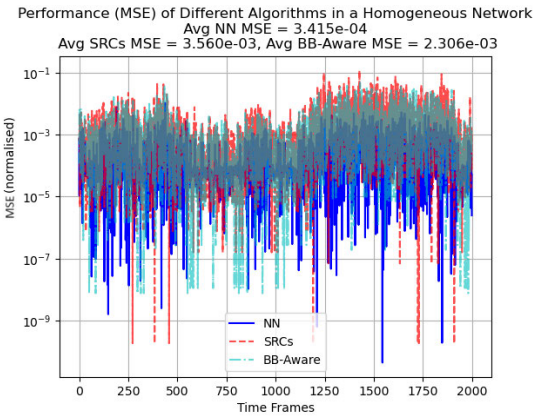


FIGURE 7. The figure shows the normalized MSEs achieved by the NN, SRC$_s$, and BB-Aware methods, for a student network with $l_{BB} = 100$, $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames.

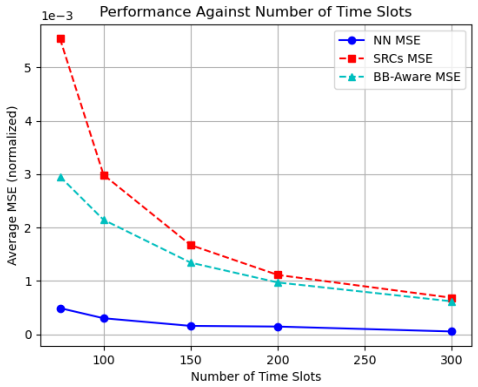


FIGURE 8. The figure shows the performance of different methods versus the number of time slots per time frame, with $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames.

the BB trial, a different student network was trained for each length of trial. Each trained student network was then evaluated on a scenario where the number of active nodes evolved as per a DTMC, as in (14). The performance (i.e, the normalised MSE) was averaged over 20 runs of 2000 time frames each. The normalised MSE was then compared with those of the SRC$_s$ protocol and BB-Aware protocol. SRC$_s$ and BB-Aware were configured to make sure that each algorithm takes the same number of time slots for generating an estimate for a time frame as in (10). The result is shown in Figure 8. It is seen that the errors of all the methods decrease with an increase in the length of trial, which is to be expected because a longer trial provides more information about the number of active nodes than a short trial, as there are less collisions. Also, as discussed in Section IV-A, the length of the BB trial in SRC$_s$ is given by $l_{BB} = \frac{65}{(1-0.04^{\epsilon})^2}$, where ϵ is the relative error tolerated. Thus, increasing the number of time slots, l_{BB}, will lead to a lower relative error with high probability, which explains why the error of SRC$_s$ decreases with an increase

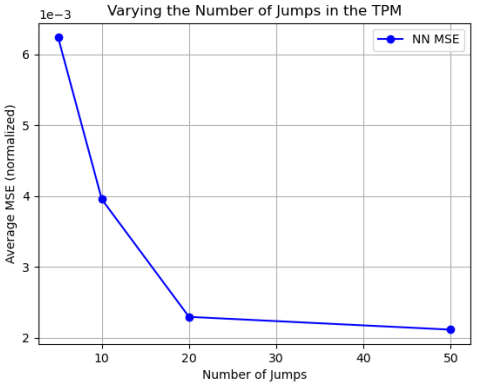


FIGURE 9. The figure shows the performance of the student network versus the number of jumps in the TPM, for a student network with $l_{BB} = 100$, $\alpha = 0.1$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames.

in the length of trial. The BB-Aware method is identical to SRC$_s$, except that it uses the previous time frame's estimate as the rough estimate in the current time frame, instead of computing it using the results of the LoF trials. This explains why the error of BB-Aware decreases in the length of trial. Next, consider the performance of the NN method. Increasing the trial length (number of time slots) leads to a larger input vector and a larger NN architecture. The experimental setup is such that the same time series of the numbers of active nodes is used, while changing the length of the trial. This leads to a sparser BB trial result for a higher length of trial, thus decreasing the number of collisions in the vector. This leads to an increase in performance of the NN, leading to lower average normalised MSE. The NN performs better than SRC$_s$ and BB-Aware consistently. Hence, for achieving the same normalized MSE, a NN can make use of a shorter trial than the SRC$_s$ and BB-Aware protocols.

Conclusion: An increase in the length of the BB trial causes the errors of NN, SRC$_s$ and BB-Aware to decrease, with NN outperforming SRC$_s$ and BB-Aware for every length.

Experiment 2: Changing the Number of Jumps k: To study how the methods perform when the system evolves faster or slower, the DTMC according to which the number of active nodes in a time frame evolves (see Section VI-A) is changed by changing the number of jumps taken in one time frame. Specifically, for a DTMC with transition probability matrix P^k, by changing the number of jumps k, one can model a faster or slower changing time series. For each situation, a different student network is trained and the performance is evaluated by averaging over 20 runs. Figure 9 shows that the normalized MSE achieved by the NN decreases with an increase in the number of jumps. This trend can be explained as follows. For the same number of time frames that the student is trained on, a DTMC with higher k offers more variation, and thus the NN is trained better and its achieved error decreases. That is, as the time series is fast varying, the student network gets trained on more adverse scenarios, which affects the performance in the same way that more

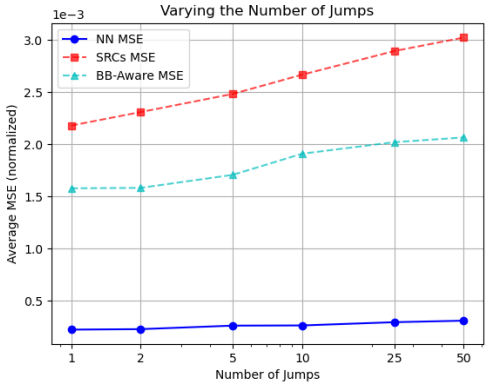


FIGURE 10. The figure shows the MSEs achieved by different algorithms versus the number of jumps with $l_{BB} = 100$, $\alpha = 0.1$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames. The same student model trained on 5 jumps is used for the estimation for every value of the number of jumps.

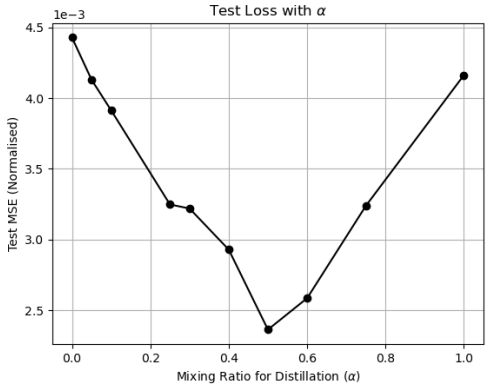


FIGURE 11. The figure shows the variation of the test loss with the mixing ratio α for a student network with $l_{BB} = 100$, $k = 5$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames.

outliers in a dataset decrease the tendency of a NN to regress to the mean. In short, training the student on faster time series reduces overfitting.

Conclusion: As the NN model is trained on more 'adverse' scenarios when the number of jumps in the DTMC is higher (more outliers), the error decreases with an increase in the number of jumps.

Experiment 3: Testing the Same NN Model With Fast or Slow Time Series: Since the student model is trained on a synthetic time series, the performance of the algorithm with faster and slower time series needs to be tested. If a single trained student model is evaluated with different DTMCs (different values of k), the error does not vary much, as seen in Figure 10. Note that in faster varying systems (higher k), the estimate from the previous time frame, which is used as the rough estimate for the BB trial in the current time frame, is more unreliable. The fact that despite this, the error achieved by the NN does not increase much in k suggests that the student network has learned to map the results of the current trial to the number of active nodes well, rather than relying heavily on the estimate of the previous time frame. Also, to compare the behavior of NN with the other two methods, the performance of the three methods is studied together in Figure 10. The performance of SRC$_s$ degrades with a faster time series as more 'adverse' outliers are seen with an increase in the number of jumps. BB-Aware outperforms SRC$_s$, but the former also performs worse under a faster time series (higher k), which is expected since it uses its estimate from the previous time frame as the rough estimate for the BB trial of the current time frame, and the correlation between the true values of the numbers of active nodes in the previous time frame and in the current time frame decreases as k increases. The error in the trained student, in contrast, does not vary significantly for faster or slower time series. This indicates that the network is not overfit to any particular time series, fast or slow. Figure 10 also shows that for all values of k, NN significantly outperforms BB-Aware and SRC$_s$.

Conclusion: The NN model learns the dependence between the v_t vector and the target n_t^{truth}, and does not simply repeat $\hat{n}_{t-1}^{Stu}$; also, for all values of k, it significantly outperforms BB-Aware and SRC$_s$.

Experiment 4: Varying the Mixing Ratio α: Next, the dependence of the test loss on the mixing ratio α (see (9)) is studied. It can be seen from Figure 11 that the test loss decreases when α is increased up to a certain point, then increases again as α approaches 1. Figure 11 shows that distillation offers a significant benefit over blind training the student ($\alpha = 1$). Also, the figure shows that the best result (lowest test loss) is achieved for a value of α around 0.5. These trends can be explained as follows. There are two contrasting effects at play: $\alpha = 1$ corresponds to no teacher input, which means blind training. So as $\alpha \to 1$, the student has a greater tendency to ignore the teacher's predictions, which are made on privileged data. Thus, the student essentially tries to predict on just the general features and no distillation benefit can be obtained. On the other hand, as $\alpha \to 0$, the student tries to mimic the teacher more. This also implies that the student overfits to the teacher's prediction, trying to capture the standard deviation in the teacher's prediction. This also contributes to the test loss. Hence, the most benefit is obtained at values of α around the middle of the range (0, 1).

Conclusion: Distillation offers a significant benefit over regular NN model training.

C. HETEROGENEOUS NETWORK: SYNTHETIC DATASET CASE

Figure 12 shows the training curves for offline training of a teacher model. It is seen that the test loss and training loss both decrease and settle quickly, indicating a relatively simple function to model. In contrast, Figure 13 shows the training curves for offline training of the corresponding student model. As the student model is larger than the teacher model,

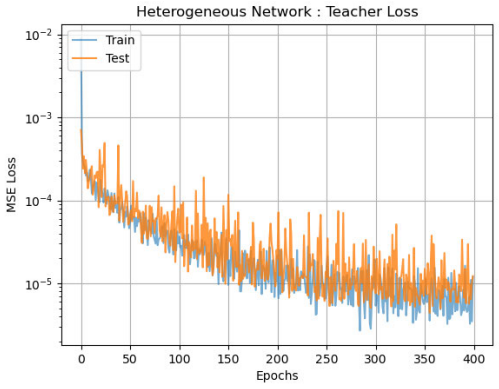


FIGURE 12. The figure shows the training of the teacher in the heterogeneous network case with $l_{3-SS-BB} = 100$, $T = 3$, $k = 5$, and a teacher dataset of 10^4 time frames.

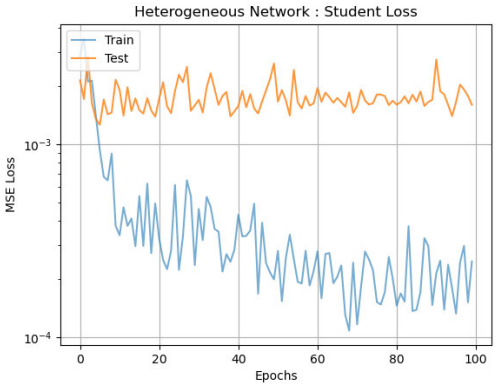


FIGURE 13. The figure shows the training of the student in the heterogeneous network case, with $l_{3-SS-BB} = 100$, $T = 3$, $\alpha = 0.1$, $k = 5$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames.

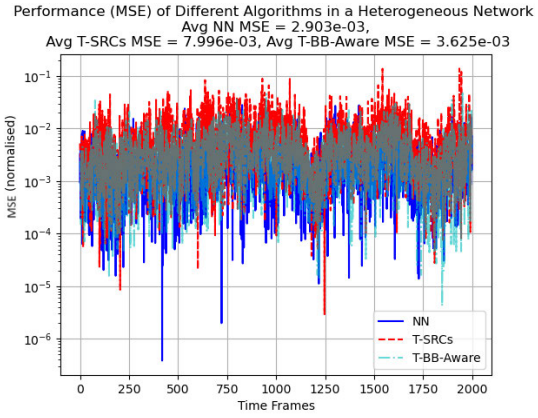


FIGURE 14. The figure shows the normalized MSEs achieved by the NN, T-SRC$_s$, and T-BB-Aware methods, with $l_{3-SS-BB} = 100$, $T = 3$, $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames, and an evaluation run of 2000 time frames.

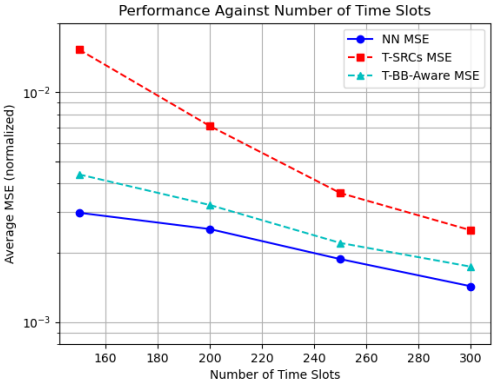


FIGURE 15. The figure shows the average MSEs achieved by different algorithms versus the number of time slots taken by each algorithm in a time frame, with $T = 3$, $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 2×10^4 time frames and a student dataset of 2×10^4 time frames.

it is slower to train. The function mapping the student input to the target is also significantly complex. This leads to the model overfitting to the training data, and the test loss remains roughly constant, while the training loss decreases.

After the training is complete, the trained student model is deployed in an online scenario. Figure 14 shows the performances of the student model, T-SRC$_s$ and T-BB-Aware methods versus the time frame number for 1000 time frames. The average MSEs achieved by different methods are given above the plot. It can be seen that the NN model significantly outperforms T-SRC$_s$ as well as T-BB-Aware.

Experiment 1: Changing the Length of the Trial: The length of the trial ($l_{3-SS-BB}$) is changed, and a different student model is trained on the generated data for each value of $l_{3-SS-BB}$. The other two methods, viz., T-SRC$_s$ and T-BB-Aware, are also configured to use the same total number of time slots per time frame to produce estimates as the NN method. The results are shown in Figure 15. The trends in this figure and the reasons for them are similar to those for the corresponding homogeneous case– see our discussion of Figure 8 for details. In particular, as expected, in Figure 15, the error decreases for the NN, T-SRC$_s$, and T-BB-Aware methods as the number of time slots increases; this is because the number of collisions decreases under each method. The NN method significantly outperforms both the T-SRC$_s$ and T-BB-Aware methods, which indicates that to achieve the same average error, the NN method can deliver with a shorter trial than both T-SRC$_s$ and T-BB-Aware. T-BB-Aware outperforms T-SRC$_s$, as it uses a longer trial and uses information (estimates of numbers of active nodes of different types) from the previous trial.

Conclusion: The NN model requires a far shorter trial than T-SRC$_s$ and T-BB-Aware to offer a specified average MSE.

Experiment 2: Changing the Number of Types of Nodes: Keeping the total number of nodes across all types present in the network the same (equal to 192), the number of types of nodes (T) is now varied. The maximum number of active nodes of each type in a time frame is taken to be $\lfloor 192/T \rfloor$. The errors achieved by different methods are shown in Figure 16.

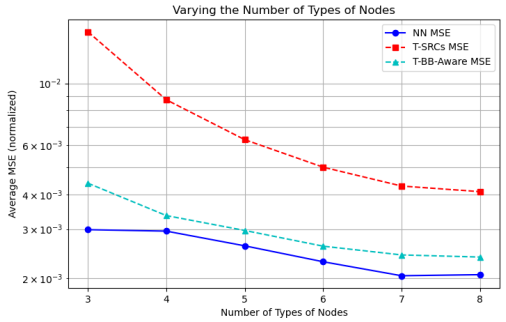


FIGURE 16. The figure shows the average MSEs achieved by different algorithms versus the number of types of nodes (T), with $l_{3-SS-BB} = 75$, $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 2×10^4 time frames and a student dataset of 2×10^4 time frames.

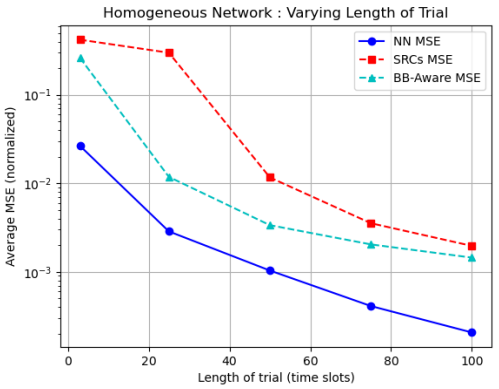


FIGURE 17. The figure shows the performance of different methods versus the length of trial, with $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames, for the real dataset and a homogeneous network.

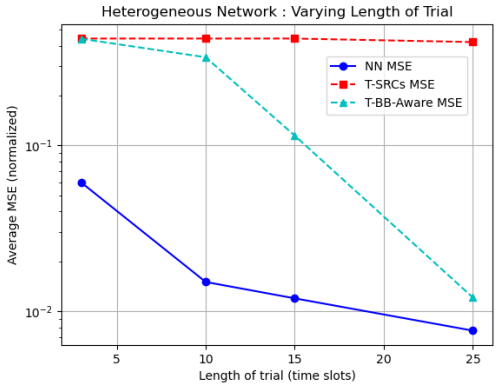


FIGURE 18. The figure shows the performance of different methods versus the length of trial, with $T = 3$, $\alpha = 0.1$, $k = 5$, num_lof $= 3$, $l_{lof} = 8$, a teacher dataset of 10^4 time frames and a student dataset of 10^4 time frames, for the real dataset and a heterogeneous network.

The trends in this figure can be explained as follows. Note that $l_{3-SS-BB}$ is constant; also, in order to keep the time taken by each method the same, for T-SRC$_s$, the length of each BB trial, l_{BB-SRC_s}, is governed by (11). So as T increases, l_{BB-SRC_s} also increases. Also, the maximum number of active nodes of each type in a time frame is $\lfloor \frac{192}{T} \rfloor$. Thus, as T increases, the length of the trial increases, and the maximum number of active nodes of each type decreases. This leads to smaller errors for T-SRC$_s$ and, similarly, also for T-BB-aware, due to the fact that there are fewer collisions. The NN architecture for higher numbers of types of nodes has a larger number of weights due to the fully connected output layer of size T (see Figure 4b). It is interesting to note that the NN method consistently gives lower average MSEs than the other two methods, even though the complexity of the mapping problem increases with T. Also, the figure shows that the NN method significantly outperforms both the T-SRC$_s$ and T-BB-Aware methods.

Conclusion: The NN method can adapt well to a higher number of types of nodes (T) while achieving significantly lower error than T-SRC$_s$ and T-BB-Aware for all values of T.

D. PERFORMANCE EVALUATION WITH REAL DATASET

Next, the models that are trained on synthetic data, as explained earlier, are tested with a real dataset– the Intel Lab Data dataset [56]. This dataset consists of measurements from 54 sensors deployed in the Intel Berkeley Research Lab. The time series of the number of active nodes (nodes that transmit sensor readings) was processed and binned into 1 minute intervals. Two kinds of experiments were performed: (i) performance evaluation for a homogeneous network, which is the network of all the 54 sensors, and (ii) performance evaluation for a heterogeneous network with $T = 3$ types of nodes, which is obtained by dividing the set of 54 sensors into three subsets of 18 sensors each. Next, the results of these experiments are described.

Figure 17 shows the performance of the student NN method, SRC$_s$ and BB-Aware on varying the trial length with a homogeneous population of nodes. The normalised MSE is calculated across 20 runs of length 2000 time frames each. The trends are similar to those in Figure 8; in particular, the performance of all three methods improves with a longer length of trial. The reasons for these trends are similar to those explained for Figure 8. Also, it is seen that the student NN consistently outperforms the other two methods for all values of the length of trial. In particular, the student NN method achieves a MSE that is 92.35% lower on average than that achieved by the SRC$_s$ protocol.

Figure 18 shows the performance of the student NN method, T-SRC$_s$ and T-BB-Aware on varying the trial length with a heterogeneous population. Again, 20 runs of length 2000 time frames each are performed for each length of trial. The trends, and reasons for them, are similar to those explained for Figure 15. In particular, the student NN method consistently outperforms the other two methods for all values of the trial length: specifically, it achieves a MSE that is

94.08% lower on average than that achieved by the T-SRC$_s$ protocol.

VII. CONCLUSION AND FUTURE WORK

A novel methodology for node cardinality estimation in homogeneous as well as heterogeneous wireless networks, which uses the PFD technique and works using a neural network with a teacher-student model, has been proposed. Using extensive simulations conducted on a synthetic dataset as well as on a real dataset, it has been shown that the neural networks trained using PFD significantly outperform state-of-the-art node cardinality estimation algorithms. In particular, for a fixed number of time slots per time frame, the proposed PFD based algorithms achieve much lower average normalised MSE than SRC$_s$ and T-SRC$_s$. Moreover, the proposed PFD based algorithms also outperform the SRC$_s$ based BB-Aware and T-BB-Aware methods, which use information from the previous time frame and hence have longer BB trials, in homogeneous and heterogeneous wireless networks, respectively. Our work demonstrates that PFD is a promising approach for effectively solving the problem of node cardinality estimation in wireless networks.

In this paper, it is assumed that the BS is stationary. A direction for future research is to extend the techniques proposed in this paper to the case where an MBS moves around, making multiple stops, for node cardinality estimation in a large region in which a homogeneous or heterogeneous wireless network is deployed.

REFERENCES

[1] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2M: From mobile to embedded Internet," *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 36–43, Apr. 2011.

[2] Y. Liu, C. Yuen, X. Cao, N. U. Hassan, and J. Chen, "Design of a scalable hybrid MAC protocol for heterogeneous M2M networks," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 99–111, Feb. 2014.

[3] A. Rajandekar and B. Sikdar, "A survey of MAC layer issues and protocols for machine-to-machine communications," *IEEE Internet Things J.*, vol. 2, no. 2, pp. 175–186, Apr. 2015.

[4] S. Kadam, C. S. Raut, and G. S. Kasbekar, "Fast node cardinality estimation and cognitive MAC protocol design for heterogeneous M2M networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–7.

[5] S. Kadam, C. S. Raut, A. D. Meena, and G. S. Kasbekar, "Fast node cardinality estimation and cognitive MAC protocol design for heterogeneous machine-to-machine networks," *Wireless Netw.*, vol. 26, no. 6, pp. 3929–3952, Aug. 2020.

[6] J. T. Liew, F. Hashim, A. Sali, M. F. A. Rasid, and A. Jamalipour, "Probability-based opportunity dynamic adaptation (PODA) of contention window for home M2M networks," *J. Netw. Comput. Appl.*, vol. 144, pp. 1–12, Oct. 2019.

[7] K. Kanistras, G. Martins, M. J. Rutherford, and K. P. Valavanis, "A survey of unmanned aerial vehicles (UAVs) for traffic monitoring," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, May 2013, pp. 221–234.

[8] G. Giambene, E. O. Addo, and S. Kota, "5G aerial component for IoT support in remote rural areas," in *Proc. IEEE 2nd 5G World Forum (5GWF)*, Sep. 2019, pp. 572–577.

[9] T. D. Dinh, D. T. Le, T. T. T. Tran, and R. Kirichek, "Flying ad-hoc network for emergency based on IEEE 802.11 p multichannel MAC protocol," in *Proc. Int. Conf. Distrib. Comput. Commun. Netw.* Cham, Switzerland: Springer, 2019, pp. 479–494.

[10] L. Arjona, H. Landaluce, A. Perallos, and E. Onieva, "Timing-aware RFID anti-collision protocol to increase the tag identification rate," *IEEE Access*, vol. 6, pp. 33529–33541, 2018.

[11] C.-G. Liu, I.-H. Liu, C.-D. Lin, and J.-S. Li, "A novel tag searching protocol with time efficiency and searching accuracy in RFID systems," *Comput. Netw.*, vol. 150, pp. 201–216, Feb. 2019.

[12] X. Liu, J. Yin, J. Liu, S. Zhang, and B. Xiao, "Time efficient tag searching in large-scale RFID systems: A compact exclusive validation method," *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1476–1491, Apr. 2022.

[13] J. Yu, W. Gong, J. Liu, L. Chen, and K. Wang, "On efficient tree-based tag search in large-scale RFID systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 42–55, Feb. 2019.

[14] J. Yu, W. Gong, J. Liu, and L. Chen, "Fast and reliable tag search in large-scale RFID systems: A probabilistic tree-based approach," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 1133–1141.

[15] K. Liu, L. Chen, J. Huang, S. Liu, and J. Yu, "Revisiting RFID missing tag identification," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 710–719.

[16] A. Fahim, T. Elbatt, A. Mohamed, and A. Al-Ali, "Towards extended bit tracking for scalable and robust RFID tag identification systems," *IEEE Access*, vol. 6, pp. 27190–27204, 2018.

[17] W. Zhu, X. Meng, X. Peng, J. Cao, and M. Raynal, "Collisions are preferred: RFID-based stocktaking with a high missing rate," *IEEE Trans. Mobile Comput.*, vol. 19, no. 7, pp. 1544–1554, Jul. 2020.

[18] Y. Zhang, S. Chen, Y. Zhou, and O. Odegbile, "Missing-tag detection with presence of unknown tags," in *Proc. 15th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2018, pp. 1–9.

[19] J. Liu, X. Chen, S. Chen, W. Wang, D. Jiang, and L. Chen, "Retwork: Exploring reader network with $*COTS*$$*RFID*$ systems," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2020, pp. 889–896.

[20] C. Qian, H. Ngan, Y. Liu, and L. M. Ni, "Cardinality estimation for large-scale RFID systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1441–1454, Sep. 2011.

[21] L. Arjona, H. Landaluce, A. Perallos, and E. Onieva, "Scalable RFID tag estimator with enhanced accuracy and low estimation time," *IEEE Signal Process. Lett.*, vol. 24, no. 7, pp. 982–986, Jul. 2017.

[22] Y. Hou, J. Ou, Y. Zheng, and M. Li, "PLACE: Physical layer cardinality estimation for large-scale RFID systems," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2702–2714, Oct. 2016.

[23] J. Liu, Y. Zhang, S. Chen, M. Chen, and L. Chen, "Collision-resistant communication model for state-free networked tags," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 656–665.

[24] Q. Lin, L. Yang, C. Duan, and Z. An, "Tash: Toward selective reading as hash primitives for Gen2 RFIDs," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 819–834, Apr. 2019.

[25] Z. Zhou and B. Chen, "RFID counting over time-varying channels," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2018, pp. 1142–1150.

[26] P. C. Ng, J. She, P. Spachos, and R. Ran, "A fast item identification and counting in ultra-dense beacon networks," in *Proc. GLOBECOM IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[27] S. Duan, V. Shah-Mansouri, Z. Wang, and V. W. S. Wong, "D-ACB: Adaptive congestion control algorithm for Bursty M2M traffic in LTE networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9847–9861, Dec. 2016.

[28] K. Ashrafuzzaman and A. O. Fapojuwo, "Efficient and agile carrier sense multiple access in capillary Machine-to-Machine communication networks," *IEEE Access*, vol. 6, pp. 4916–4932, 2018.

[29] C.-Y. Oh, D. Hwang, and T.-J. Lee, "Joint access control and resource allocation for concurrent and massive access of M2M devices," *IEEE Trans. Wireless Commun.*, vol. 14, no. 8, pp. 4182–4192, Aug. 2015.

[30] J. Liu, W. Zhou, and L. Song, "A novel congestion reduction scheme for massive machine-to-machine communication," *IEEE Access*, vol. 5, pp. 18765–18777, 2017.

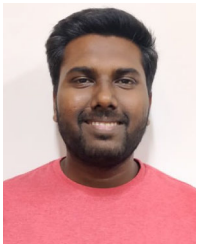
[31] M. Tavana, A. Rahmati, and V. Shah-Mansouri, "Congestion control with adaptive access class barring for LTE M2M overload using Kalman filters," *Comput. Netw.*, vol. 141, pp. 222–233, Aug. 2018.

[32] M. El Tanab and W. Hamouda, "Machine-to-machine communications with massive access: Congestion control," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3545–3557, Apr. 2019.

[33] A.-T.-H. Bui, C. T. Nguyen, T. C. Thang, and A. T. Pham, "A novel effective DQ-based access protocol with load estimation for massive M2M communications," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017, pp. 1–7.

[34] G.-Y. Lin, S.-R. Chang, and H.-Y. Wei, "Estimation and adaptation for bursty LTE random access," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2560–2577, Apr. 2016.
[35] M. Shirvanimoghaddam, M. Dohler, and S. J. Johnson, "Massive multiple access based on superposition raptor codes for cellular M2M communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 1, pp. 307–319, Jan. 2017.
[36] M. Kodialam, T. Nandagopal, and W. C. Lau, "Anonymous tracking using RFID tags," in *Proc. IEEE INFOCOM 26th IEEE Int. Conf. Comput. Commun.*, May 2007, pp. 1217–1225.
[37] Y. Zheng and M. Li, "PET: Probabilistic estimating tree for large-scale RFID estimation," *IEEE Trans. Mobile Comput.*, vol. 11, no. 11, pp. 1763–1774, Nov. 2012.
[38] Y. Zheng and M. Li, "ZOE: Fast cardinality estimation for large-scale RFID systems," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 908–916.
[39] W. Gong, K. Liu, X. Miao, and H. Liu, "Arbitrarily accurate approximation scheme for large-scale RFID cardinality estimation," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2014, pp. 477–485.
[40] X. Liu et al., "Fast tracking the population of key tags in large-scale anonymous RFID systems," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 278–291, Feb. 2017.
[41] X. Liu et al., "RFID estimation with blocker tags," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 224–237, Feb. 2017.
[42] S. V. Y., P. H. Prasad, R. Kumar, S. Kadam, and G. S. Kasbekar, "Rapid node cardinality estimation in heterogeneous machine-to-machine networks," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–7.
[43] S. Kadam, S. V. Yenduri, P. H. Prasad, R. Kumar, and G. S. Kasbekar, "Rapid node cardinality estimation in heterogeneous machine-to-machine networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1836–1850, Feb. 2021.
[44] S. Kadam and G. S. Kasbekar, "Node cardinality estimation using a mobile base station in a heterogeneous wireless network deployed over a large region," in *Proc. Int. Conf. Signal Process. Commun. (SPCOM)*, Jul. 2020, pp. 1–5.
[45] S. Kadam, K. S. Bhargao, and G. S. Kasbekar, "Node cardinality estimation in a heterogeneous wireless network deployed over a large region using a mobile base station," *J. Netw. Comput. Appl.*, vol. 221, Jan. 2024, Art. no. 103779.
[46] Q. Xiao et al., "Estimating cardinality of arbitrary expression of multiple tag sets in a distributed RFID system," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 748–762, Apr. 2019.
[47] Y. Zhang, S. Chen, Y. Zhou, O. O. Odegbile, and Y. Fang, "Efficient anonymous temporal-spatial joint estimation at category level over multiple tag sets with unreliable channels," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2174–2187, Oct. 2020.
[48] X. Liu et al., "Multi-category RFID estimation," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 264–277, Feb. 2017.
[49] S. Yang, S. Sanghavi, H. Rahmanian, J. Bakus, and S. V. N. Vishwanathan, "Toward understanding privileged features distillation in learning-to-rank," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 26658–26670.
[50] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[51] K. Markov and T. Matsui, "Robust speech recognition using generalized distillation framework," in *Proc. Interspeech*, Sep. 2016, pp. 2364–2368.
[52] Z. Gao et al., "Privileged modality distillation for vessel border detection in intracoronary imaging," *IEEE Trans. Med. Imag.*, vol. 39, no. 5, pp. 1524–1534, May 2020.
[53] W. Lee, J. Lee, D. Kim, and B. Ham, "Learning with privileged information for efficient image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, vol. 12369. Cham, Switzerland: Springer, Nov. 2020, pp. 465–482.
[54] M. Abbasi, A. Shahraki, J. Prieto, A. G. Arrieta, and J. M. Corchado, "Unleashing the potential of knowledge distillation for IoT traffic classification," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 221–239, 2024.
[55] P. Page. (2024). *Simulation Datasets*. [Online]. Available: https://rb.gy/fx2lz8
[56] S. Madden. (2004). *Intel Berkeley Research Lab Data*. [Online]. Available: https://db.csail.mit.edu/labdata/labdata.html
[57] M. Bacco, T. De Cola, G. Giambene, and A. Gotta, "TCP-based M2M traffic via random-access satellite links: Throughput estimation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 2, pp. 846–863, Apr. 2019.
[58] Z. Zhou, B. Chen, and H. Yu, "Understanding RFID counting protocols," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 312–327, Feb. 2016.
[59] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural Netw.*, vol. 22, nos. 5–6, pp. 544–557, Jul. 2009.
[60] D. Pechyony, R. Izmailov, A. Vashist, and V. Vapnik, "SMO-style algorithms for learning using privileged information," *Dmin*, vol. 10, pp. 235–241, Jul. 2010.
[61] V. Sharmanska, N. Quadrianto, and C. H. Lampert, "Learning to rank using privileged information," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 825–832.
[62] M. Lapin, M. Hein, and B. Schiele, "Learning using privileged information: SVM+ and weighted SVM," *Neural Netw.*, vol. 53, pp. 95–108, May 2014.
[63] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 2018, *arXiv:1802.05668*.
[64] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.
[65] J. Tang and K. Wang, "Ranking distillation: Learning compact ranking models with high performance for recommender system," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2289–2298.
[66] S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury, "Improving efficient neural ranking models with cross-architecture knowledge distillation," 2020, *arXiv:2010.02666*.
[67] S. Reddi, "RankDistil: Knowledge distillation for ranking," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 2368–2376.
[68] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1607–1616.
[69] Z. Qin et al., "Born again neural rankers," in *Proc. ICLR*, 2021, pp. 1–13.
[70] D. Lopez-Paz, L. Bottou, B. Schölkopf, and V. Vapnik, "Unifying distillation and privileged information," 2015, *arXiv:1511.03643*.
[71] N. C. Garcia, P. Morerio, and V. Murino, "Learning with privileged information via adversarial discriminative modality distillation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2581–2593, Oct. 2020.
[72] C. Xu et al., "Privileged features distillation at Taobao recommendations," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2590–2598.
[73] D. Pechyony and V. Vapnik, "On the theory of learning with privileged information," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010, pp. 1–11.
[74] C. Gong, X. Chang, M. Fang, and J. Yang, "Teaching semi-supervised classifier via generalized distillation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2156–2162.
[75] Y. Chen, X. Jin, J. Feng, and S. Yan, "Training group orthogonal neural networks with privileged information," 2017, *arXiv:1701.06772*.
[76] S. Wang, K. Zhang, L. Wu, H. Ma, R. Hong, and M. Wang, "Privileged graph distillation for cold start recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1187–1196.
[77] M. Collier, R. Jenatton, E. Kokiopoulou, and J. Berent, "Transfer and marginalize: Explaining away label noise with privileged information," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 4219–4237.
[78] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022.
[79] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Worcester, U.K.: PWS Publishing Co., 1997.

PRANAV S. PAGE received the B.Tech. and M.Tech. degrees in electrical engineering with a specialization in communications and signal processing (CSP) and a minor in computer science from IIT Bombay, in 2023. He is currently a Data Scientist/Engineer with Carnot Technologies, an agri-tech startup. At Carnot, he focuses on building analytics and geospatial intelligence solutions for farmers. His research interests include wireless communication, routing protocols, the IoT, and machine learning.



ANAND S. SIYOTE received the M.Tech. degree in industrial engineering and operations research from IIT Bombay. He is currently a Senior Research Engineer and a Data Scientist with TIH IoT, where he develops artificial intelligence/machine learning (AI/ML) solutions tailored to industry needs. His research interests include AI and ML, particularly in areas like computer vision, large language models (LLMs), and deep learning frameworks, such as TensorFlow and PyTorch. He is committed to advancing these fields through continued research and collaboration with industry experts.

VIVEK S. BORKAR (Life Fellow, IEEE) received the B.Tech. degree in EE from IIT Bombay in 1976, the M.S. degree in systems and control from Case Western Reserve University in 1977, and the Ph.D. degree in EECS from the University of California at Berkeley, in 1980. He has held regular positions with the TIFR Centre for Applicable Mathematics and Indian Institute of Science, Bengaluru, and Tata Institute of Fundamental Research and Indian Institute of Technology Bombay, Mumbai, from where he retired recently. He has held visiting positions with the Technical University of Twente, MIT, University of Maryland at College Park, University of California at Berkeley, and University of Illinois at Urbana–Champaign. His research interests are in stochastic optimization and control, theory and algorithms, and applications thereof, mainly to communications. He is a fellow of AMS, TWAS, and various science and engineering academies in India.

GAURAV S. KASBEKAR (Member, IEEE) received the B.Tech. degree in electrical engineering from IIT Bombay, Mumbai, India, in 2004, the M.Tech. degree in electronics design and technology (EDT) from Indian Institute of Science, Bengaluru, India, in 2006, and the Ph.D. degree from the University of Pennsylvania, Philadelphia, PA, USA, in 2011. He is currently an Associate Professor with the Department of Electrical Engineering, IIT Bombay. His research interests include communication networking and network security. He received the CEDT Design Medal for being adjudged the best master's student in EDT with IISc.