Adaptive Target-Condition Neural Network: DNN-Aided Load Balancing for Hybrid LiFi and WiFi Networks

Han Ji, Student Member, IEEE, Xiping Wu, Senior Member, IEEE, Qiang Wang, Student Member, IEEE, Stephen J. Redmond, Senior Member, IEEE, and Iman Tavakkolnia, Senior Member, IEEE

Abstract—Load balancing (LB) is a key challenge in hybrid light fidelity (LiFi) and wireless fidelity (WiFi) networks (HLWNets), due to the nature of heterogeneous access points (APs). Machine learning has the potential to provide a complexity-friendly LB solution with near-optimal network performance, at the cost of a non-trivial training process. The state-of-the-art learning-aided LB methods require retraining when the network environment (particularly the user number) changes, significantly limiting their practicability. In this paper a novel deep neural network (DNN) structure, named adaptive target-condition neural network (A-TCNN), is proposed to tackle the LB issue for a varying number of users, without the need for retraining. Unlike the existing LB methods conducting AP selection for all users together, the new method performs AP selection for a single target user, upon the condition of other users. Also, A-TCNN involves an adaptive mechanism which maps any smaller number of users to a preset number by splitting the users’ data rate requirements, without affecting the AP selection result for the target user. Once trained, A-TCNN can be used for any user numbers not exceeding the maximum user number that the network can support. Results show that apart from the adaptiveness to a varying user number, A-TCNN provides a higher network throughput (up to 45%) than the conventional DNN in most cases, especially for a larger scale of network. In terms of computational complexity, A-TCNN can achieve a sub-millisecond level runtime, which is 2 orders of magnitude lower than fuzzy logic and 3 orders of magnitude lower than game theory.

Index Terms—Light fidelity (LiFi), visible light communications (VLC), hybrid networks, load balancing, deep neural network (DNN), machine learning.

Manuscript received 6 March 2023; revised 28 July 2023 and 8 October 2023; accepted 21 November 2023. Date of publication 20 December 2023; date of current version 12 July 2024. The work of Han Ji and Qiang Wang was supported by the China Scholarship Council under Grant 20216620012 and Grant 202006540003. The work of Xiping Wu was supported by the Royal Irish Academy (RIA) Charlemont Grant Award. The work of Stephen J. Redmond was supported in part by the Science Foundation Ireland President of Ireland Future Research Leaders Award 17/FRL/4832. The associate editor coordinating the review of this article and approving it for publication was P. Hu. (Corresponding author: Xiping Wu.)

Han Ji, Xiping Wu, Qiang Wang, and Stephen J. Redmond are with the School of Electrical and Electronic Engineering, University College Dublin, Dublin 4, D04 V18 Ireland (e-mail: han.ji@ucdconnect.ie; xiping.wu@ucd.ie; qiang.wang@ucdconnect.ie; stephen.redmond@ucd.ie).

Iman Tavakkolnia is with the Electronic and Electrical Engineering Department, University of Strathclyde, G1 1XQ Glasgow, U.K. (e-mail: i.tavakkolnia@strath.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2023.3339503.

Digital Object Identifier 10.1109/TWC.2023.3339503

I. INTRODUCTION

The Cisco Annual Internet Report (2018-2023) forecasts a four-fold increase in the number of wireless fidelity (WiFi) hotspots between 2018 and 2023, reaching 628 million by the end of 2023 [1]. The dense deployment of WiFi hotspots would lead to intense competitions for available channels, due to the limited radio-frequency (RF) spectrum. This drives research on light fidelity (LiFi) [2], which operates in a way similar to WiFi but explores the extremely wide visible light spectrum (~300 THz). In contrast to WiFi, LiFi offers several prominent advantages including license-free, availability in RF-restricted areas, and high-level physical layer security. Upon the existing light infrastructure, LiFi can also render a simultaneous provision of illumination and communication, boosting the energy efficiency. Recent experimental work shows that LiFi is capable of providing a link data rate over 24 Gbps [3].

However, LiFi offers a relatively small coverage area with a single access point (AP), usually 2–3 metres in diameter. In addition, LiFi is susceptible to channel blockage caused by opaque objects such as human bodies and furniture [4]. Combining the respective advantages of LiFi and WiFi, the hybrid LiFi and WiFi networks (HLWNets) have been gaining significant research momentum in recent years [5]. While such hybrid networks can greatly improve network capacity over stand-alone LiFi or WiFi networks [6], AP selection becomes tricky due to the fact that WiFi APs have a larger coverage area but a lower system capacity than LiFi APs. While the signal strength strategy (SSS)1 is widely used in homogeneous networks, it is much less effective in HLWNets since user equipment (UEs) would be more likely to choose WiFi over LiFi, making the WiFi system prone to overload. Consequently, load balancing (LB) becomes an essential and significant issue in HLWNets.

A. Related Work

The conventional LB methods for HLWNets can be classified into three categories: i) centralised optimisation, ii) iterative methods, and iii) rule-based decision-making. The centralised optimisation methods, e.g. [7], formulate the LB problem as a single objective function with a certain fairness

1The signal strength strategy selects the AP that provides the highest received signal power to the user.
A summary of the load balancing methods for HLWNets

| Ref. | Method                          | Optimality | Complexity | Fairness | LiFi AP density | Remark                                      |
|------|--------------------------------|------------|------------|----------|-----------------|---------------------------------------------|
| [7]  | Centralised optimisation        | Optimal    | Extremely high | PF       | 16              | Computational complexity exponentially increases with the UE number |
| [8]  | Iterative method: Distributed optimisation | Near-optim. | High       | PF       | 16              | Computational complexity exhibits quadratic growth with the UE number |
| [9]  | Iterative method: Game theory    |            |            | MF, PF, EPF | 4, 16          |                                             |
| [10] | Rule-based method: Fuzzy logic (FL) | Sub-optim. | Low        | PF       | 16              | Need dedicated design of fuzzy rules for a specific network environment |
| [11] | Rule-based method: Mixed FL and optim. | Near-optim. | Medium    | PF       | 4, 16          |                                             |
| [12] | Learning-aided method: DNN       | Sub-optim. | Ultra low  | N/A      | 4               | Need a dedicated DNN structure for each specific UE number |
| [13] | Learning-aided method: Q-learning (off-policy) | Near-optim. | Ultra low  | N/A      | 2               | Need retraining when the UE number changes |
| [14] | Learning-aided method: Policy-based RL |            |            | PF       | 4               |                                             |
| This work | A-TCNN                           |            |            | PF       | 4, 9            | Handle a varying UE number without the need for retraining |

B. Motivation and Contribution

The existing learning-aided LB methods [12], [13], [14] lack adaptiveness to a varying number of UEs, which happens in practice when the existing UEs leave or new UEs join the network. The DNN-based method in [12] needs a dedicated DNN model in each case of the UE number, due to the fixed numbers of input and output nodes. As a result, the method lacks generality when dealing with different UE numbers. The work [12] only presents several cases of the UE number up to 9. The Q-learning LB algorithm in [13] can suit different UE numbers, but the learning process is expensive for large-scale networks. Accordingly, the study [13] only considers 2 LiFi APs with a UE number up to 9. Relative to the Q-learning algorithm, the policy-based RL method is scalable to larger network sizes. However, this method would require a retraining process to update its optimal policy when the UE number varies, greatly limiting its practicability. In [14], the RL method is exemplified with only two cases of the UE number, 5 and 10. In summary, the existing learning-aided LB methods require either i) a dedicated model for each UE number, or ii) a general model but with a retraining process to suit different UE numbers.

In this paper we propose a novel learning-aided LB method to handle a varying number of UEs without the need for retraining. The main contributions are:

- A novel DNN model is proposed to tackle LB in HLWNets. While the conventional DNN model carries out AP selection for all UEs together, the new model conducts AP selection for a single target UE upon the conditions of other UEs. The proposed model is thus termed target-condition neural network (TCNN), which provides two key benefits over the conventional DNN: i) TCNN can achieve a substantially higher accuracy especially for large UE numbers, due to the reduced size

2The LiFi AP density refers to the number of LiFi APs per coverage area of one WiFi AP.
of output; and ii) TCNN makes it feasible to adapt to different UE numbers, thanks to the unique structure of target-condition. See details in the following points.

- An adaptive mechanism is designed to enable TCNN to cope with a varying number of UEs, by mapping any smaller UE number to a preset number. Specifically, mirroring UEs are created which have the same location as the original UE and equally split its required data rate. As the traffic demands imposed by the condition UEs remain the same, the mapping process does not affect the AP selection result for the target UE. Combining the adaptive mechanism and TCNN, A-TCNN is constructed which can carry out LB for a varying number of UEs without the need for retraining.

- Ablation experiments are implemented to validate the effectiveness and necessity of the key components in the A-TCNN model, including the target-condition structure, adaptive mechanism, and mixture training. Results show that with the same sample dataset, A-TCNN can improve the AP selection accuracy by up to 14% against the conventional DNN. In addition, the ablation study demonstrates that unlike DNN, A-TCNN performs well for the UE numbers that are not involved in the training dataset.

- Simulations are carried out to verify the performance of A-TCNN in two cases of LiFi AP density (4 and 9) and with various UE numbers up to 50. Results show that A-TCNN approaches the network throughput of the sample dataset with a gap of 5% to 8% on average. Against the conventional DNN, A-TCNN can improve the network throughput by up to 45%. Also, the computational complexity of A-TCNN exhibits a very slight increase with the network scale and the UE number. Compared to FL and GT, A-TCNN can reduce the runtime by 2 and 3 orders of magnitude, respectively.

The remainder of this paper is organised as follows. The system model is presented in Section II. The A-TCNN model is proposed in Section III. The ablation study of the proposed learning model is carried out in Section IV, and simulation results are presented in Section V. Finally, conclusions are drawn in Section VI.

**Notations:** In the paper, scalar variables are represented by italic lowercase letters, whereas vectors and matrices are denoted by bold lowercase letters and bold uppercase letters, respectively. The notations \(|cdot|\) and \(||cdot||\) stand for the absolute value and the Frobenius norm of a variable; \(E[cdot]\) and \(\text{Var}[cdot]\) are the expectation and the variance of a random variable. Let \([cdot]\) and \([^cdot]\) denote the floor and ceiling functions, respectively. The operator \(\mod(a,b)\) returns the remainder of \(a\) divided by \(b\). The notation \(\mathbb{R}^n\) represents the real coordinate space with a dimension of \(n\).

### II. System Model

The system model of HLWNet is introduced in this section, including: i) network architecture, ii) channel models and link capacity of LiFi and WiFi, iii) network performance metrics, and iv) load balancing problem formulation.

#### A. Network Architecture

Fig. 1 shows an indoor HLWNet that consists of 1 WiFi AP and a number of LiFi APs. The WiFi AP is deployed in the centre of the room, providing coverage to the entire room. The LiFi APs are arranged in a grid, and each AP is embedded into a ceiling lamp, covering a confined area. Frequency reuse with a reuse factor of 4 is adopted to avoid inter-cell interference (ICI) among adjacent LiFi APs [17]. The ICI among further APs is insignificant and can be neglected. The UEs are randomly located on the ground with a uniform distribution. The data rate requirements of the UEs are considered to be independent and identically distributed random variables. Without loss of generality, it is assumed that these variables follow a Gamma distribution with shape parameter \(\alpha\) and inverse scale parameter \(\beta\) [18]. Let \(R_j\) denote the data rate required by the \(j\)-th UE. The expected value of the average data rate requirement is \(R = \alpha/\beta\). Each UE is served by one AP, either WiFi or LiFi, while each AP can serve multiple UEs via time-division multiple access.

#### B. Channel Models and Link Capacity

Quasi-static channels are considered. Regarding WiFi, the log-distance path loss model in [10, eq. (7)] is adopted. As for LiFi, the channel is comprised of the line-of-sight (LoS) and first-order non-line-of-sight (NLoS) paths, as illustrated in Fig. 1. The corresponding channel expressions can be found in [19, eq. (10) and eq. (12)]. The capacity of WiFi is bounded by Shannon capacity. With respect to LiFi, a tighter bound can be found in [20] due to the non-negative real signals in LiFi. Let \(\gamma_{i,j}\) denote the signal-to-noise ratio (SNR) of the link between AP \(i\) and UE \(j\). The link capacity \(C_{i,j}\) can be expressed as follows:

\[
C_{i,j} = \begin{cases} 
\frac{B_i}{2} \log_2 \left( 1 + \frac{e}{2\pi \gamma_{i,j}} \right), & \text{for LiFi} \\
\frac{B_i}{\log_2(1 + \gamma_{i,j})}, & \text{for WiFi}
\end{cases}
\]

where \(e\) is the Euler’s number and \(B_i\) denotes the bandwidth of AP \(i\).

#### C. Network Performance Metrics

Two performance metrics are considered: throughput and UE fairness. The overall network throughput is denoted by \(\Gamma\), which can be calculated by:

\[
\Gamma = \sum_{j \in U} \sum_{i \in S} \rho_{i,j} \chi_{i,j} C_{i,j},
\]

where \(S = \{i|i \in [1,2,...,N_a]\}\) is the set of APs and \(U = \{j|j \in [1,2,...,N_u]\}\) is the set of UEs, with \(N_a\) and \(N_u\) the number of APs and UEs, respectively.
denoting the number of APs and the number of UEs, respectively; $\chi_{i,j}$ is a binary variable: $\chi_{i,j} = 1$ indicates that a connection exists between AP $i$ and UE $j$, and otherwise $\chi_{i,j} = 0$; $\rho_{i,j} \in [0, 1]$ denotes the portion of time resource that is allocated by AP $i$ to UE $j$.

The fairness among UEs is measured by Jain’s fairness index, which is given by [10, eq. (14)]:

$$\xi = \frac{\left(\sum_{j \in U} S_j\right)^2}{Nu \sum_{j \in U} S_j},$$

where $S_j$ is the satisfaction degree of UE $j$. When the UE’s achievable throughput $\Gamma_j$ is larger or equal to its data rate requirement $R_j$, we have $S_j = 1$; otherwise, $S_j$ is the ratio between $\Gamma_j$ and $R_j$. Thus, $S_j$ can be expressed as follows:

$$S_j = \min \left\{ \sum_{i \in S} \rho_{i,j} \chi_{i,j} C_{i,j}, 1 \right\}.$$  \hspace{1cm} (4)

**D. Load Balancing Problem Formulation**

The centralised optimisation method in [7] aims to maximise the overall network throughput with proportional fairness. However, this scheme does not take into account the distinctive data rate requirements among the UEs, depending on the applications in use. For example, voice over Internet Protocol (VoIP) typically needs a data rate of 85-100 kbps, while streaming a 4K video requires an Internet speed of 25 Mbps [21]. As a result, the objective function in [7] would allocate resources more than needed to some UEs, causing waste of resources. In this paper, we focus on maximising the sum of all UEs’ satisfaction with proportional fairness. Accordingly, the objective function can be formulated as:

$$\max_{\chi, \rho} \sum_{j \in U} \log (S_j)$$

s.t. $\sum_{i \in S} \chi_{i,j} = 1$, $\forall j \in U$;  
$$\sum_{i \in U} \chi_{i,j} \rho_{i,j} \leq 1$, $\forall i \in S$;  
$$0 \leq \rho_{i,j} \leq 1$, $\forall i \in S, j \in U$;  
$$\chi_{i,j} \in \{0, 1\}$, $\forall i \in S, j \in U$,

where $\chi$ is the set of all $\chi_{i,j}$, and $\rho$ is the set of all $\rho_{i,j}$.

The above is a mixed-integer nonlinear programming (MINLP) problem, which can be solved by the OPTI toolbox [22]. However, it would take a prohibitive amount of time to acquire the globally optimal solution for large numbers of APs and UEs. For this reason, centralised optimisation is not considered as a benchmark. Alternatively, iterative algorithms are taken into account, which split the optimisation process related to $\chi$ and $\rho$. Given $\chi$, the optimisation of $\rho$ can be formulated as follows:

$$\max_{\rho} \sum_{j \in U} \log (S_j)$$

s.t. $\sum_{j \in U} \chi_{i,j} \rho_{i,j} \leq 1$, $\forall i \in S$;  
$$0 \leq \rho_{i,j} \leq 1$, $\forall i \in S, j \in U$.

In this work, two iterative algorithms are considered upon the partial optimal solution of $\rho$ in (6). The first method optimises the AP selection result for a specific UE at a time:

$$\max_{\chi_j} \sum_{j \in U} \log (S_j)$$

s.t. $\sum_{i \in U} \chi_{i,j} = 1$, $\forall j \in U$;  
$$\chi_{i,j} \in \{0, 1\}$, $\forall i \in S, j \in U$,  \hspace{1cm} (7)

where $\chi_j$ is the set of all $\chi_{i,j}$ related to UE $j$. For each possible choice of $\chi_j$, $\rho$ is computed by (6). The above process is repeated for each UE subsequently, until $\chi$ stops changing for all UEs. This method is referred to as distributed optimisation (DO) in the rest of the paper. The other method is GT, which is similar to [8] except its objective function of resource allocation is replaced by (6) to ensure a fair comparison. This method is also employed to collect sample data for training the proposed learning model.

### III. Adaptive Target-Condition Neural Network

The A-TCNN model consists of two key components: i) target-condition neural network (TCNN), and ii) adaptive mechanism. The relationship between the two components is illustrated in Fig. 2, which depicts the block diagram of the A-TCNN model. In this section, the two components are introduced separately, followed by the procedures of training and test. At last, the convergence and optimality of A-TCNN are analysed.

#### A. Target-Condition Neural Network

As mentioned previously, the conventional DNN model carries out AP selection for all UEs together. Such a model cannot support mapping a smaller UE number to a larger number, since this might generate conflicting AP selection results for the same UEs. To break this limitation, the DNN model is designed to determine AP selection for a single UE, which is referred to as target UE, while other UEs are deemed as the condition and thus called condition UEs. As shown in Fig. 2, the target UE and condition UEs are processed through separate neural networks, of which the outputs are then combined in a third neural network to deliver the AP selection result for the target UE. In this way, mapping a smaller UE number to a larger number becomes feasible, without affecting the AP selection result for the target UE. The details of the three neural networks are given below.

1) Target Neural Network: Assume UE $k$ to be the target UE, where $k \in \{1, 2, \ldots, N_u\}$. Let $\gamma_k = [\gamma_{1,k}, \gamma_{2,k}, \ldots, \gamma_{N_a,k}]$ denote the set of SNR values between the target UE and each AP. The input is denoted by $x_k = [\gamma_{1,k}, R_k]$, which contains two pieces of information, $\gamma_k$ and $R_k$. The input is fed to a fully-connected (FC) layer with 6 neurons, while the weight matrix and the bias vector are denoted by $W_{i,k} \in \mathbb{R}^{(N_a+1) \times 6}$ and $b_i \in \mathbb{R}^6$, respectively. The output of the target neural network can be expressed as follows:

$$u_k = W_i x_k + b_i.$$  \hspace{1cm} (8)
2) **Condition Neural Network:** The input of the condition neural network contains the information related to all UEs including the target UE, as redundant information is proved to enhance the accuracy of neural networks [23]. The input is denoted by \( x_C = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_M] \), where \( \tilde{x}_m \neq k \) denotes the set of information for condition UE \( m \), and \( M \) is a preset number of UEs. For a stand-alone TCNN model without the adaptive mechanism, \( M \) is equivalent to the actual UE number \( N_u \). With the adaptive mechanism, \( M \) can be set as the maximum UE number that the HLWNet can support. To reduce the dimension of the input while extracting its features, two FC layers with 64 and 6 neurons are employed. The corresponding weight matrices and bias vectors are \( W_2 \in \mathbb{R}^{M(N_u+1) \times 64} \), \( b_2 \in \mathbb{R}^{64} \), \( W_3 \in \mathbb{R}^{64 \times 6} \), and \( b_3 \in \mathbb{R}^6 \), respectively. A batch normalisation (BN) layer is added after each FC layer to avoid the vanishing gradient problem [24]. The BN operation is formulated by:

\[
\text{f}_\text{BN}(x) = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} + \mu + \nu, \tag{9}
\]

where \( \epsilon \), which is usually set to be 0.001, is a small constant to avoid zero denominator in (9); \( \mu \) and \( \nu \) are learnable parameters to achieve an accurate normalisation of each batch. Here the initial values of \( \mu \) and \( \nu \) are set to be 1 and 0. The activation function adopted is a rectified linear unit (ReLU), which is defined as \( f\text{ReLU}(x) = \max(0, x) \). The output of the condition neural network can be expressed as follows:

\[
u_C = f\text{ReLU}\left\{ f\text{BN}\left[ W_3 \{ f\text{ReLU}\left[ f\text{BN}\left( W_2 x_C + b_2 \right) \right] + b_3 \} \right]\right\} + b_4 \in \mathbb{R}^{N_u}. \tag{10}
\]

Then softmax, which is a probability-based activation function, is used to generate a probability vector \( p_k \).

\[
p_k = f\text{softmax}\left[ W_4 \left( \text{Cat}(u_k, u_C) + b_4 \right) \right]. \tag{11}
\]

A soft decision maker is adopted to select the AP that has the highest value of probability in \( p_k \) to be the host AP for the target UE. Let \( i^* \) denote the selected AP:

\[
i^* = \max_{i \in \mathbb{B}} p_k. \tag{12}
\]

The final output \( y_k = [y_1, y_2, \ldots, y_{N_u}] \) is a vector that contains \( N_u \) binary elements, with each element indicating the AP selection result related to the corresponding AP. Specifically, the \( i^* \)-th element of the output is \( y_k(i^*) = 1 \), signifying that AP \( i^* \) is selected for the target UE \( k \), while the other elements are all zero.

### B. Adaptive Mechanism

With TCNN alone, it can only cope with a fixed number of UEs, due to the fixed number of input nodes in condition neural network. An adaptive mechanism is thus designed to enable the learning model to adapt to different UE numbers. In general, the adaptive mechanism can map \( N_u \) UEs to \( M \) UEs for any \( N_u \leq M \). The information of all original UEs \( x_O = [x_1, x_2, \ldots, x_k, \ldots, x_{N_u}] \) are fed to a mapping operator, which generates mirroring UEs for each condition UE. The mirroring UEs have the same values of SNR as the original UE and equally split its required data rate. Hence, the total required data rate of the mirroring UEs is the same as that of the original UE. In the view of the target UE, the mirroring UEs have the same impact as the original condition UEs. Hence, the mapping process does not affect the optimal AP selection result for the target UE.
Algorithm 1 Adaptive Mechanism

Input: \( x_C = [x_1, x_2, \ldots, x_k, \ldots, x_{N_u}] \), \( M \)

Output: \( x_C = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_k, \ldots, \tilde{x}_M] \)

1. \( N_j \leftarrow \text{eq. (13)} \) \( \forall j \in [1, 2, \ldots, N_u], j \neq k; \)
2. Initialise: \( j \leftarrow 1, n \leftarrow 1; \)
3. for \( m = 1 \) to \( M \) do
   1. if \( m = k \) then
      1. \( x_C(m) \leftarrow x_k; \)
   2. else
      1. if \( n > N_j \) then
         1. \( j \leftarrow \text{next } j \) (excluding \( j = k \));
         2. \( n \leftarrow 1; \)
      2. \( x_C(m) \leftarrow [\gamma_j, R_j/N_j]; \)
      3. \( n \leftarrow n + 1; \)
   3. end
4. end

The pseudocode of the adaptive mechanism algorithm is given in Alg. 1, with input \( x_C \) and output \( x_C \). The entire output \( x_C \) is then fed to the condition neural network, while the information of the target UE (i.e., \( x_k \)) is solely fed to the target neural network.

C. Training Process

In this paper, the sample dataset is yielded through the GT method in [8]. Note that the training sets of TCNN and A-TCNN are constructed in different ways. For TCNN, the training set consists of sample data for a fixed UE number. This is referred to as fixed training. As for A-TCNN, the training set is a mixture of the samples related to different UE numbers, and this is referred to as mixture training. The two types of training process are introduced below.

1) Fixed Training: In this case, the training set contains \( N \) batches. Each batch is a matrix containing \( T \) columns, and each column is a sample of \( x_C \). Therefore, the total number of samples in this training set is \( N \times T \). It is worth noting that the two pieces of information contained in \( x_C \) (i.e., \( \gamma_j \) and \( R_j \)) comply with different probability distributions. According to the characteristics of probability distributions, linear normalisation is used for \( \gamma_j \), whereas logarithmic normalisation is chosen for \( R_j \). After the normalisation process, the batches are grouped together and fed into TCNN. The ground-truth labels are denoted by \( y_k \), which is a vector similar to \( y_k \) but contains the AP selection results obtained by GT. Since only one element in \( y_k \) is 1 while others are all 0, the categorical variables can be naturally represented by a one-hot encoding.

Stochastic learning is adopted instead of batch learning as it can speed up learning, particularly for large datasets with redundancy [25]. In addition, cross-entropy is considered as the loss function, since it is ideal for multi-label classification tasks [26]. The cross-entropy loss function can be formulated as follows:

\[
L_{CE}(\theta) = -\frac{1}{T} \sum_{k=1}^{T} \sum_{i=1}^{N_u} y_k(i) \log(p_k(i)),
\]

where \( p_k \) is the probability vector generated by the softmax function in (11), and the values of \( p_k \) are determined by \( \theta \), which consists of all the weight matrices and bias vectors in the neural networks. Adaptive moment estimation (Adam) [27] is employed to train \( \theta \) through \( \theta \leftarrow \theta - \eta \nabla L_{CE}(\theta) \), where \( \eta \) is the learning rate, and \( \nabla L_{CE}(\theta) \) stands for the gradient of the loss function with respect to \( \theta \). The values of training-related parameters are summarised in Section V-A.

2) Mixture Training: With the adaptive mechanism, the samples of different UE numbers can be mixed together, since they are mapped to the same preset number \( M \). In this paper, 10 different UE numbers (from 5 to 50 with a step of 5) are chosen to compose the training set for A-TCNN. In each case of the UE number, the training set contains the same number of batches as fixed training. Therefore, the total number of batches for training A-TCNN is \( 10 \times N \). This setup provides a fair comparison between A-TCNN and TCNN, since TCNN would need \( 10 \times N \) batches to train 10 different models. The remaining training process is the same as described in the above subsection.

D. Validation and Test Process

The validation set contains 10 batches, with each batch for one case of UE number. This size of the validation set is sufficient for fine-tuning the trained model after each epoch. In the validation process, the metric loss in (15) is evaluated which can verify whether the learning model is overfit. Specifically, overfitting occurs when the training loss is substantially lower than the validation loss.

The test process is carried out in two aspects: i) trained model, and ii) wireless network. In the aspect of the trained model, the metric accuracy is measured, which is defined as the ratio between the number of UEs that acquire the same AP selection results as the ground-truth labels and the total number of UEs in the test. In the aspect of wireless network, achievable throughput is a direct metric to measure the network performance. Here we focus on evaluating the throughput gap between the trained model and the ground-truth label.

E. Analysis of Convergence and Optimality

Fig. 3 presents the loss and accuracy of A-TCNN as a function of the number of epochs. Two cases of HLVW nets are taken into account: Case I (4 LiFi APs with 1 WiFi AP) and Case II (9 LiFi APs with 1 WiFi AP). It can be seen that as the number of epochs increases, the training loss decreases gradually and approaches a steady level within 10 epochs in
TABLE II
METHODS COMPARED IN ABATION STUDY

| Method       | Description                        | Input size                  | Output size          |
|--------------|------------------------------------|-----------------------------|----------------------|
| DNN          | Acquire AP selection for all UEs together | \((N_a + 1)N_u\)            | \(N_aN_u\)          |
| TCNN         | No adaptive mechanism              | Target: \((N_a + 1)\); Condition: \((N_a + 1)N_u\) | \(N_a\)             |
| A-TCNN*\@\(N_u\) | A-TCNN trained with a fixed \(N_u\) | Target: \((N_a + 1)\); Condition: \((N_a + 1)M\) | \(N_a\)             |
| A-TCNN       | A-TCNN trained with a mixture of different \(N_u\) |                          |                      |

![Fig. 3. Loss and accuracy of A-TCNN.](image)

![Fig. 4. Achievable throughput of A-TCNN.](image)

...both cases. Also, the validation loss exhibits a close match to the training loss, indicating that A-TCNN is well trained without overfitting. Meanwhile, the test accuracy increases from 20% and saturates at around 72% after 6 epochs for Case I. A similar trend of test accuracy is found in Case II, but with a slightly longer period of convergence (8 epochs). The value of accuracy in Case II, which increases from 11% and saturates at 66%, is also marginally smaller than Case I. These results match the fact that without training (i.e., when the number of epochs is 0) the host AP is randomly selected, and the corresponding values of accuracy are 1/5 for Case I and 1/10 for Case II.

IV. ABATION STUDY

In this section, an ablation study is conducted to verify the effectiveness of A-TCNN by separately removing three key components: target-condition, adaptive mechanism, and mixture training. The study is exemplified with Case I (4 LiFi APs and 1 WiFi AP). The methods involved in the ablation experiments are listed in Table II.

A. Removal of Target-Condition

The conventional DNN model gathers the information of all UEs and makes the decision on AP selection for all UEs together. This method is referred to as DNN for simplicity. Compared with TCNN, the output size of DNN is \(N_u\) times larger, making it difficult to provide a high accuracy. To evaluate the effectiveness of the target-condition structure, the accuracy of TCNN is compared with DNN in Fig. 5(a). To ensure a fair comparison, DNN has the same neural network setup as TCNN, with details discussed in Section V-A. It is observed that DNN renders a lower accuracy than TCNN, especially for a larger number of UEs. When \(N_u = 10\), for example, TCNN achieves an accuracy 5% higher than DNN. This gap increases to 14% when \(N_u = 50\). This proves that the target-condition structure is beneficial for improving the accuracy of DNN.

B. Removal of Adaptive Mechanism

The adaptive mechanism is a key component to make the proposed learning model adapt to a varying number of UEs. The ablation study on this component is focused on investigating its impact on the accuracy of TCNN. To make a fair comparison, the same training set is used for both A-TCNN and TCNN. In other words, A-TCNN is trained with the sample data of a fixed UE number, and this method is referred to as A-TCNN*. As shown in Fig. 5(b), A-TCNN* exhibits almost the same accuracy as TCNN for various UE numbers. This signifies that the adaptive mechanism does not degrade the accuracy of TCNN.
C. Removal of Mixture Training

In Fig. 5(c), the effect of mixture training is studied by comparing the accuracy of A-TCNN with that of A-TCNN*. Three outcomes are observed: i) mixture training does not degrade the accuracy of A-TCNN for the trained UE numbers. There are three trained UE numbers in Fig. 5(c): 10, 30, and 50. When \( N_u = 10 \) and 30, A-TCNN achieves an accuracy close to A-TCNN*@10 and A-TCNN*@30. As for \( N_u = 50 \), A-TCNN acquires an accuracy 7% higher than TCNN*@50, due to the diverse sample set offered by mixture training; ii) mixture training enables A-TCNN to perform well for various trained UE numbers, whereas A-TCNN* can only work properly for a specific trained UE. For instance, the accuracy of A-TCNN*@30 decreases from 66% to 45% as \( N_u \) decreases from 30 to 10. Meanwhile, the accuracy of A-TCNN increases from 68% to 80%, delivering an accuracy improvement of 35% against A-TCNN*@30 when there are 10 UEs. The reason for this trend is that A-TCNN*@30 is dedicatedly trained for \( N_u = 30 \); and iii) mixture training allows A-TCNN to operate well for the untrained UE numbers. Two untrained UE numbers (24 and 44) are presented in Fig. 5(c). Though the training set does not include any sample data of these UE numbers, the trained A-TCNN can still achieve a rational accuracy. This is attributed to the generality of A-TCNN that is yielded when the samples of different UE numbers are mixed for training.

V. SIMULATION RESULTS

In this section, simulations are carried out to evaluate the performance of the proposed learning model A-TCNN, in terms of both wireless network performance and computational complexity. Five baseline methods are taken into account, including SSS, GT, DO, FL, and DNN. Two scales of HLWNets are considered: Case I (4 LiFi APs and 1 WiFi AP) and Case II (9 LiFi APs and 1 WiFi AP).

A. Simulation Setup

1) Model Training: The sample dataset is collected by implementing GT in MATLAB2021a, whereas the training and test of A-TCNN are programmed in Python3, on a desktop computer with an Intel Core i5-10500@3.1GHz processor. The relevant codes are available in [28]. To guarantee a fair comparison, the DNN model is designed to be a fully-connected neural network with the same activation function and loss function as A-TCNN. The DNN model for Case I consists of two FC layers with 128 and 64 neurons, respectively. As for Case II, the DNN model contains three FC layers, with 256, 64, and 128 neurons, respectively. Also, the same training set is provided to both A-TCNN and DNN. The training set for A-TCNN consists of 10 subsets, with each subset containing 100 batches for a specific UE number. As for DNN, each subset of the same batches is used to train a dedicated model for the corresponding UE number. In total, 10 DNN models and 1 A-TCNN model are trained in each case of HLWNet. Other parameters related to training are listed in Table III.

2) Model Evaluation: Monte Carlo simulations are conducted in MATLAB2021a to evaluate the performance of the proposed learning model in HLWNets. Two room sizes are considered: 5m \( \times \) 5m for Case I and 9m \( \times \) 9m for Case II. To fit the LiFi APs, the room area is divided into 4 and 9 square cells for Case I and Case II, respectively. The LiFi APs are placed at the centre of each cell on the ceiling. A single WiFi AP is deployed at the centre of the
room area, with a height of 0.5m from the ground. To reduce the computational complexity of GT and DO in Case II, only the closest four LiFi APs (in addition to the WiFi AP) are engaged in the process of AP selection for any given UE, since further LiFi APs provide significantly lower values of SNR. Other simulation parameters are given in Table III.

B. Wireless Network Performance

To comprehensively evaluate the performance of A-TCNN in HLWNets, the achievable throughput and UE fairness are measured with the impact of different UE numbers and different data rate requirements, respectively.

1) Impact of UE Numbers: In Fig. 6, the achievable throughput is shown as a function of the UE number, with a fixed $\bar{R} = 100$ Mbps for Case I and $\bar{R} = 200$ Mbps for Case II. As shown, the two iterative methods (i.e., GT and DO) outperform other methods in most situations, while FL provides the highest throughput for larger UE numbers, with a marginal improvement over the iterative methods. This is because with a large UE number, the iterative methods tend to achieve the Nash equilibrium before reaching the global optimum. Nevertheless, the iterative methods outmatch FL in terms of network throughput in general. When $N_u$ increases, the throughput gap between A-TCNN and GT first increases and then reduces, as explained in Section III-E. In contrast, the gap between DNN and GT keeps increasing, particularly for a larger value of $N_u$. In Case II, for example, the throughput of DNN increases from 616 Mbps to 1675 Mbps when $N_u$ increases from 5 to 20, but it decreases to 1510 Mbps when $N_u$ further increases to 50. Meanwhile, the throughput of A-TCNN increases from 603 Mbps to 2185 Mbps, providing an improvement of 45% against DNN when $N_u = 50$. This is because the output size of DNN is $N_u$ times larger than that of A-TCNN. Thus, when $N_u$ increases, the accuracy of DNN decreases more significantly than that of A-TCNN, as previously shown in Fig. 5(a).

Fig. 7 presents the UE fairness as a function of the UE number in Case II with a fixed $\bar{R} = 200$ Mbps. As can be observed, the DO method has the highest UE fairness, followed by the GT and FL methods. The fairness gap among the three methods is marginal, with the largest gap less than 5%. For A-TCNN, it achieves a UE fairness slightly lower than FL, with a gap within 3%. On the contrary, the UE fairness of DNN decreases drastically when $N_u$ increases above 20. When $N_u > 38$, DNN renders a UE fairness even lower than SSS. This corresponds with the throughput trend of DNN in Fig. 6, due to the fact that the accuracy of DNN deteriorates as $N_u$ increases. As a result, A-TCNN provides a much better UE fairness than DNN for larger UE numbers.

When $N_u = 50$, for instance, A-TCNN obtains a UE fairness 18% higher than DNN. A similar but less pronounced trend is found in Case I.

2) Impact of Data Rate Requirements: Fig. 8 presents the achievable throughput as a function of the average data rate requirement $\bar{R}$, while the UE number is fixed to be 25. As shown, the throughput gap between A-TCNN and GT increases towards a saturation when $\bar{R}$ increases. In Case II, for example, this gap grows from 0 to 12.5% as $\bar{R}$ increases from 10 Mbps to 100 Mbps. When $\bar{R}$ further increases to 200 Mbps, the gap remains almost the same with a very slight increase of 1%. In addition, it is found that A-TCNN achieves a higher throughput than DNN in Case II, whereas the opposite situation occurs in Case I. When $\bar{R} = 150$ Mbps, for example, the throughput achieved by A-TCNN is 20% higher than that of DNN in Case II but 10% lower in Case I. This will be discussed in the following paragraph, in association with the outcomes observed from Fig. 9.

Fig. 9 presents the throughput versus the average data rate for $N_u = 50$. Similar to Fig. 8,
the throughput gap between A-TCCN and GT increases towards a saturation when $R$ increases. However, the throughput gap here is much smaller than that in Fig. 8. When $R = 200$ Mbps in Case II, for example, the gap between A-TCCN and GT is merely 2%, in contrast to the gap of 13.5% in Fig. 8. This is due to the impact of the UE numbers, as explained. It is worth noting that in Fig. 9, A-TCCN outperforms DNN in both Case I and Case II, unlike the situation in Fig. 8. Taken $R = 150$ Mbps as an example, A-TCCN achieves a throughput 23% higher than DNN in Case I and 46% higher in Case II. Along with the results exhibited in Fig. 8, it can be found that A-TCCN can provide a higher throughput than DNN in most cases, especially when the network scale, the UE number, and the data rate requirement are large. When $R$ is low, A-TCCN and DNN achieve almost the same throughput, since both approaches can readily meet a low data rate requirement. In some cases with a small network scale and a small UE number (such as Case I in Fig. 8), the throughput performance of A-TCCN is overtaken by DNN. The reason is two-fold: i) A-TCCN has a smaller size of output than DNN, providing a higher accuracy of AP selection than DNN when $N_a$ and $N_u$ are large; and ii) A-TCCN has a more sophisticated structure than DNN, compromising the accuracy of AP selection when $N_a$ and $N_u$ are small.

C. Computational Complexity

1) Big-O Complexity Analysis: An analysis of the Big-O complexity for A-TCCN along with the baseline methods is summarised in Table IV. SSS chooses the AP with the highest SNR for each UE, and the complexity of finding the largest number is $O(N_a)$. Thus, the Big-O complexity of SSS is $O(N_a N_u)$. As for the iterative methods, the Big-O complexity is $O(N_a N_u I_GT)$, where $I$ denotes the number of iterations. Note that $I_GT$ (i.e., the value of $I$ in GT) is usually smaller than $I_{DO}$ (i.e., the value of $I$ in DO), due to the effectiveness of the mutation strategy in GT [8]. The Big-O complexity of FL is $O(N_a^2 K_1 K_2)$ [10], where $K_1$ is the number of FL inputs and $K_2$ denotes the number of fuzzy rules.

Regarding the learning-aided methods, the computational complexity is analysed for three individual parts: sample collection, training, and implementation. The computational complexity for sample collection is linear with the total number of samples. For a specific case of UE number, the number of samples is the product of the batch number and batch size, i.e., $N \times T$, leading to a Big-O complexity of $O(NT)$. However, it is worth noting that DNN needs $NT$ samples for each case of the UE number, resulting in $MNT$ samples in total. Unlike DNN, A-TCCN can use a mixture training with $M$ cases of the UE number, where $M < M$. As mentioned, in this work $M = 50$ and $M = 10$, allowing A-TCCN to employ just one fifth of the total samples that are needed by DNN.

The complexity of training depends on the number of epochs, which is denoted by $N_{epoch}$. In each epoch, the computational complexity of back propagation is determined by: i) the number of samples, and ii) the number of weights, which is proportional to the input and output sizes [29]. Therefore, the complexity of training the DNN model for $N_u$ UEs can be estimated by $O(N_{epoch} NT N_a N_u)$. As for A-TCCN, the training process is dominated by the condition neural network, of which the input size is $(N_a + 1)M$. As a result, the complexity of training A-TCCN can be estimated by $O(N_{epoch} MNT N_u M)$. It is worth noting that the values of $N_{epoch}$ could be different in DNN and A-TCCN. Once the learning models are trained, the AP selection decision is made through a single-time forward propagation. This process is referred to as implementation. Its complexity in DNN can be approximated by $O(N_a N_u K_m + K_o)$, where $O(K_m)$ stands for the complexity of additions and multiplications in the FC layers, while $O(K_o)$ indicates the complexity of other operations including the BN and activation functions. As for A-TCCN, its Big-O complexity of implementation is similar to DNN, except that $N_u$ needs to be replaced by $M$.

2) Runtime Evaluation: The runtime of A-TCCN is compared against the baseline methods in Table V. Here the runtime of sample collection is for one batch in a specific case of the UE number. Collecting all the samples that are required for training A-TCCN costs about 7 hours in Case I and 18 hours in Case II, respectively. As for DNN, it takes 5 times the hours to acquire a complete dataset, as in this paper it is set that $M/M = 5$. In the training process, DNN costs a much lower runtime per epoch than A-TCCN, as a single DNN model uses only one tenth of the sample dataset of A-TCCN. However, DNN needs a relatively large number of epochs (which is 50 on average) to achieve convergence, while A-TCCN requires 10 epochs only. It takes about 108 and
the SSS method has the lowest runtime within a few µs, but it is still significant even when values of $N$. DO costs the longest runtime, which is over 100ms for larger runtime of the other baseline methods. As shown in Table V, the learning-aided methods is compared with the algorithm operations. For this reason, the implementation runtime of preprocessed, their runtime does not affect the real-time number.

304 seconds in Case I and 460 seconds in Case II in total to training process of A-TCNN in [10]. Particularly, its runtime remains almost the same when $N_a$ as indicated by $O(N_a N_u K_m)$ in its Big-O complexity. Meanwhile, the runtime of A-TCNN increases very slightly with $N_u$ increases. Particularly, its runtime remains almost the same when $N_a$ is above 30. This is because the condition neural network in A-TCNN has a fixed input size, regardless of $N_u$. Further, different network scales (i.e., $N_u$) impose a feeble impact on the runtime of A-TCNN. The reason is that $O(K_m)$ plays a dominant role in the implementation process, weakening the impact of $O(N_a M K_m) due to the change of $N_a$. In summary, A-TCNN and DNN offer a similar amount of runtime, which is substantially lower than the other baseline methods. Compared with FL and GT, A-TCNN can reduce the runtime by up to 687 and 2088 times, respectively.

186 seconds to complete the training process of A-TCNN in Case I and Case II, respectively. Regarding DNN, it costs about 304 seconds in Case I and 460 seconds in Case II in total to train the DNN models for all the selective cases of the UE number.

Since the stages of sample collection and training are preprocessed, their runtime does not affect the real-time operations. For this reason, the implementation runtime of the learning-aided methods is compared with the algorithm runtime of the other baseline methods. As shown in Table V, the SSS method has the lowest runtime within a few µs. DO costs the longest runtime, which is over 100ms for larger values of $N_a$ and $N_u$. GT and FL require a shorter runtime than DO, but it is still significant even when $N_a$ and $N_u$ are small. In Case I, for instance, the runtime is between 2ms and 47ms for GT, and it is between 3ms and 9ms for FL. This level of runtime cannot well meet the low latency requirements in 6G. In contrast, the learning-aided methods cost an ultra-low runtime around 100µs. In addition, it is found that the runtime of DNN exhibits a linear increase with $N_u$, as indicated by $O(N_a N_u K_m)$ in its Big-O complexity. Meanwhile, the runtime of A-TCNN increases very slightly as $N_u$ increases. Particularly, its runtime remains almost the same when $N_a$ is above 30. This is because the condition neural network in A-TCNN has a fixed input size, regardless of $N_u$. Further, different network scales (i.e., $N_u$) impose a feeble impact on the runtime of A-TCNN. The reason is that $O(K_m)$ plays a dominant role in the implementation process, weakening the impact of $O(N_a M K_m)$ due to the change of $N_a$. In summary, A-TCNN and DNN offer a similar amount of runtime, which is substantially lower than the other baseline methods. Compared with FL and GT, A-TCNN can reduce the runtime by up to 687 and 2088 times, respectively.

### TABLE V

| Method                  | UE number | 10  | 20  | 30  | 40  | 50  |
|-------------------------|-----------|-----|-----|-----|-----|-----|
| Collection (per batch)  | DNN       | 8.20| 17.00| 27.00| 37.90| 49.60|
| (Case I)                | A-TCNN    |     |     |     |     |     |
| Collection (per batch)  | DNN       | 24.20| 48.90| 75.10| 103.00| 136.00|
| (Case II)               | A-TCNN    |     |     |     |     |     |
| Training (per epoch)    | DNN       | 471 | 537 | 585 | 667 | 745 |
| (Case I)                | A-TCNN    |     |     |     |     |     |
| Training (per epoch)    | DNN       | 647 | 784 | 941 | 1060| 1270|
| (Case II)               | A-TCNN    | 18.60|     |     |     |     |

| Implementation          | SSS       | 0.0013| 0.0022| 0.0033| 0.0043| 0.0050|
| (Case I)                | GT        | 2.17  | 6.27  | 16.1  | 31.7  | 46.7  |
|                         | DO        | 2.56  | 10.4  | 24.6  | 45.5  | 68.2  |
|                         | FL        | 2.66  | 4.08  | 6.04  | 7.47  | 9.24  |
|                         | DNN       | 0.0651| 0.0856| 0.0998| 0.1052| 0.1224|
|                         | A-TCNN    | 0.0819| 0.0872| 0.0962| 0.0989| 0.0992|

| Implementation          | SSS       | 0.0015| 0.0024| 0.0035| 0.0045| 0.0050|
| (Case II)               | GT        | 4.90  | 18.8  | 44.7  | 94.3  | 138   |
|                         | DO        | 7.41  | 28.9  | 77.4  | 166   | 257   |
|                         | FL        | 3.09  | 4.48  | 6.10  | 7.69  | 10.3  |
|                         | DNN       | 0.0875| 0.1071| 0.1294| 0.1377| 0.1573|
|                         | A-TCNN    | 0.0867| 0.1010| 0.1213| 0.1221| 0.1232|

VI. CONCLUSION

In this paper, a novel learning-aided LB method named A-TCNN was proposed for HLWNets, providing a unique property of handling a varying number of UEs without the need for retraining. Unlike the conventional LB methods determining the AP selection for all UEs together, A-TCNN makes the decision for a single target UE upon the condition of other UEs. This new DNN structure, in conjunction with the adaptive mechanism, enables A-TCNN to handle various UE numbers using a single model with the same coefficients. This makes A-TCNN more practical in comparison with the existing learning-aided LB methods, which require either a dedicated model for each UE number or a general model but with a retraining process. Ablation studies were carried out to validate the effectiveness of the three key components in A-TCNN, including the target-condition neural network, adaptive mechanism, and mixture training. Simulations were conducted to evaluate the network performance of A-TCNN against several baseline methods. Results show that A-TCNN can achieve a higher throughput than DNN in most cases, with an improvement of up to 45%. In addition, A-TCNN offers an ultra-low runtime of around 100µs, reducing computational complexity by 2 and 3 orders of magnitude against FL and GT, respectively. This signifies that A-TCNN has the potential to meet the ultra-low latency requirement of future wireless systems. Further work will investigate A-TCNN in a mobile environment, with the impact of LiFi channel blockage taken into account.

REFERENCES

[1] Cisco Annual Internet Report (2018–2023) White Paper, Cisco, San Jose, CA, USA, 2020, pp. 1–35.
[2] H. Haas, L. Yin, Y. Wang, and C. Chen, “What is LiFi?” J. Lightw. Technol., vol. 34, no. 6, pp. 1533–1544, Mar. 17, 2016.
[3] F. Hu et al., “Si-substrate LEDs with multiple superlattice interlayers for beyond 24 Gbps visible light communication,” Photon. Res., vol. 9, no. 8, pp. 1581–1591, 2021.
[4] X. Wu, “Performance analysis of visible light communications with channel blockage caused by human bodies,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2023, pp. 1–5.
[5] X. Wu, M. D. Soltani, L. Zhou, M. Safari, and H. Haas, “Hybrid LiFi and WiFi networks: A survey,” IEEE Commun. Surveys Tuts., vol. 23, no. 2, pp. 1398–1420, 2nd Quart., 2021.
[6] D. A. Basnayaka and H. Haas, “Hybrid RF and VLC systems: Improving user data rate performance of VLC systems,” in Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring), May 2015, pp. 1–5.
[7] X. Li, R. Zhang, and L. Hanzo, “Cooperative load balancing in hybrid visible light communications and WiFi,” IEEE Trans. Commun., vol. 63, no. 4, pp. 1319–1329, Apr. 2015.
[8] Y. Wang, X. Wu, and H. Haas, “Load balancing game with shadowing effect for indoor hybrid LiFi/RF networks,” IEEE Trans. Wireless Commun., vol. 16, no. 4, pp. 2366–2378, Apr. 2017.
[9] S. Aboagye, T. M. N. Ngatched, O. A. Dobre, and A. Ibrahim, “Joint access point assignment and power allocation in multi-tier hybrid RF/VLC HetNets,” IEEE Trans. Wireless Commun., vol. 20, no. 10, pp. 6329–6342, Oct. 2021.
[10] X. Wu, M. Safari, and H. Haas, “Access point selection for hybrid Li-Fi and Wi-Fi networks,” IEEE Trans. Veh. Technol., vol. 65, no. 12, pp. 5375–5385, Dec. 2017.
[11] H. Ji and X. Wu, “A novel method of combining decision making and optimization for LiFi resource allocation,” in Proc. IEEE Globecom Workshops (GC Wkshps), Dec. 2022, pp. 1616–1621.
[13] A. M. Alenezi and K. A. Hamdi, “Reinforcement learning approach for hybrid WiFi-VLC networks,” in Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring), May 2020, pp. 1–5.

[14] R. Ahmad, M. D. Soltani, M. Safari, and A. Srivastava, “Reinforcement learning-based near-optimal load balancing for heterogeneous LiFi WiFi network,” IEEE Syst. J., vol. 16, no. 2, pp. 3084–3095, Jun. 2022.

[15] X. You et al., “Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts,” Sci. China Inf. Sci., vol. 64, no. 1, pp. 1–74, Nov. 2020.

[16] E. Gures, I. Shayea, M. Ergen, M. H. Azmi, and A. A. El-Salah, “Machine learning-based load balancing algorithms in future heterogeneous networks: A survey,” IEEE Access, vol. 10, pp. 37689–37717, 2022.

[17] X. Wu and D. C. O’Brien, “Parallel transmission LiFi,” IEEE Trans. Wireless Commun., vol. 19, no. 10, pp. 6268–6276, Oct. 2020.

[18] Y. Qi, M. Hunakumbe, M. Nekovee, J. Loreca, and V. Sgarladi, “Quantifying data rate and bandwidth requirements for immersive 5G experience,” in Proc. IEEE Int. Conf. Commun. Workshops (ICC), May 2016, pp. 455–461.

[19] J. M. Kahn and J. R. Barry, “Wireless infrared communications,” Proc. IEEE, vol. 85, no. 2, pp. 265–298, Feb. 1997.

[20] J.-H. Wang, Q.-S. Hu, J. Wang, M. Chen, and J.-Y. Wang, “Tight bounds on channel capacity for dimmable visible light communications,” J. Lightw. Technol., vol. 31, no. 23, pp. 3771–3779, Dec. 17, 2013.

[21] 5G ToB Service Experience Standard White Paper, Huawei, Shenzhen, China, Jan. 2021, pp. 1–140. [Online]. Available: https://carrier.huawei.com/-/media/cnbvy2/download/products/services/5g-b2b-service-experience-standard-white-paper-en1.pdf

[22] P. Belotti, C. Kirches, S. Leyffer, J. Luedtke, and A. Mahajan, “Mixed-integer nonlinear optimization,” Acta Numer., vol. 22, pp. 1–131, May 2013.

[23] D. A. Medler and M. R. Dawson, “Using redundancy to improve the performance of artificial neural networks,” in Proc. Biennial Conf. Can. Soc. Comput. Stud. Intell. Mississauga, ON, Canada: Canadian Information Processing Society, 1994, pp. 131–138.

[24] P. Kohli and C. G. C. Teyssey, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 446–454.

[25] G. Montavon, G. Orr, and K.-R. Müller, Neural Networks: Tricks of the Trade, vol. 7000. Cham, Switzerland: Springer, 2012.

[26] Z. Zhang and M. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” in Proc. Adv. Neural Inf. Process. Syst., vol. 31, 2018, pp. 1–11.

[27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[28] H. Ji, Q. Wang, X. Wu, S. J. Redmond, and I. Tavakkolnia. (Jul. 2023). A research intern position with the Smart Network Technology Lab at Huawei Technologies. He has been collaborating with the University of New South Wales, Sydney, Australia, where he held an Australian Research Council Future Fellowship, prior to his return to UC Berkeley in 2019. He is currently pursuing the Ph.D. degree with University College Dublin under the guidance of Prof. Stephen J. Redmond. His Ph.D. research primarily focuses on applying machine learning techniques to physical control tasks.

[29] Stephen J. Redmond (Senior Member, IEEE) received the bachelor’s and Ph.D. degrees in electronic engineering from University College Dublin (UCD) in 2002 and 2006, respectively. He is currently an Associate Professor in biomedical engineering with the School of Electrical and Electronic Engineering, UCD. His Ph.D. studies was in the areas of biological signal processing, focusing on pattern classification, in particular the application of machine learning techniques to physiological signals. He has extensive experience in monitoring human movement using wearable sensors, predictive modelling of health events, and robust measurement of physiological signals in the home. He spent ten years with The University of New South Wales, Sydney, Australia, where he held an Australian Research Council Future Fellowship, prior to his return to UCD in 2019. He returned to Ireland to take up a prestigious Science Foundation Ireland President of Ireland Future Research Leaders&’39 and Award (2019–2023) to develop next-generation tactile sensors for robotic and prosthetic gripping that can feel slipperiness. He has a strong track record in the exploitation and commercialization of research. He has worked in an advisory/consultancy capacity for several biomedical engineering companies, worked as the Data and Insights Director of Irish medtech start-up, FIRE1, developing a technology to manage heart failure, and co-founded Australian tactile sensor company, Contactile.

[30] Qiang Wang (Student Member, IEEE) received the Ph.D. degree from The University of Edinburgh, U.K., in 2015. He is currently an Assistant Professor with University College Dublin (UCD), Ireland. Prior to joining UCD, he was a Research Associate with the University of Oxford. His main research interests are in the areas of 6G mobile communications, optical wireless communications (OWC), and the Internet of Things. A particular focus is on developing hybrid wireless networks that integrate OWC and radio frequency, empowered by software-defined networking and artificial intelligence. He has authored or coauthored over 50 publications in these areas. He was a recipient of the Best Paper Award at IEEE ICC in 2021 and a recipient of the Royal Irish Academy (RIA) Charlemont Grant Award in 2022.