

RESEARCH ARTICLE

An Optimized Multi-Task Learning Model for Disaster Classification and Victim Detection in Federated Learning Environments

YI JIE WONG¹, MAU-LUEN THAM¹, BAN-HOE KWAN²,
EZRA MORRIS ABRAHAM GNANAMUTHU¹,
AND YASUNORI OWADA³, (Member, IEEE)

¹Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Selangor 43000, Malaysia

²Department of Mechatronics and Biomedical Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Selangor 43000, Malaysia

³Resilient ICT Research Center, Network Research Institute, National Institute of Information and Communications Technology (NICT), Tokyo 184-8795, Japan

Corresponding author: Mau-Luen Tham (thamml@utar.edu.my)

This work is the output of the ASEAN IVO (http://www.nict.go.jp/en/asean_ivo/index.html) project, Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network, and financially supported by NICT (<http://www.nict.go.jp/en/index.html>).

ABSTRACT Disaster classification and victim detection are two important tasks in enabling efficient rescue operations. In this paper, we propose a multi-task learning (MTL) model which accomplishes these two tasks simultaneously. The idea is to attach one pruned head model to another backbone network. We mathematically pinpoint the optimal branching location and the depth of the pruned head model. Apart from the decoupled task training capability, the MTL model offers lesser memory requirements (12.8 MB saving) and better disaster classification accuracy (1-2% gain), while preserving the same detection performance (0.694 of average precision (AP)), as compared to the traditional method. Such advantages of flexibility, speed and accuracy facilitate the large-scale deployment of Internet of Things (IoT) applications, where we explore the potential of federated learning (FL) and active learning (AL). Given the high ambiguity within disaster images, a modified version of AL-based technique is introduced. For realistic implementation, production-ready OpenFL and OpenVINO tools are adopted to update the global FL model and to optimize the trained model, respectively. Experiment results are promising: the FL-based techniques are comparable to or better than their centralized learning (CL) counterparts. Also, our application portability is demonstrated via different hardware such as CPU and Raspberry Pi.

INDEX TERMS Disaster classification, victim detection, convolution neural network (CNN), hard parameter sharing, representation similarity analysis, multi-task learning, federated learning, uncertainty sampling, optimal branching, OpenVINO, OpenFL.

I. INTRODUCTION

Annually, natural disasters inflict damages, monetary costs, injuries, and deaths. For instance, the 2021 Fukushima earthquake inflicted 187 casualties, while causing significant damage across Japan [1]. Given that the first 72 hours after a disaster are critical for rescuing survivors [2], disaster detection plays a vital role in facilitating search and rescue efforts.

The associate editor coordinating the review of this manuscript and approving it for publication was Turgay Celik¹.

The successfulness of these operations heavily relies on the reported activity of disasters and number of victims.

Deep learning (DL) can extract the aforementioned features through a convolutional neural network (CNN). Disaster classification task can be readily trained by utilizing CNN architectures such as VGG16 [3] and MobileNet [4]. Whereas for victim counting, it falls into the class of object detection task, which can be addressed by the popular CNN models such as You Only Look Once (YOLO) [5] and Single-Shot Detector (SSD) [6]. In the literature on disaster detection, these two tasks are generally studied in isolation. How to

design a joint disaster classification and victim detection CNN model is a topic worthy of investigation.

Training a disaster detection model in practice presents another technical hurdle. Existing works commonly assume that the abundant labelled dataset is available at a centralized server with high-performance graphical processing units (GPUs) [7]. These assumptions do not hold in a large-scale disaster monitoring environment, especially with a massive deployment of relatively low powered Internet of Things (IoT) devices. Within an IoT, all connected devices are able to collect and exchange data. However, such flexibility is accompanied with several challenges such as the scarcity of labelled dataset, data privacy concerns and prohibitive cost of transmitting data as training samples. Federated learning (FL) is an emerging paradigm that can help to build an accurate global CNN model via a collaborative training among edge IoT devices, without sharing the confidential and bandwidth-hungry data.

A few recent works such as [8] and [9] have demonstrated the promising performance of disaster classification via FL. However, training-level evaluation results do not necessarily translate into good inference performance. For actual model deployment in production environment, the legitimate judges of CNN model quality are IoT local devices, serving as monitoring nodes. Given the heterogeneity of IoT system, the portability and acceleration of inference process are crucial towards scalable disaster monitoring frameworks.

In this paper, we optimize the CNN performance at both training and inference stages. The starting point is the design of an efficient multi-task learning (MTL) model that simultaneously performs disaster classification and victim detection. The training burden is relieved by active learning (AL), which allows the training algorithm to interactively query and label informative data from the pool of unlabelled dataset in each local IoT device.

Once the model is trained, we aim to minimize the processing time while maximizing classification and detection performance at the inference phase. Indeed, this stage must be designed and analyzed correctly in order to achieve a robust model working in production environment. To this end, we first accelerate the inference process and port the optimized model on different Intel platforms via the Intel OpenVINO toolkit [10]. It is comprehensive toolkit which fine-tunes and optimizes DL inference performance on target low-powered devices. Note that the optimized model facilitates edge computing, which is one of goals of the ASEAN IVO project titled “Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network”.

Experiment results are encouraging: the FL-based disaster detection techniques are comparable to or better than their centralized learning (CL) counterparts. Our application portability is demonstrated via different hardware such as CPU and Raspberry Pi. Under the same hardware, the optimized model achieves 151% of frames per second (FPS) gain over the original MTL model, while having higher accuracy and slightly lower average precision (AP).

A preliminary version of this article appeared at the IEEE UEMCON 2021 [12]. While sharing the same basic solution concept, this version includes a substantial amount of new material, including a discussion on how optimal branching can be determined by quantitative analysis instead of empirical approach, an extended framework with the aid of AL and FL, and new results for deployment in production environment. The main contributions of this work are summarized as follows:

1. Existing studies focus on solving single-task issue of disaster classification [13], [16], [27], [28], [29] and victim detection [18], [19], [20], [21], [31], [32], [33] separately. In contrast, we introduce a MTL model by attaching a disaster classification head model to the backbone of a victim detection model. Different from existing MTL works [34], [35], [36], [37], [38], we employ an efficient mathematical analysis to pinpoint the optimal branching location and to prune the head model.
2. The framework design decouples training of two tasks. Solutions can be found in a per-task fashion before merging them into one unified model, which has smaller size than a combination of two separate single-task models. Such lightweight network architecture facilitates both bandwidth-sensitive FL training and cost-limited inference. On top of being lightweight, the proposed model can even produce better classification-related accuracy while preserving the same detection-related AP.
3. Most AL methods advocate uncertainty sampling, which selects the most uncertain samples from the unlabeled data pool to label [22]. Such strategy is ill-suited for disaster dataset, where samples from different classes exhibit high similarity. To enable efficient AL-based FL, we introduce a simple heuristic by combining both uncertainty and diversity samplings.
4. The correctness of the post-training optimization results, especially for model accuracy, is very crucial for actual deployment. The majority of the research in [23], [24], [25], and [26] tries to accelerate the inference process without detailing the degree of accuracy loss. In contrast, our measurement outputs are based on open-source and production-ready frameworks to ensure reusability, interoperability, and scalability.

The rest of the paper is organized as follows. Section II describes the related work. Section III presents the proposed solution. Section IV discusses the experimental setup, followed by results and discussions. Section VI concludes the paper and outlines future research directions.

II. RELATED WORK

To give the readers a big picture of the works in this broad area, this section reviews related works on disaster classification and victim detection, MTL, FL and AL, followed by inference optimization.

A. DISASTER CLASSIFICATION AND VICTIM DETECTION

The performance of disaster monitoring is tightly connected with the quality and quantity of dataset. The authors in [15] collected and filtered tweet messages that people post during disasters into one dataset, known as Artificial Intelligence for Disaster Response (AIDR). Similar work can be found in [14], where a large multimodal dataset collected from Twitter during different natural disasters, known as Crisis-MMD was released. To facilitate benchmarking purpose, the authors in [16] consolidated the aforementioned datasets into a dataset called Crisis Image Benchmarks Dataset (Crisis-IBD), which will be served as input dataset in this paper.

Inspired by the richness of dataset information, various disaster classification methods have been devised. The work in [28] analyzed the aerial images for flood magnitude assessment. However, the assessment is limited to only single disaster type. By focusing on four natural disasters, the authors in [27] proposed a damage assessment method which outperforms traditional machine learning approach. The work in [16] also investigated multi-disaster classifications by harnessing the power of several existing CNN models such as VGG16 and MobileNet. However, these CNNs are directly used without any modification for further improvement. Differently, we prune the MobileNetv2 network in such a way that it can be attached to another CNN backbone network and yet performs better than the original version. Another CNN framework was adopted in [13], where multiple pre-trained unimodal CNNs that extract textual and visual features independently are combined and fed into a final classifier for disaster damage identification. The results in [13], [16], [27], and [28] however, did not discuss the inference speed aspect, which is critical for real-time disaster response. Besides that, the aforementioned works focus on single-task domain.

In [29], the authors presented a cross-domain dataset, called FloodNet, which incorporates tasks of image classification, semantic segmentation and visual question answering. These tasks are accomplished by executing three separate models. Such approach (hereafter referred to as conventional approach), however, requires high memory footprint and computational resources.

Unlike the previous works [13], [15], [27], [29] which focus on single-task classification, the same authors in [16] extended their work to a multi-task classification model [17], which targets on (i) disaster types, (ii) informativeness, (iii) humanitarian, and (iv) damage severity assessment. However, the solution is limited to the image-classification domain, without considering the victim detection.

Another pool of literature is exploring the potential of IoT technologies in detecting victims. Unmanned aerial vehicle (UAV) has emerged as one of the effective IoT solutions for dealing with a broad affected area [30]. In [18], the authors leveraged a MobileNet-SSD model to detect victims of natural disaster through Raspberry Pi camera installed on a drone. The work in [19] investigated similar problem by considering a thermal camera. Results show that their victim detection from aerial thermal view can achieve up to AP of 82.49 %.

The studies in [20] and [21] shifted their focus from aerial view to burning building and flood scenes, respectively. Apart from the aforementioned image-based victim detection, the authors in [31] proposed an integrated audio-visual human search system, in order to boost the system performance. The works in [32] and [33] took another divergent approach by locating mobile terminals based on radio frequency (RF) signal. However, this method is effective only when user equipment and victims are in the proximity of each other. Furthermore, none of the above works [18], [19], [20], [21], [31], [32], [33] consider a multitask system that concurrently strives for two coupled goals.

From the literature survey, it is observed that disaster classification and victim detection are generally studied in isolation. In contrast, our work aims to develop a MTL model which executes these two tasks simultaneously.

B. MULTI-TASK LEARNING (MTL)

MTL is to perform more tasks using one model, without the need of using a separate model for each task. In the context of object detection, MTL can be categorized into three types. In the first category, the number of head models represents the total tasks needed to perform. If the head models share a backbone, it is called hard parameter sharing. Whereas for soft parameter sharing, each task has its own backbone. Examples of using hard parameter sharing can be found in [34] and [35]. In self-driving car application, the work in [34] added another head model for lane lines detection to the joint segmentation and detection model. The scheme in [35] adopted four head models for (i) citrus detection and (ii) segmentation, as well as (iii) maturity and (iv) quality classification on the citrus detection. On the other hand, the authors in [36] resorted to the soft parameter sharing approach, for achieving joint detection and segmentation.

Secondly, multi-tasking is made possible with minimal modifications on the original detector model. It was demonstrated in [37] for the application of joint vehicle classification and distance estimation. The idea is to make the distance prediction a classification task and subsequently merge it with the task of vehicle classification in order to form a unified task. Thirdly, some models improve their main tasks based on several auxiliary tasks. For example, [38] defined three auxiliary tasks, namely (i) closeness labelling, (ii) multi-object labelling and (iii) foreground labelling, in order to refine the learning process of the object detection model.

The successes of the aforementioned MTL solutions are proven via a centralized data availability. Such assumption does not hold in a large-scale disaster monitoring scenario. How effectively MTL can be trained from distributed datasets at local devices is still largely missing. Also, majority of these works adopt empirical approach to determine the best branching settings by performing transfer learning on different combinations and subsequently selecting the optimal one. Such approach requires intensive computation due to the additional training on each combination to evaluate the transfer learning performance.

This paper aims to cast some light on these aspects by utilizing FL and smarter branching selection strategy.

C. FEDERATED LEARNING (FL) AND ACTIVE LEARNING (AL)

In FL, only the model weights have to be transferred across the network for aggregation, which is more efficient as compared to sharing the entire dataset. Such FL benefits are exploited in a wide variety of applications ranging from healthcare [39], wireless communications [40], through vehicular edge computing [41], to manufacturing [42]. In the context of disaster detection, the work in [9] proposed a FL and autonomous UAVs for hazardous zone detection. The CNN-LSTM model weights trained within each UAV are transmitted to a central server for global model aggregation. Despite promising results, the FL usage has been limited by single-task models adopted in these previous works.

The scheme in [8] also considered FL based single-task disaster classification, with additional concern regarding the annotation burden for each local training. Armed with AL, the authors reported that the proposed AL-based FL framework performs equally well under two strategies namely uncertainty sampling and query by committee. Our work distinguishes itself by offering more insights into the properties of disaster dataset. For dataset samples that are close to classification boundary, uncertainty sampling may always choose similar samples without diversity [43]. Furthermore, most of the aforementioned works such as [8] and [9] do not use production-ready tools for FL implementation.

D. INFERENCE OPTIMIZATION

Efficient execution of a CNN model is undoubtedly another important criterion for implementing production-ready DL solutions. This is especially true for deploying heterogeneous IoT devices of different hardware constraints. How to enable fast inference on low-powered embedded platforms remains an open research question. Intel OpenVINO toolkit emerges as an extremely useful tool of choice since it optimizes DL models across Intel hardware while minimizing the inference time [11]. A large portion of the studies discussed above quite commonly neglect this design aspect and demonstrates their DL solutions based on expensive GPU resources.

By recognizing the importance of inference optimization, a plethora of works utilized OpenVINO on various use cases such as license plate detection [23], person re-identification system [24] and face recognition [25]. Work that explicitly optimizes OpenVINO model for disaster scenario was found in [26]. However, all these research tries to accelerate the inference process without detailing the degree of accuracy loss. An allied question is: How much accuracy and AP we need to sacrifice while pursuing faster inference? In contrast, our measurement outputs are based on OpenVINO DL Workbench [11], which is an open-source and production-ready framework to ensure reusability, interoperability, and scalability.

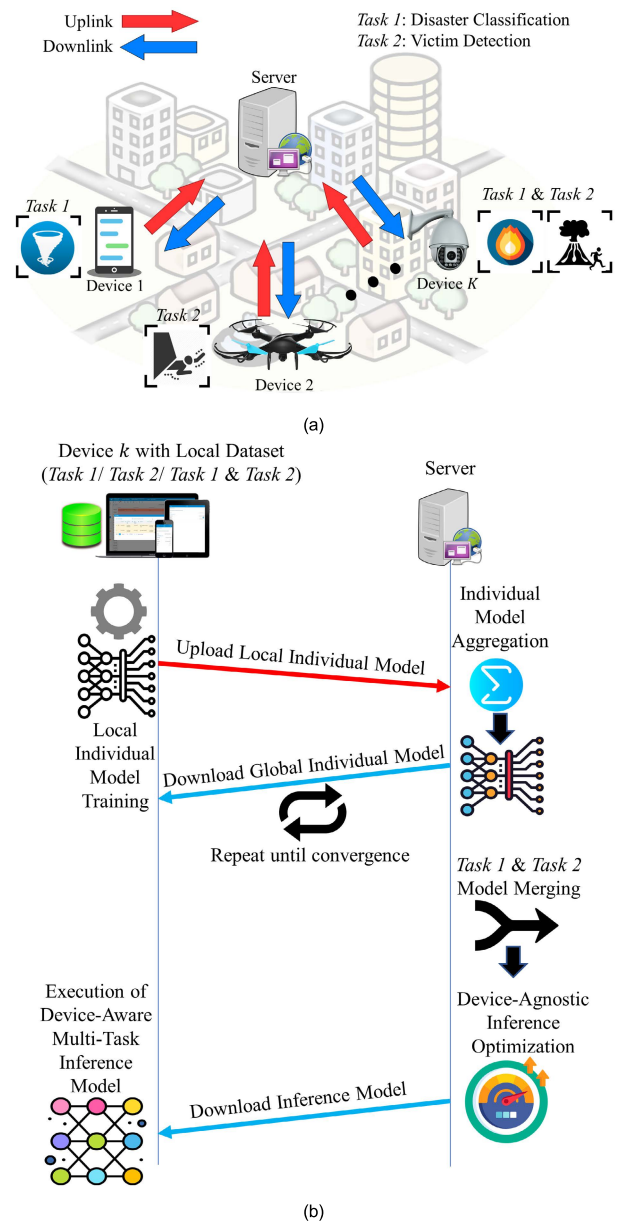


FIGURE 1. Overview of the proposed disaster detection framework. (a) Communications between server and devices. (b) Interaction from training to inference.

III. PROPOSED APPROACH

Fig. 1 displays the overview of the proposed disaster detection framework. In the federated network, there are K devices communicating with a server. Each IoT device can locally train its model for *Task 1*: Disaster Classification or *Task 2*: Victim Detection, or in combination of both. Note that even within the same device, both tasks are trained individually for the following rationales. Firstly, it allows fine-tuning of specific tasks, depending on target performance requirements. Secondly, not all clients have gathered both task information. Thirdly, some devices are not powerful enough to train both tasks.

The overall procedures are illustrated in Fig. 1(b). Firstly, the local training for *Task 1* or/and *Task 2* are executed.

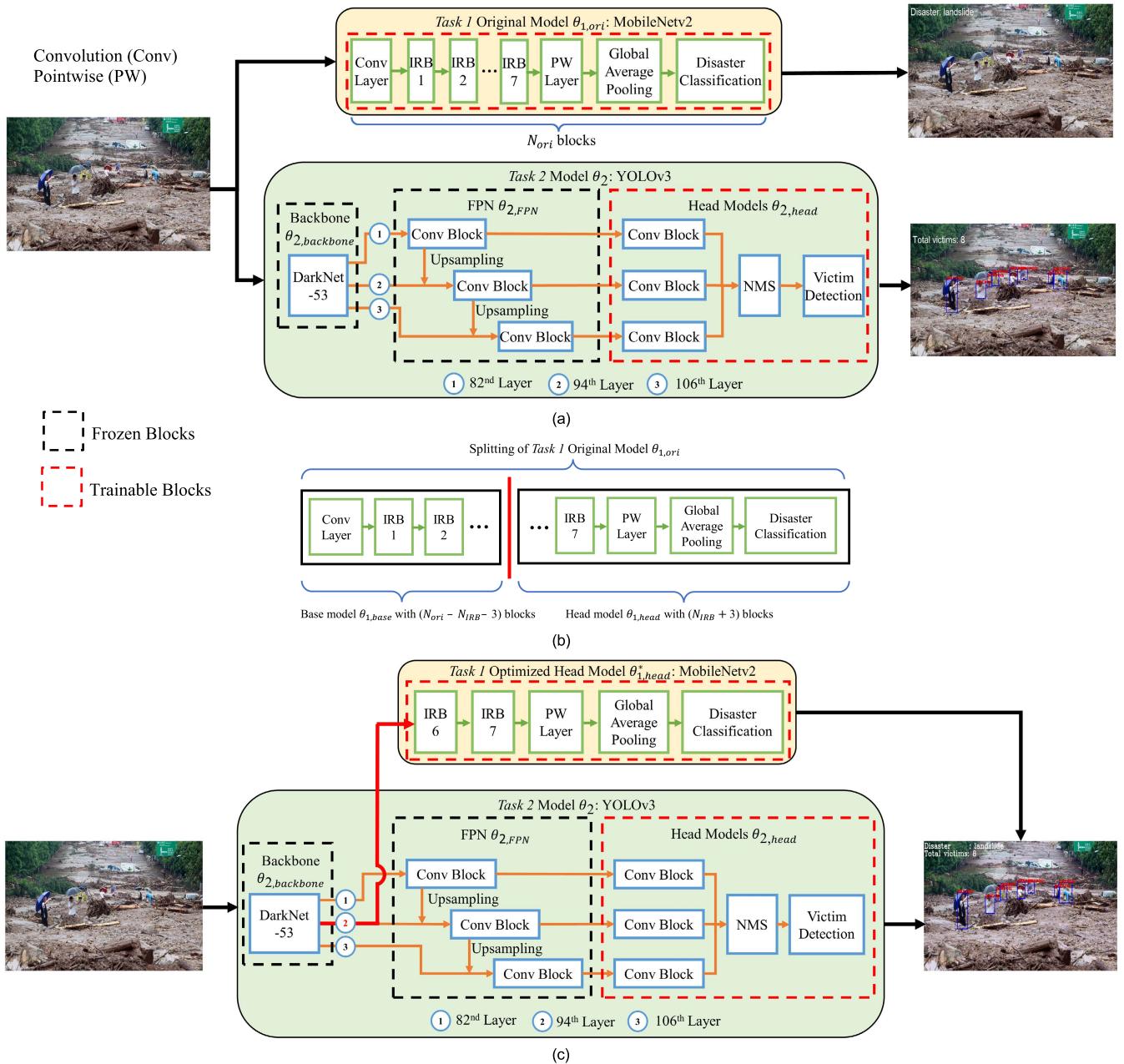


FIGURE 2. Network architecture of Task 1 and Task 2. (a) Conventional model. (b) Task 1 splitting. (c) Proposed model.

Secondly, the local model weights are transmitted to the server for model aggregation. Thirdly, the global fine-tuned model is sent back to each device for another round of training. This process repeats until convergence. Fourthly, both individual trained models are merged into a single unified model. Fifthly, the multi-task model is optimized in a device agnostic manner. Lastly, the optimized model is executed on device k by setting the inference engine mode compatible with their own hardware.

A. MTL MODEL

Since hard parameter sharing is the most frequently used approach in MTL [45], our design follows this setting by

branching a disaster classification head model from the backbone of a selected object detection model.

In this work, we select MobileNetV2 [46] and YOLOv3 as the model for Task 1 and Task 2, respectively. MobileNetV2 is one of the lightweight network architectures, which is suitable for real-time disaster classification. Whereas for YOLOv3, it is one of the most widely used object detector [47], thanks to its superiority in achieving the trade-off between accuracy and speed [48]. Note that our proposed branching strategy is not limited to only these two CNNs and can be expectably applicable to other CNNs such as YOLOv7 [49].

Fig. 2 (a) depicts the “conventional approach” where the same image serves as an input to two separate models,

which undergo transfer learning for accomplishing *Task 1* and *Task 2*, respectively. Specifically, *Task 1* adopts a pre-trained model on ImageNet and finetunes all N_{ori} blocks for disaster classification. This original model is denoted as $\theta_{1,ori}$, which consists of one convolution layer, seven inverted residual blocks (IRBs), one pointwise convolution layer followed by a global average pooling, and one more pointwise convolution layer for the classification.

On the other hand, *Task 2* initially extracts all weights learned from the MS-COCO pre-trained model. Then, the backbone known as DarkNet-53 and Feature Pyramid Network (FPN) are kept frozen. FPN takes three feature maps from the 82nd, 94th and 106th of DarkNet-53 as its inputs. Correspondingly, there are three head models which detect object at different scales. To detect victim, the weights belonging to three head models are fine-tuned. Given that an object detector will likely predict more than one bounding box for the same object, we apply the non-maximum suppression (NMS) technique for removing redundant object. Here, we denote *Task 2* model as θ_2 .

To merge these two CNNs into one unified model, the following questions arise. Questions: How do we split $\theta_{1,ori}$ into one base model $\theta_{1,base}$ and one head model $\theta_{1,head}$, as shown in Fig. 2 (b)? Where do we attach $\theta_{1,head}$ among three different depth of the shared DarkNet-53? Fig. 2 (c) provides the answers, where the optimized head model $\theta_{1,head}^*$ consists of two IRBs, followed by the remaining blocks, and $\theta_{1,head}^*$ is branched from the 94th location. Another question arising is: How do we decide these optimal settings with quantitative analysis? Hence, it is important to investigate the relationship between *Task 1* and *Task 2*.

The study in [50] demonstrates that representation similarity analysis (RSA) can measure the task similarity using the learned representations, without any subsequent training. Their results show that a higher score of task similarity leads to better model selection strategy for transfer learning. Here, we adopt the RSA to decide the optimal branching location. We enumerate the steps to compute the similarity score for a different merging combinations of *Task 1* and *Task 2* in the following paragraph.

Firstly, a subset of images is randomly selected from CrisisIBD as the conditions for dissimilarity computation. We can acquire the representation or feature map of each image at any layers of a CNN by forward passing the image until the target layer. The dissimilarity score of a pair of images can be expressed as $1 - \rho$, where ρ is the Pearson's correlation coefficient of the feature maps of the two images. ρ is formulated as follows:

$$\rho(x, y) = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^N (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^N (y_i - \bar{y})^2}} \quad (1)$$

where N represents the feature map size. Then, a representation dissimilarity matrix (RDM) is populated by the dissimilarity scores for all pair of images in the subset. This process

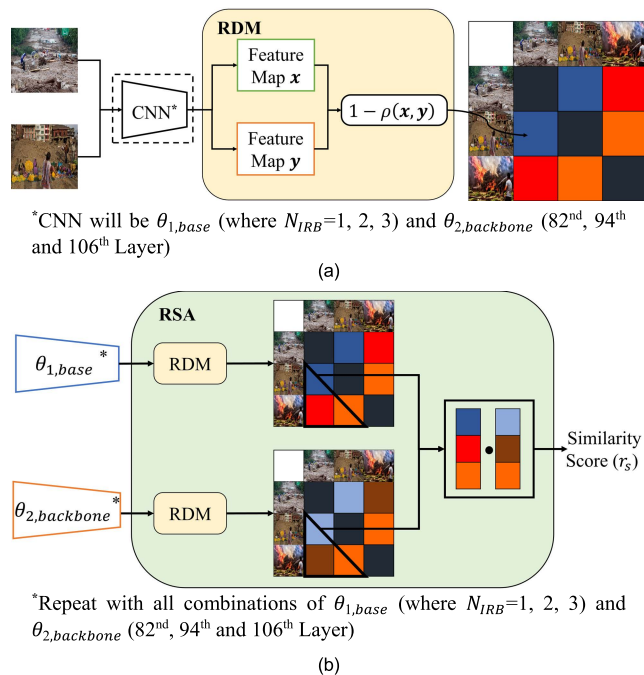


FIGURE 3. RSA approach to quantify the similarity score between *Task 1* and *Task 2*. (a) RDM computation. (b) r_s computation.

is repeated six times for different CNN frameworks, as shown in Fig. 3 (a).

Secondly, the similarity between the RDMs of two CNNs can be computed with the Spearman's correlation (r_s) between the upper or lower triangular part of the RDMs, as shown in (2):

$$r_s = 1 - \frac{6 \sum_{i=1}^M d_i}{M(M^2 - 1)} \quad (2)$$

where d_i denotes the difference between the ranks of i^{th} elements of the lower triangular part of the two RDMs in Fig. 3 (b), and M is the number of elements in the lower triangular part of the RDM. This procedure is repeated nine times for various combinations of two CNNs, as shown in Fig. 3 (b). Intuitively, the combination pair with the highest r_s yield the best multitasking performance, which will be validated in Section IV.

The unified model in Fig. 2 (c) deserves further elaboration. The frozen model weights from $\theta_{2,backbone}$ and $\theta_{2,FPN}$ allows the training process of *Task 1* $\theta_{1,head}^*$ and *Task 2* $\theta_{2,head}$ to be decoupled. This indicates that solutions can be found in a per-task fashion before merging them into one unified model. Such lightweight network architecture facilitates both bandwidth-sensitive FL training and cost-limited inference. On top of being lightweight, the proposed model can even produce better classification-related accuracy while preserving the same detection-related AP. This is accomplished by transferring feature representations from denser network $\theta_{2,backbone}$ to learn *Task 1*.

Overall, the benefits of using the model are flexibility, speed, and accuracy. The training procedures of *Task 1*

Algorithm 1 Training Strategy for *Task 1*

Input: Labeled Dataset, \mathcal{L}
 Number of Epoch, \mathcal{N}_{t1}
 Learning rate, α
Task 1 Head Model, $\theta_{1,head}^*$
 Categorical Cross-Entropy Loss Function, I_{CCE}

Output: Trained *Task 1* Head Model, $\theta_{1,head}^*$

- 01 \mathcal{L} is divided into mini batches of data, l
- 02 Obtain and freeze pre-trained $\theta_{2,backbone}$ (until 94th Layer)
- 03 // Train *Task 1* Head Model, $\theta_{1,head}^*$
- 04 **for** $t = 1 : \mathcal{N}_{t1}$ **do**
- 05 **for** l in \mathcal{L} **do**
- 06 // Use $\theta_{2,backbone}$ to extract l 's feature maps, f
- 07 $f \leftarrow \theta_{2,backbone}(l)$
- 08 // Compute gradients and update model
- 09 $\nabla \leftarrow \frac{\partial}{\partial x} \mathcal{J}_{CCE}(\theta_{1,head}^*, f)$
- 10 $\theta_{1,head}^{*t+1} \leftarrow \theta_{1,head}^{*t} - \alpha \nabla$
- 11 **end for**
- 12 **end for**

and *Task 2* are described in Algorithm 1 and Algorithm 2, respectively.

B. ACTIVE LEARNING (AL)

There exist two pool-based strategies, namely uncertainty sampling and query by committee. We choose the former since it is one of the most popular approaches [22] and consumes lesser computational power [51]. The category of uncertainty sampling can be further divided into three subgroups namely least confidence, entropy sampling, and margin sampling. Here, we focus on only the third technique since these three methods perform equally well in a disaster classification scenario [8].

Fig. 4 visualizes the t-SNE results for the test images from CrisisIBD [16]. From the figure, it is observed that most of the images, regardless of the classes, are clustered near to the centre. These samples, labelled as ‘‘hard’’, would always be prioritized by margin sampling in terms of selection. Table 1 illustrates some sample images from the CrisisIBD [16] with high ambiguity.

A better strategy is to incorporate diversity into the query process [43]. To this end, we design a simple heuristic by combining both uncertainty and diversity samplings. Apart from the hard samples, our modified sampling process as shown in Algorithm 3 considers two additional categories namely ‘‘easy’’ and ‘‘moderately-hard (mod)’’. These samples represent those that are far away from the centre and have clear classification boundaries. Specifically, for each round of query selection in Phase 2, all available unlabelled samples are ranked in terms of uncertainty and sorted in a descending order. Hard, moderately-hard and easy samples are then picked according to the lines 9, 10, 11 of Algorithm 3. These selected samples are removed from the unlabelled

Algorithm 2 Training Strategy for *Task 2*

Input: Labeled Dataset, \mathcal{L}
 Number of Epoch, \mathcal{N}_{t2}
 Mini Batch Gradient Accumulation Round, \mathcal{B}
 Learning Rate, α
Task 2 Head Model, $\theta_{2,head}$
 YOLOv3 Loss Function, \mathcal{J}_{Y3}

Output: Trained *Task 2* Head Model, $\theta_{2,head}$

- 01 \mathcal{L} is divided into mini batches of data, l
- 02 Obtain and freeze both pre-trained $\theta_{2,backbone}$ and $\theta_{2,FPN}$
- 03 // Train *Task 2* Head Model, $\theta_{2,head}$
- 04 **for** $t = 1 : \mathcal{N}_{t2}$ **do**
- 05 Counter: $c \leftarrow 0$
- 06 Accumulated Gradients: $\nabla_{accumulate} \leftarrow 0$
- 07 **for** l in \mathcal{L} **do**
- 08 // Compute & accumulate gradients
- 09 $\nabla \leftarrow \frac{\partial}{\partial x} \mathcal{J}_{Y3}(\theta_{2,head}^t, l)$
- 10 $\nabla_{accumulate} \leftarrow \nabla_{accumulate} + \nabla$
- 11 // Update model
- 12 **if** $c \bmod \mathcal{B} = 0$ **then**
- 13 $\nabla_{accumulate} \leftarrow \nabla_{accumulate} / \mathcal{B}$
- 14 $\theta_{2,head}^{t+1} \leftarrow \theta_{2,head}^t - \alpha \nabla$
- 15 $\nabla_{accumulate} \leftarrow 0$
- 16 **end if**
- 17 // Increase α after 10 epochs of warm up training
- 18 **if** $t \bmod 10 = 0$ **then**
- 19 $\alpha \leftarrow \alpha \times 10$
- 20 **end if**
- 21 **end for**
- 22 **end for**

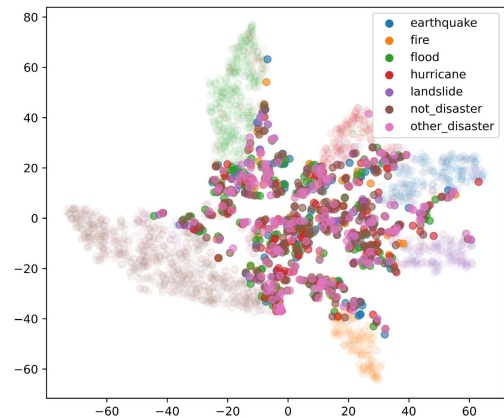




FIGURE 4. t-SNE results for the test dataset. Bold color corresponds to 33% of hard samples.

dataset pool and the process repeats until the communication epoch \mathcal{N}_a is reached.

C. FEDERATED LEARNING (FL)

To minimize the effort of implementation, we choose the simple Federated Averaging (FedAvg) algorithm as in [52], [53], and [54]. FedAvg combines the model parameters collected from each local device via averaging. Algorithm 4 describes the overall process. Firstly, a FL server is initialized with a global model. Secondly, it will share the global copy with

TABLE 1. Examples of high-similarity images in CrisisIBD [16].

Sample Image	Predicted Class	Actual Class
	Flood	Hurricane
	Not Disaster	Hurricane

Algorithm 3 The Proposed Active Learning Process

Input: Initial Model, $\theta_{1,head}^*$
Number of Active Learning Epoch, N_a
Small Labeled Dataset (seed), \mathcal{L}_0
Unlabeled Dataset, \mathcal{U}_t
Uncertainty Function, \mathcal{F}_{unc}
Query Batch Size, $(\mathcal{K}_{easy}, \mathcal{K}_{mod}, \mathcal{K}_{hard})$

Output: Labeled Dataset, \mathcal{L}_t

- 01 **Each Client executes:**
- 02 // Phase 1: Warm Up the Model
- 03 $\theta_{1,head}^{*0} \leftarrow \text{train } \theta_{1,head}^*$ for 5 epochs using \mathcal{L}_0
- 04 **for** $t = 1 : N_a$ **do**
- 05 // Phase 2: Query Selection
- 06 $\mathcal{Q} \leftarrow \mathcal{F}_{unc}(\mathcal{U}_t, \theta_t)$; Rank the uncertainty of each data point in \mathcal{U}_t
- 07 Arrange \mathcal{Q} in descending order based on uncertainty
- 08 $\mathcal{Q}_t^{easy} \leftarrow \{\mathcal{U}_{t,i} | inc_i \in \text{argbtmK}(\mathcal{Q}, \mathcal{K}_{easy})\}$; Pick the last corresponding \mathcal{K}_{easy} samples from \mathcal{Q}
- 09 $\mathcal{Q}_t^{mod} \leftarrow \{\mathcal{U}_{t,i} | inc_i \in \text{argmidK}(\mathcal{Q}, \mathcal{K}_{mod})\}$; Pick the corresponding $(1 + |\mathcal{U}_t|/2 \text{ to } \mathcal{K}_{mod} + |\mathcal{U}_t|/2)$ samples from \mathcal{Q}
- 10 $\mathcal{Q}_t^{hard} \leftarrow \{\mathcal{U}_{t,i} | inc_i \in \text{argtopK}(\mathcal{Q}, \mathcal{K}_{hard})\}$; Pick the first corresponding \mathcal{K}_{hard} samples from \mathcal{Q}
- 11 $\mathcal{Q}_t = \mathcal{Q}_t^{easy} \cup \mathcal{Q}_t^{mod} \cup \mathcal{Q}_t^{hard}$
- 12 // Phase 3: Sample Annotation
- 13 $\mathcal{Y}_t \leftarrow \text{annotate } \mathcal{Q}_t$
- 14 $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1} \cup \{(\mathcal{X}, \mathcal{Y}) | \mathcal{X} \in \mathcal{Q}_t, \mathcal{Y} \in \mathcal{Y}_t\}$
- 15 // Phase 4: Update Model
- 16 $\theta_{1,head}^{*t+1} \leftarrow \text{fine-tuning } \theta_{1,head}^{*t}$ using \mathcal{L}_t
- 17 $\mathcal{U}_{t+1} \leftarrow \mathcal{U}_t \setminus \mathcal{Q}_t$
- 18 **if** $|\mathcal{U}_{t+1}| = 0$
- 19 **break**
- 20 **end for**
- 21 **return** \mathcal{L}_{t+1}

a group of selected clients participating in the local model training. Thirdly, the trained model parameters are collected and averaged at the FL server. Lastly, this process repeats until it reaches the threshold of N_e . The entire FL framework

Algorithm 4 Train *Task 1* or *Task 2* using Federated Learning

Input: Initial Model, $\theta_{1,head}^*$ or $\theta_{2,head}$
Number of Communication Round, N_c
Total Number of Clients, K

Output: Trained Model, $\theta_{1,head}^*$ or $\theta_{2,head}$

- 1 **If** *Task 1*, set $\theta = \theta_{1,head}^*$; else, set $\theta = \theta_{2,head}$
- 2 **Server executes:**
- 3 Initialize a global model, θ^{global}
- 4 **for** $t = 1 : N_c$ **do**
- 5 **Operations on the server side:**
- 6 // Select a fraction of Clients, C
- 7 $m \leftarrow \max(C \cdot K, 1)$
- 8 $S_t \leftarrow \{\text{random set of } m \text{ clients}\}$
- 9 // Train each selected client, θ^k
- 10 **for** each client $k \in S_t$ **in parallel do**
- 11 $\theta_{t+1}^{global} \leftarrow \text{ClientUpdate}(\theta_t^{global})$
- 12 **end for**
- 13 $\theta_{t+1}^{global} \leftarrow \sum_{k=1}^K \frac{n_k}{n} \theta_{t+1}^k$
- 14 **end for**
- 15 **ClientUpdate**(θ^{global}):
- 16 // Train the client model using local dataset
- 17 $\theta \leftarrow \theta^{global}$
- 18 update θ using any preferred strategy
- 19 **return** θ

is implemented using the OpenFL [55]. It is a Python 3 open-source FL framework that supports many real world applications such as medical imaging [39], [56], [57].

D. INFERENCE OPTIMIZATION

Once the individual head models for *Task 1* and *Task 2* are trained, they are merged into a unified model. Given the heterogeneity of IoT devices, it is favourable to accelerate the inference in such a way that the same optimized model can be executed across different hardware. OpenVINO is a promising candidate to meet these portability requirements. It calibrates the model for execution on several hardware types including Intel CPU, Intel Integrated GPU, Intel FPGA, and Intel Movidius Neural Compute Stick 2 (NCS2). Overall, OpenVINO involves two major steps as follows.

1. **Model Optimizer:** It converts the trained model into an OpenVINO format, known as intermediate representation (IR). IR consists of two files (*.xml + *.bin). The former and the latter contain the network topology and model weights, respectively.
2. **Inference Engine:** It is a C++ library with a set of C++ classes to infer input data (images) and obtain a result. The C++ library provides an API to read the IR, set the input and output formats, and execute the model on target devices.

IV. EXPERIMENT, RESULTS, AND DISCUSSIONS

A. DATASETS

The datasets used in *Task 1* and *Task 2* are listed in Tables 2 and 3, respectively. All images are extracted from CrisisIBD [16]. For *Task 1* dataset, those events related to

TABLE 2. Data split for disaster types.

Class Label	Train	Validation	Test
Fire	1270	121	280
Hurricane	1444	175	352
Flood	2336	266	599
Earthquake	2058	207	404
Landslide	940	123	268
Other Disaster	1132	143	302
Not Disaster	3666	435	990
Total	12846	1470	3195

TABLE 3. Data split for victim detection.

Class Labels	Count
Train	5994
Validation	634
Test	1448
Total	8076

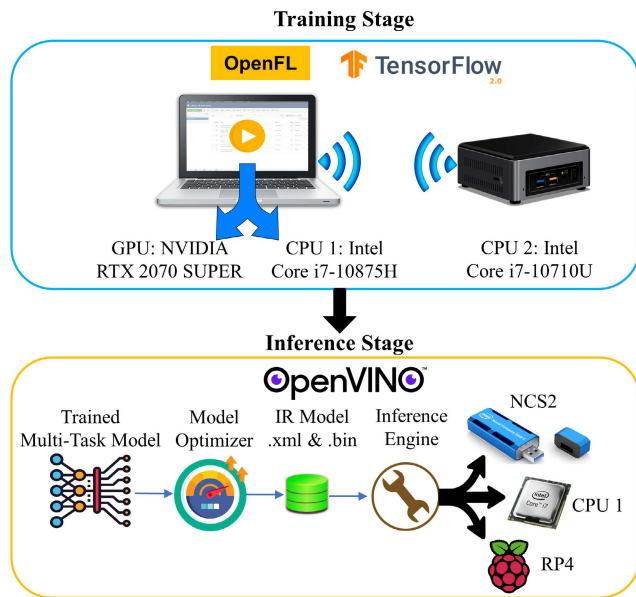


FIGURE 5. Experimental setup.

road accident, plane crash, explosion, and war are classified as “other disaster”. For *Task 2* dataset, additional annotation efforts are required since there is a lack of publicly available victim detection datasets. Specifically, we identify those images containing victims from [16] and generate bounding boxes via a combination of automatic [58] and manual annotations.

B. EXPERIMENTAL SETUP

Fig. 5 depicts the experiment with following setup.

1. Training phase: A maximum of three FL clients ($K = 3$) can be instantiated by OpenFL. A workstation consists of an Intel core i7 processor with 2.30GHz, 64 GB of DDR4 RAM memory and NVIDIA RTX 2070 SUPER. The workstation hosts two FL clients whereas

the remaining client is executed at an Intel NUC with an Intel core i7 processor with 4.70GHz and 64 GB of DDR4 RAM memory. This yields a sum of one Tensorflow GPU and two Tensorflow CPU operators. During the FL training, these two hardware are connected via Wi-Fi and model weights are shared for each communication epoch. Clearly, the local training completion time differs for each FL client and model aggregation can be initiated once all participating FL clients finish their tasks. Without loss of generality, we made the following assumptions:

- All FL clients always participate in each round
 - All FL clients train *Task 1* and *Task 2*
 - The workstation concurrently acts as the FL server
2. Inference phase: We calibrate the model to a variety of IR format, ranging from single-precision floating-point (FP32), through half-precision floating-point (FP16) to unsigned integer value (INT8). Obviously, the lower the quantization bits, the higher the throughput capacity. These models are benchmarked over three hardware: CPU 1, NCS2 and Raspberry Pi 4 (RP4) via OpenVINO DL Workbench. NCS2 is a dedicated hardware accelerator for inference with ultra-low power consumption. The great power savings, however, is accompanied by two limitations: (i) it can run only FP16 mode and (ii) it does not support the NMS feature.

C. TRAINING STAGE

$\theta_{1,head}^*$ is trained using the Cosine Decay strategy. Different from *Task 2*, we train FL model of *Task 1* in combination with offline AL technique, as proposed in Algorithm 3. This implies that the FL phase will only commence after the completion of AL at each client. We do not use the online AL mode in order to bypass the time-consuming round-by-round sample selection in FL [8].

For *Task 2*, we adopt the gradient accumulation strategy to facilitate the training at edge level. The hyperparameter for both *Task 1* and *Task 2* are tabulated in Table 4.

D. RESULTS AND DISCUSSIONS

1) RSA SIMILARITY

Firstly, we validate our hypothesis that the optimized head model $\theta_{1,head}^*$ consists of two IRBs, followed by the remaining blocks, and its optimal branching location is at the 94th location of $\theta_{2,backbone}$. To do so, we select 200 images from CrisisIBD and use equation (2) to compute the r_s for each possible combination of $\theta_{1,base}$ (where $N_{IRB} = 1, 2, 3$) and $\theta_{2,backbone}$ (82nd, 94th and 106th Layer), as shown in Table 5. It can be observed that all the r_s score at 82nd layer has the lowest value. This makes sense as the feature maps produced at this level are considerably too low-level. On the other hand, the highest score can be identified at 94th layer, instead of 106th layer. One possible reason is that the feature maps generated by this deepest layer are highly specialized for victim detection.

TABLE 4. Important hyperparameter for Task 1 and Task 2.

Parameters		Values
Task 1	Max AL round, \mathcal{N}_a	32
	AL Seed, $ \mathcal{L}_0 $	700
	Size of AL Labeled Dataset in Each Client, $ \mathcal{L}_t $	2716
	Query Batch Size, $(\mathcal{K}_{easy}, \mathcal{K}_{mod}, \mathcal{K}_{hard})$	21
	Number of epoch, \mathcal{N}_{t1}	40
	Number of Communication Round, \mathcal{N}_c	40
	Batch Size	32
	Initial Learning Rate, α	5×10^{-3}
	Size of Local Dataset in Each Client (2-Client Setup)	6423
	Size of Local Dataset in Each Client (3-Client Setup)	4282
Task 2	Number of epochs, \mathcal{N}_{t2}	20
	Mini Batch Size	8
	Mini Batch Gradient Accumulation Round, \mathcal{B}	8
	Initial Learning Rate, α	5×10^{-3}
	Size of Local Dataset in Each Client (2-Client Setup)	2997
	Size of Local Dataset in Each Client (3-Client Setup)	1998

These explanations are justified by using Grad-CAM [59] to visualize the activation maps as shown in Fig. 6. We limit our analysis to $N_{IRB} = 2$ since this configuration gives the best result in Table 5. A direct inspection suggests that among all three layers, $\theta_{2,backbone}$ (94th Layer) at Fig. 6 (b) yields the highest similarity with $\theta_{1,head}$ ($N_{IRB} = 2$) at Fig. 6 (d). This indicates that $\theta_{2,backbone}$ at this layer still preserves the meaningful semantic background knowledge needed for disaster scene classification.

To prove that higher task similarity leads to better branching selection, we first attach $\theta_{1,head}$ to $\theta_{2,backbone}$ for a total of nine combinations as shown in Table 5 and retrain $\theta_{1,head}$ for Task 1. Then, the computed F1 score is displayed in Table 6. It can be observed that the performance of F1 score is generally consistent with that of r_s , where both optimal points lie at the same location.

2) TASK 1: DISASTER CLASSIFICATION

The performance of the CL-trained $\theta_{1,head}^*$ is compared to the benchmarks provided by [17]. Note that their reported results stem from several single-task CNN models that are trained exclusively for Task 1. Also, for fair comparisons, we retrain the entire MobileNetv2 in our environment (labelled as MobileNetv2*). Table 7 compares the performance from four perspectives.

Among all models, the most closely related model is the MobileNetv2* since $\theta_{1,head}^*$ inherits similar network structure. Interestingly, the ability to distinguish disasters on top of a victim-detection model does not jeopardize the classification performance. In fact, it achieves 1-2% of performance gain, in terms of accuracy, precision, recall and F1 score. The rationale behind this is that $\theta_{2,backbone}$ has a denser network than MobileNetv2* to learn Task 1. Quantitatively speaking,

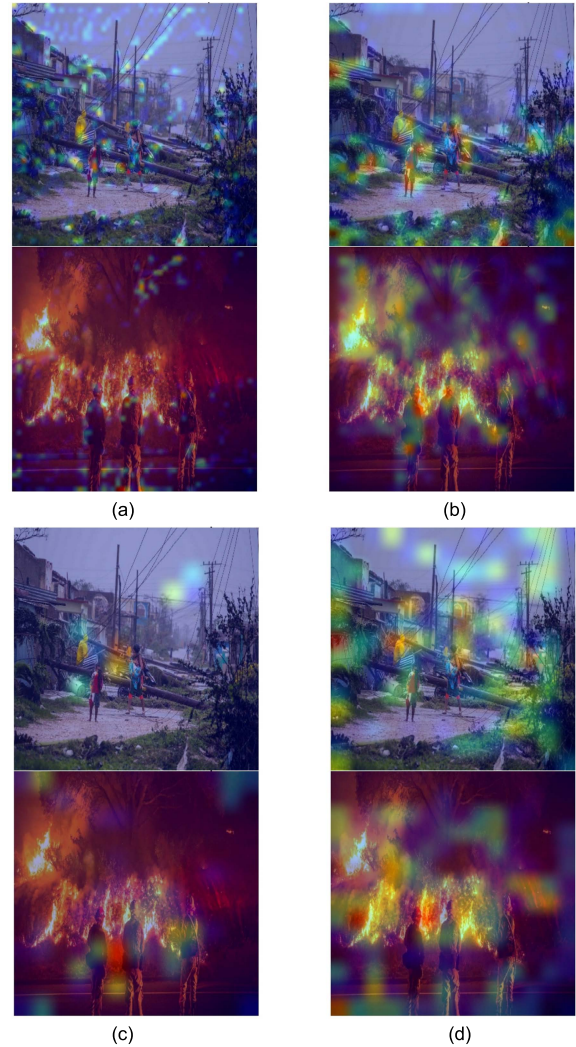


FIGURE 6. Grad-CAM visualization of activation maps. (a) $\theta_{2,backbone}$ (82nd Layer.) (b) $\theta_{2,backbone}$ (94th Layer.) (c) $\theta_{2,backbone}$ (106th Layer.) (d) $\theta_{2,base}$ ($N_{IRB} = 2$.)

TABLE 5. Similarity (r_s) between each $\theta_{1,base}$ and $\theta_{2,backbone}$.

$\theta_{1,base} \backslash \theta_{2,backbone}$	82 nd Layer	94 th Layer	106 th Layer
$N_{IRB} = 1$	0.195	0.343	0.393
$N_{IRB} = 2$	0.366	0.490	0.487
$N_{IRB} = 3$	0.338	0.408	0.422

TABLE 6. F1 Score of $\theta_{1,head}$ on top of each $\theta_{2,backbone}$ after retraining.

$\theta_{1,head} \backslash \theta_{2,backbone}$	82 nd Layer	94 th Layer	106 th Layer
$N_{IRB} = 1$	0.755	0.761	0.764
$N_{IRB} = 2$	0.769	0.792	0.782
$N_{IRB} = 3$	0.759	0.765	0.759

the total parameters of $\theta_{2,backbone}$ is 6.6x more than that of $\theta_{1,base}$ ($N_{IRB} = 2$).

A direct comparison from Table 7 suggests that EfficientNetb1 [60] will be always the best choice. However, another important factor in model selection is the computational

TABLE 7. CL-Trained head model ($\theta_{1,head}^*$) Vs. Benchmarks in [17]. MobileNetv2* was Retrained in the same environment as $\theta_{1,head}^*$ to ensure fair comparisons.

Backbone	Accuracy	Precision	Recall	F1 Score
ResNet101	0.819	0.815	0.816	0.816
AlexNet	0.755	0.753	0.753	0.753
VGG16	0.803	0.797	0.798	0.798
SqueezeNet	0.726	0.719	0.717	0.717
InceptionNetv2	0.808	0.801	0.802	0.802
MobileNetv2	0.793	0.788	0.793	0.789
EfficientNetb1	0.838	0.834	0.838	0.835
MobileNetv2*	0.776	0.787	0.776	0.781
$\theta_{1,head}^*$	0.792	0.796	0.792	0.792

TABLE 8. Comparison between the disaster classification head models trained via CL, FL and AL-FL. Methods labelled with an Asterisk (*) are trained using 3 FL clients.

Method	Accuracy	Precision	Recall	F1 Score
CL (all data)	0.792	0.796	0.792	0.792
CL (1/2 data)	0.754	0.757	0.752	0.743
CL (1/3 data)	0.732	0.739	0.727	0.721
FL (2 clients)	0.800	0.805	0.794	0.793
FL (3 clients)	0.796	0.800	0.790	0.788
AL-FL hard*	0.719	0.720	0.720	0.720
AL-FL mod/hard*	0.722	0.731	0.715	0.740
AL-FL easy/mod/hard*	0.767	0.774	0.759	0.758

TABLE 9. Average precision (AP) comparison for Task 2.

Method	Average Precision
CL (all data)	0.694
CL (1/2 data)	0.467
CL (1/3 data)	0.400
FL (2 clients)	0.590
FL (3 clients)	0.542

efficiency, which is ignored in [17]. In fact, MobileNetv2 has less than doubled the parameters required by EfficientNetb1 [61]. Nevertheless, there exist some state-of-the-art models with high accuracy and yet fast processing such as CustomNet [62]. We argue that our proposed branching strategy is also applicable to these models, provided that the task similarity between two merging candidate networks is good enough. Overall, our solution is considered robust given that it can handle two tasks.

So far, $\theta_{1,head}^*$ as tabulated in Table 7 is a CL-trained model. In Table 8, we will use this as the benchmark (labelled as ‘‘CL (all data)’’) with respect to the FL and AL-FL performance. We also consider two scenarios (‘‘CL (1/2 data)’’ and ‘‘CL (1/3 data)’’) where IoT devices individually train the

TABLE 10. Network model size comparison.

	Conventional Approach	Proposed Method
Network Structure	$\theta_{1,ori} + \theta_2$	$\theta_{1,head}^* + \theta_2$
Full Model Size (MB)	27.6 + 247 = 274.6	14.8 + 247 = 261.8
Trainable Network	$\theta_{1,ori} + \theta_{2,head}$	$\theta_{1,head}^* + \theta_{2,head}$
Task 1 Trainable Model Size, s_1 (MB)	27.6	14.8
Task 2 Trainable Model Size, s_2 (MB)	84.0	84.0

TABLE 11. Model inference speed (FPS) before and after model optimization via OpenVINO toolkit.

Hardware	Framework	Data Format	FPS
GPU	TensorFlow GPU	FP32	20.31
CPU1	TensorFlow CPU	FP32	6.55
CPU1	OpenVINO IR Format	FP16	9.37
NCS2 on CPU1	OpenVINO IR Format without NMS	FP16	2.50
NCS2 on RP4	OpenVINO IR Format without NMS	FP16	1.80
NCS2 on RP4 (Conventional Approach)	OpenVINO IR Format without NMS	FP16	1.52
CPU1	OpenVINO IR Format	INT8	16.46

model without sharing their model weights. As expected, the individual training of each device yields inferior results due to limited dataset.

Surprisingly, it can be noticed that FL outperforms CL in both 2-client and 3-client settings. For instance, FL with 2-client and 3-client outperform CL by 1.13% and 0.50% in precision, respectively. This is a very encouraging result from a system design point of view and such performance trend is aligned with the findings in [44] and [63].

Among all the AL-FL variations, the best performer is the proposed heuristic, which picks a combination of easy, mod, and hard samples. It approximates the CL model within 3.40% F1 score gap while using 36.57% less labelled dataset.

3) TASK 2: VICTIM DETECTION

Since $\theta_{2,head}$ is trained with a custom dataset, there is no benchmark to compare the results with. We consider similar settings as in *Task 1*, except for the AL approach. Table 9 compares the results of $\theta_{2,head}$ trained on each setting. This time, it can be observed that the FL approach is weaker than the CL method for *Task 2*. The performance loss is likely attributed to the scarcity of training dataset [8]. In FL mode, *Task 2* clients has a maximum of 2997 images, which is less than half of the 6423 images used in *Task 1*. Nevertheless, the FL approaches outperform their distributed learning counterparts by up to

TABLE 12. Model accuracy and AP before and after model optimization via OpenVINO toolkit.

Framework	Data Format	Accuracy	AP
TensorFlow (GPU/CPU1)	FP32	0.792	0.694
OpenVINO IR Format	FP16	0.793	0.696
OpenVINO IR Format without NMS + custom NMS	FP16	0.793	0.696
OpenVINO IR Format	INT8	0.796	0.680

35%. These results again highlight the importance of sharing model weights for better performance.

4) BENEFITS OF USING PROPOSED MODEL IN FL ENVIRONMENT

Table 10 compares the actual parameter size between conventional and proposed methods. It can be observed that the proposed model saves about 11.3% of the transmission payload for every communication round \mathcal{N}_c . To train a specific task in an FL environment, the total size of model weights $w_{1/2}$ needed to exchange with a FL server can be calculated as follows:

$$w_{1/2} = \mathcal{N}_c \times K \times s_{1/2} \tag{3}$$

5) INFERENCE RESULTS VIA DL WORKBENCH

To ensure reusability, interoperability, and scalability, we measure the inference results via the DL workbench tool. Table 11 compares the speed in terms of FPS among three hardware as mentioned Fig. 5.

As expected, the highest inference speed is attained by the powerful GPU mode. A direct deployment in the CPU 1 will drastically drop from 20.31 to 6.55 FPS. This unveils the need of using OpenVINO models. Under the same hardware and data format, the optimized model achieves 43% of FPS gain. The speed can be further boosted to 151 % by using INT8 IR model. For NCS2, the performance tradeoff is visible through the reported FPS value of 2.50. The FPS stemming from plugging the NCS2 into less powerful RP4 further drops to 1.8. However, this is acceptable since NCS2 consumes power of only 1.5 W [64], which is important in establishing sustainable IoT solutions. To reveal more insight, we also convert the models in conventional approach into two separate OpenVINO models. A sequential execution of these two models on RP4 results in another FPS slowdown of 18%.

At this point, it is important to determine how much is the accuracy and precision drop. Since the inference model is multi-tasking, Table 12 compares both classification (accuracy) and detection (AP) related metrics. At first glance, all the accuracy accrued by OpenVINO models surprisingly outperforms the original TensorFlow model. An in-depth analyse reveals that such trend conforms to the OpenVINO mechanism. In fact, the OpenVINO model optimizer uses 20% of the test dataset during the model calibration. Similar performance trend can be observed for

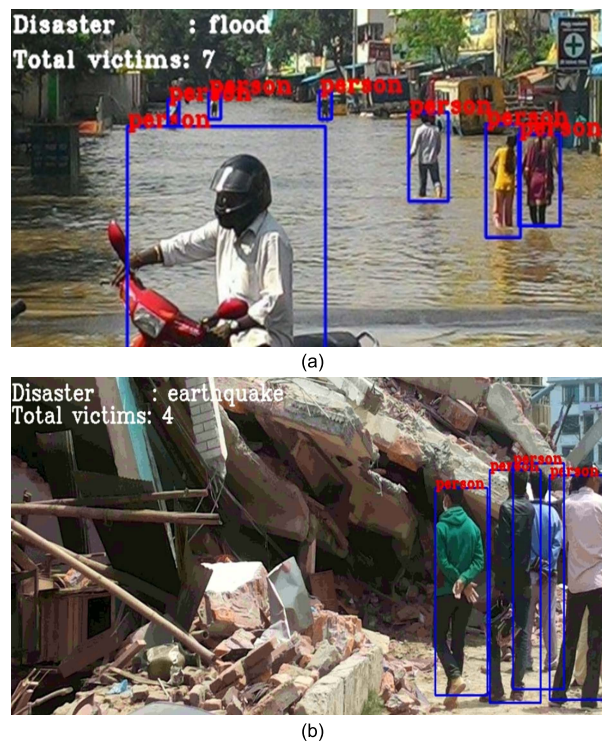


FIGURE 7. Inference output of the multi-task model at different area. (a) Flood. (b) Earthquake. The joint disaster classification and victim count prediction are labeled at the top left corner of the input images.

AP of Task 2. Overall, the MTL model performance is retained after optimization and such encouraging results will promote the IoT deployment. Note that we wrote a custom Python 3 NMS code to complement the OpenVINO IR Format without NMS.

V. CONCLUSION

In this paper, we have devised a MTL model that performs joint disaster classification and victim detection. Our two merging CNN networks are MobileNetv2 and YOLOv3, which can be trained separately. Through rigorous mathematical analysis, we proved that optimal branching location and the number of IRBs are 94th layer and two, respectively. As compared to the conventional approach, the proposed model has lesser memory requirements and better classification-related results, while preserving the same detection-related performance. The first advantage would be very useful in IoT environment, where the data (e.g., network weights) are exchanged. We showed that AL and FL can complement each other to bring positive impact to the IoT scenario, where massive data is generated within different devices and requires exhaustive human annotation efforts. As a proof of concept, we implemented our solution onto different hardware by utilizing several open-source and production-ready tools. Even for the low-cost and low-powered Raspberry Pi 4, the proposed method can still reach up to 1.8 FPS, which is 18% faster than the conventional method.

Three potential directions have been identified as our future works. Firstly, the communication between each FL client and server is based on Wi-Fi technology, which has transmission distance limitation. An alternative of long-distance wireless technology such as LoRa and NB-IOT can be considered. Secondly, the existing Wi-Fi implementation operates in star topology, which is vulnerable to disaster damage. Therefore, we need to explore a disaster-resilient mesh network. Thirdly, the FL approach always requires all clients to train their own models for every communication round. In practice, some devices may have limited computational capacity, scarce dataset and poor channel conditions. Therefore, we need to select a subset of FL clients in each round more efficiently.

ACKNOWLEDGMENT

The ASEAN IVO (http://www.nict.go.jp/en/asean_ivo/index.html) project, Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network, was involved in the production of the contents of this work and financially supported by NICT (<http://www.nict.go.jp/en/index.html>).

REFERENCES

- [1] S. Evans. (2021). *Claims Paid for Japan's M7 Quake in Feb. 2021 Nearing 900 m*. Accessed: Oct. 14, 2021. [Online]. Available: <https://www.artemis.bm/news/claims-paid-for-japans-m7-quake-in-feb-2021-nearing-900m/>
- [2] (2017). United Nations Office for the Coordination of Humanitarian Affairs. *Five Essentials for the First 72 Hours of Disaster Response*. OCHA. Accessed: Aug. 20, 2021. [Online]. Available: <https://www.unocha.org/story/five-essentials-first-72-hours-disaster-response>
- [3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
- [4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015, *arXiv:1506.02640*.
- [6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37, doi: [10.1007/978-3-319-46448-0_2](https://doi.org/10.1007/978-3-319-46448-0_2).
- [7] A. Saeed, F. D. Salim, T. Ozelebi, and J. Lukkien, "Federated self-supervised learning of multisensor representations for embedded intelligence," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1030–1040, Jan. 2021, doi: [10.1109/JIOT.2020.3009358](https://doi.org/10.1109/JIOT.2020.3009358).
- [8] L. Ahmed, K. Ahmad, N. Said, B. Qolomany, J. Qadir, and A. Al-Fuqaha, "Active learning based federated learning for waste and natural disaster image classification," *IEEE Access*, vol. 8, pp. 208518–208531, 2020, doi: [10.1109/ACCESS.2020.3038676](https://doi.org/10.1109/ACCESS.2020.3038676).
- [9] P. Chhikara, R. Tekchandani, N. Kumar, M. Guizani, and M. M. Hassan, "Federated learning and autonomous UAVs for hazardous zone detection and AQI prediction in IoT environment," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15456–15467, Oct. 2021, doi: [10.1109/JIOT.2021.3074523](https://doi.org/10.1109/JIOT.2021.3074523).
- [10] Intel. *Intel Distribution of OpenVINOTM Toolkit*. Accessed: Mar. 15, 2022. [Online]. Available: <https://www.intel.com/content/www/us/en/developer/tools/opencvino-toolkit/overview.html>
- [11] A. Demidovskij, A. Tugaryov, A. Suvorov, Y. Tarkan, M. Fatekhov, I. Salnikov, A. Kashchikhin, V. Golubenko, G. Dedyukhina, A. Alborova, R. Palmer, M. Fedorov, and Y. Gorbachev, "OpenVINO deep learning workbench: A platform for model optimization, analysis and deployment," in *Proc. IEEE 32nd Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2020, pp. 661–668, doi: [10.1109/ICTAI50040.2020.00106](https://doi.org/10.1109/ICTAI50040.2020.00106).
- [12] M.-L. Tham, Y. J. Wong, B. H. Kwan, Y. Owada, M. M. Sein, and Y. C. Chang, "Joint disaster classification and victim detection using multi-task learning," in *Proc. IEEE 12th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Dec. 2021, pp. 407–412, doi: [10.1109/UEMCON53757.2021.9666576](https://doi.org/10.1109/UEMCON53757.2021.9666576).
- [13] H. Mouzannar, Y. Rizk, and M. Awad, "Damage identification in social media posts using multimodal deep learning," in *Proc. Int. ISCRAM Conf.*, May 2018, pp. 529–543.
- [14] F. Alam, F. Ofli, and M. Imran, "CrisisMMD: Multimodal Twitter datasets from natural disasters," in *Proc. 12th Int. AAAI Conf. Web Social Media (ICWSM)*, 2018, pp. 465–473, [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85050637466&partnerID=40&md5=0fb528332fb3182d641214df5e854665>
- [15] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg, "AIDR: Artificial intelligence for disaster response," in *Proc. 23rd Int. Conf. World Wide Web*, Apr. 2014, pp. 159–162, doi: [10.1145/2567948.2577034](https://doi.org/10.1145/2567948.2577034).
- [16] F. Alam, F. Ofli, M. Imran, T. Alam, and U. Qazi, "Deep learning benchmarks and datasets for social media image classification for disaster response," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Dec. 2020, pp. 151–158, doi: [10.1109/ASONAM49781.2020.9381294](https://doi.org/10.1109/ASONAM49781.2020.9381294).
- [17] F. Alam, T. Alam, M. Imran, and F. Ofli, "Robust training of social media image classification models for rapid disaster response," 2021, *arXiv:2104.04184*.
- [18] D. R. Hartawan, T. W. Purboyo, and C. Setianingsih, "Disaster victims detection system using convolutional neural network (CNN) method," in *Proc. IEEE Int. Conf. Ind. 4.0, Artif. Intell., Commun. Technol. (IAICT)*, Jul. 2019, pp. 105–111, doi: [10.1109/IAICT.2019.8784782](https://doi.org/10.1109/IAICT.2019.8784782).
- [19] M. I. Perdana, A. Risnumawan, and I. A. Sulistijono, "Automatic aerial victim detection on low-cost thermal camera using convolutional neural network," in *Proc. Int. Symp. Community-Centric Syst. (CcS)*, Sep. 2020, pp. 1–5, doi: [10.1109/CCS49175.2020.9231433](https://doi.org/10.1109/CCS49175.2020.9231433).
- [20] F. B. Jaradat and D. Valles, "A victims detection approach for burning building sites using convolutional neural networks," in *Proc. 10th Annu. Commun. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 280–286, doi: [10.1109/CCWC47524.2020.9031275](https://doi.org/10.1109/CCWC47524.2020.9031275).
- [21] K. Naveen, K. N. Lokesh, K. PM, M. SC, and K. Prachetha, "Early Flood Detection and Disaster Victim Detection," *Int. J. Sci. Technol. Eng.*, vol. 7, no. 1, pp. 11–17, Aug. 2020. [Online]. Available: <http://ijste.org/Article.php?manuscript=IJSTE711003>
- [22] V. L. Nguyen, M. H. Shaker, and E. Hüllermeier, "How to measure uncertainty in uncertainty sampling for active learning," *Mach. Learn.*, vol. 111, no. 1, pp. 89–122, Jan. 2022, doi: [10.1007/S10994-021-06003-9/FIGURES/13](https://doi.org/10.1007/S10994-021-06003-9/FIGURES/13).
- [23] M.-L. Tham and W. K. Tan, "IoT based license plate recognition system using deep learning and OpenVINO," in *Proc. 4th Int. Conf. Sensors, Signal Image Process.*, Oct. 2021, pp. 7–14, doi: [10.1145/3502814.3502816](https://doi.org/10.1145/3502814.3502816).
- [24] E. Izutov, "Fast and accurate person re-identification with RMNet," 2018, *arXiv:1812.02465*.
- [25] D. Brown, "Mobile attendance based on face detection and recognition using OpenVINO," in *Proc. Int. Conf. Artif. Intell. Smart Syst. (ICAIS)*, Mar. 2021, pp. 1152–1157, doi: [10.1109/ICAIS50930.2021.9395836](https://doi.org/10.1109/ICAIS50930.2021.9395836).
- [26] S. Bernabe, C. Gonzalez, A. Fernandez, and U. Bhargale, "Portability and acceleration of deep learning inferences to detect rapid earthquake damage from VHR remote sensing images using Intel OpenVINO toolkit," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 6906–6915, 2021, doi: [10.1109/JSTARS.2021.3075961](https://doi.org/10.1109/JSTARS.2021.3075961).
- [27] D. T. Nguyen, F. Ofli, M. Imran, and P. Mitra, "Damage assessment from social media imagery data during disasters," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Jul. 2017, pp. 569–576.
- [28] A. Sharma and U. Verma, "Flood magnitude assessment from UAV aerial videos based on image segmentation and similarity," in *Proc. IEEE Region 10th Conf. (TENCON)*, Dec. 2021, pp. 476–481, doi: [10.1109/TENCON54134.2021.9707250](https://doi.org/10.1109/TENCON54134.2021.9707250).
- [29] M. Rahnemoonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. R. Murphy, "FloodNet: A high resolution aerial imagery dataset for post flood scene understanding," *IEEE Access*, vol. 9, pp. 89644–89654, 2021, doi: [10.1109/ACCESS.2021.3090981](https://doi.org/10.1109/ACCESS.2021.3090981).
- [30] M. Hong and R. Akerkar, "Victim detection platform in IoT paradigm," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 3, pp. 1–14, Feb. 2021, doi: [10.1002/CPE.5254](https://doi.org/10.1002/CPE.5254).
- [31] Y. Yamazaki, C. Premachandra, and C. J. Perea, "Audio-Processing-Based human detection at disaster sites with unmanned aerial vehicle," *IEEE Access*, vol. 8, pp. 101398–101405, 2020, doi: [10.1109/ACCESS.2020.2998776](https://doi.org/10.1109/ACCESS.2020.2998776).

- [32] R. Avanzato and F. Beritelli, "A smart UAV-femtocell data sensing system for post-earthquake localization of people," *IEEE Access*, vol. 8, pp. 30262–30270, 2020, doi: [10.1109/ACCESS.2020.2972699](https://doi.org/10.1109/ACCESS.2020.2972699).
- [33] C. Dorn, A. Depold, F. Lurz, S. Erhardt, and A. Hagelauer, "UAV-based localization of mobile phones for search and rescue applications," in *Proc. IEEE 22nd Annu. Wireless Microw. Technol. Conf. (WAMICON)*, Apr. 2022, pp. 1–4, doi: [10.1109/WAMICON53991.2022.9786189](https://doi.org/10.1109/WAMICON53991.2022.9786189).
- [34] Y. Qian, J. M. Dolan, and M. Yang, "DLT-Net: Joint detection of drivable areas, lane lines, and traffic objects," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4670–4679, Nov. 2020, doi: [10.1109/TITS.2019.2943777](https://doi.org/10.1109/TITS.2019.2943777).
- [35] C. Wen, H. Zhang, H. Li, H. Li, J. Chen, H. Guo, and S. Cheng, "Multi-scene citrus detection based on multi-task deep learning network," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2020, pp. 912–919, doi: [10.1109/SMC42975.2020.9282909](https://doi.org/10.1109/SMC42975.2020.9282909).
- [36] W. Zhang, K. Wang, Y. Wang, L. Yan, and F.-Y. Wang, "A loss-balanced multi-task model for simultaneous detection and segmentation," *Neurocomputing*, vol. 428, pp. 65–78, Mar. 2021, doi: <https://doi.org/10.1016/j.neucom.2020.11.024>.
- [37] Y. Chen, D. Zhao, L. Lv, and Q. Zhang, "Multi-task learning for dangerous object detection in autonomous driving," *Inf. Sci.*, vol. 432, pp. 559–571, Mar. 2018, doi: [10.1016/j.ins.2017.08.035](https://doi.org/10.1016/j.ins.2017.08.035).
- [38] W. Lee, J. Na, and G. Kim, "Multi-task self-supervised object detection via recycling of bounding box annotations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4979–4988, doi: [10.1109/CVPR.2019.00512](https://doi.org/10.1109/CVPR.2019.00512).
- [39] E. Isik-Polat, G. Polat, A. Kocyigit, and A. Temizel, "Evaluation and analysis of different aggregation and hyperparameter selection methods for federated brain tumor segmentation," 2022, *arXiv:2202.08261*.
- [40] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," Oct. 2016, *arXiv:1610.05492*.
- [41] H. Xiao, J. Zhao, Q. Pei, J. Feng, L. Liu, and W. Shi, "Vehicle selection and resource optimization for federated learning in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11073–11087, Aug. 2022, doi: [10.1109/TITS.2021.3099597](https://doi.org/10.1109/TITS.2021.3099597).
- [42] L. U. Khan, M. Alsenwi, I. Yaqoob, M. Imran, Z. Han, and C. S. Hong, "Resource optimized federated learning-enabled cognitive Internet of Things for smart industries," *IEEE Access*, vol. 8, pp. 168854–168864, 2020, doi: [10.1109/ACCESS.2020.3023940](https://doi.org/10.1109/ACCESS.2020.3023940).
- [43] G. Wang, J. N. Hwang, C. Rose, and F. Wallace, "Uncertainty sampling based active learning with diversity constraint by sparse selection," in *Proc. IEEE 19th Int. Workshop Multimedia Signal Process. (MMSP)*, Nov. 2017, pp. 1–6, doi: [10.1109/MMSP.2017.8122269](https://doi.org/10.1109/MMSP.2017.8122269).
- [44] Z. Xiong, Z. Cheng, C. Xu, X. Lin, X. Liu, D. Wang, X. Luo, Y. Zhang, N. Qiao, M. Zheng, and H. Jiang, "Facing small and biased data dilemma in drug discovery with federated learning," *BioRxiv*, vol. 2020, pp. 1–18, Jan. 2020, doi: [10.1101/2020.03.19.998898](https://doi.org/10.1101/2020.03.19.998898).
- [45] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, *arXiv:1706.05098*.
- [46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520, doi: [10.1109/CVPR.2018.00474](https://doi.org/10.1109/CVPR.2018.00474).
- [47] K. Cai, X. Miao, W. Wang, H. Pang, Y. Liu, and J. Song, "A modified YOLOv3 model for fish detection based on MobileNetv1 as backbone," *Aquacultural Eng.*, vol. 91, Nov. 2020, Art. no. 102117, doi: [10.1016/j.aquaeng.2020.102117](https://doi.org/10.1016/j.aquaeng.2020.102117).
- [48] J.-A. Kim, J.-Y. Sung, and S.-H. Park, "Comparison of faster-RCNN, YOLO, and SSD for real-time vehicle type recognition," in *Proc. IEEE Int. Conf. Consum. Electron. Asia (ICCE-Asia)*, Nov. 2020, pp. 1–4, doi: [10.1109/ICCE-ASIA49877.2020.9277040](https://doi.org/10.1109/ICCE-ASIA49877.2020.9277040).
- [49] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, *arXiv:2207.02696*.
- [50] K. Dwivedi and G. Roig, "Representation similarity analysis for efficient task taxonomy & transfer learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12379–12388, doi: [10.1109/CVPR.2019.01267](https://doi.org/10.1109/CVPR.2019.01267).
- [51] N. Grimova and M. Macas, "Query-by-committee framework used for semi-automatic sleep stages classification," *Multidisciplinary Digit. Publishing Inst. Proc.*, vol. 31, p. 80, Nov. 2019, doi: [10.3390/PROCEEDINGS2019031080](https://doi.org/10.3390/PROCEEDINGS2019031080).
- [52] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019, doi: [10.1109/JSAC.2019.2904348](https://doi.org/10.1109/JSAC.2019.2904348).
- [53] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
- [54] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.
- [55] G. A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, J. Martin, B. Edwards, M. J. Sheller, S. Pati, P. N. Moorthy, S.-H. Wang, P. Shah, and S. Bakas, "OpenFL: An open-source framework for federated learning," 2021, *arXiv:2105.06413*.
- [56] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries* (Lecture Notes in Computer Science), vol. 11383. Cham, Switzerland: Springer, 2019, pp. 92–104, doi: [10.1007/978-3-030-11723-8_9](https://doi.org/10.1007/978-3-030-11723-8_9).
- [57] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, and S. Bakas, "Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data," *Sci. Rep.*, vol. 10, no. 1, pp. 1–12, Jul. 2020, doi: [10.1038/s41598-020-69250-1](https://doi.org/10.1038/s41598-020-69250-1).
- [58] M. Hamzah. (2020). *Auto-Annotate: Automatically Annotate Your Entire Image Directory by a Single Command*, GitHub Repository. [Online]. Available: <https://github.com/mdhzm1/Auto-Annotate>
- [59] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626, doi: [10.1109/ICCV.2017.74](https://doi.org/10.1109/ICCV.2017.74).
- [60] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, May 2019, pp. 10691–10700.
- [61] R. Chaganti, V. Ravi, and T. D. Pham, "Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification," *J. Inf. Secur. Appl.*, vol. 69, Sep. 2022, Art. no. 103306, doi: [10.1016/J.JISA.2022.103306](https://doi.org/10.1016/J.JISA.2022.103306).
- [62] A. S. Winoto, M. Kristianus, and C. Premachandra, "Small and slim deep convolutional neural network for mobile device," *IEEE Access*, vol. 8, pp. 125210–125222, 2020, doi: [10.1109/ACCESS.2020.3005161](https://doi.org/10.1109/ACCESS.2020.3005161).
- [63] M. Asad, A. Moustafa, and T. Ito, "Federated learning versus classical machine learning: A convergence comparison," 2021, *arXiv:2107.10976*.
- [64] L. Libutti, F. Igual, L. Piñuel, L. C. De Giusti, and M. Naiouf, "Benchmarking performance and power of USB accelerators for inference with MLPerf," in *Proc. 2nd Workshop Accelerated Mach. Learn. (AccML)*, 2020, pp. 1–15.



YI JIE WONG received the B.Eng. degree (Hons.) in biomedical engineering from the Universiti Tunku Abdul Rahman, Sungai Long, Malaysia, in 2022, where he is currently pursuing the Ph.D. degree in digital technology with specialization of reinforcement learning-based federated learning. His research interests include the Internet of Things (IoT), machine learning, federated learning, deep reinforcement learning, and resource allocation optimization.



MAU-LUEN THAM received the Bachelor of Engineering and Doctor of Philosophy degrees in telecommunication engineering from the University of Malaya. He is currently an Assistant Professor with Universiti Tunku Abdul Rahman. He has been a principal investigator (PI) and a co-investigator of more than 15 research and development projects. This includes five international grants, two of which are simultaneously led by him as the PI/Co-PI under the support of ASEAN IVO and the British Council. He has published two IEEE Transactions papers as a principal author. His research interests include the IoT, machine learning/deep learning/deep reinforcement learning, and beyond-5G communications.



BAN-HOE KWAN received the Bachelor of Engineering (Electrical) degree, the Master of Engineering Science degree, and the Ph.D. degree in engineering from the University of Malaya (UM). He is currently working at Universiti Tunku Abdul Rahman (UTAR) as an Assistant Professor. His research interests include image processing, artificial intelligence, medical signal processing, the Internet of Things, and robotics.



EZRA MORRIS ABRAHAM GNANAMUTHU received the B.Eng. degree from Bharathiar University, India, the M.E. degree from Anna University, India, and the Ph.D. degree from Multimedia University, Malaysia. He joined the Karunya Institute of Technology as a Lecturer, in 1993, before moving to Malaysia, in 1998. In 2008, he joined as an Assistant Professor with University Tunku Abdul Rahman (UTAR), Kuala Lumpur, Malaysia, and became an Associate Professor, in 2014. He has published over 40 papers in international journals and conferences. His research interests include digital signal processing, wireless ad hoc networks, optimization using PSO, GA/IGA, and mobile communication.



YASUNORI OWADA (Member, IEEE) received the Ph.D. degree from Niigata University. He is currently a Senior Researcher with the Resilient ICT Research Center, National Institute of Information and Communications Technology (NICT). He has been engaged in the research and development of resilient, distributed wireless, and mobile access network systems called NerveNet at NICT, since 2010. He was previously the President of Space-Time Engineering Japan Inc., from 2008 to 2010, and an Assistant Professor with Niigata University, from 2007 to 2008. He was awarded with Prizes for Science and Technology; and FY2019 the Commendation for Science and Technology by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

...

Article

FedDdrl: Federated Double Deep Reinforcement Learning for Heterogeneous IoT with Adaptive Early Client Termination and Local Epoch Adjustment

Yi Jie Wong ¹, Mau-Luen Tham ^{1,*}, Ban-Hoe Kwan ² and Yasunori Owada ³

¹ Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Kajang 43000, Malaysia

² Department of Mechatronics and Biomedical Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Kajang 43000, Malaysia

³ Resilient ICT Research Center, Network Research Institute, National Institute of Information and Communications Technology (NICT), Tokyo 184-8795, Japan

* Correspondence: thamml@utar.edu.my

Abstract: Federated learning (FL) is a technique that allows multiple clients to collaboratively train a global model without sharing their sensitive and bandwidth-hungry data. This paper presents a joint early client termination and local epoch adjustment for FL. We consider the challenges of heterogeneous Internet of Things (IoT) environments including non-independent and identically distributed (non-IID) data as well as diverse computing and communication capabilities. The goal is to strike the best tradeoff among three conflicting objectives, namely global model accuracy, training latency and communication cost. We first leverage the balanced-MixUp technique to mitigate the influence of non-IID data on the FL convergence rate. A weighted sum optimization problem is then formulated and solved via our proposed FL double deep reinforcement learning (FedDdrl) framework, which outputs a dual action. The former indicates whether a participating FL client is dropped, whereas the latter specifies how long each remaining client needs to complete its local training task. Simulation results show that FedDdrl outperforms the existing FL scheme in terms of overall tradeoff. Specifically, FedDdrl achieves higher model accuracy by about 4% while incurring 30% less latency and communication costs.

Keywords: federated learning; client selection; local epoch adjustment; deep reinforcement learning; Internet of Things



Citation: Wong, Y.J.; Tham, M.-L.; Kwan, B.-H.; Owada, Y. FedDdrl: Federated Double Deep Reinforcement Learning for Heterogeneous IoT with Adaptive Early Client Termination and Local Epoch Adjustment. *Sensors* **2023**, *23*, 2494. <https://doi.org/10.3390/s23052494>

Academic Editors: Raffaele Montella, José Luis González Compeán and Sokol Kosta

Received: 30 December 2022

Revised: 3 February 2023

Accepted: 7 February 2023

Published: 23 February 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the Internet of Things (IoT), each device can collect massive amounts of data (i.e., measurements and location information) [1]. It is estimated that IoT devices will generate over 90 zettabytes of data globally by 2025 [2]. These data can be uploaded to a centralized server, where a new model can be retrained or fine-tuned using the collected dataset. This method is called centralized learning (CL) since the data must be centralized at one location for model training. However, privacy concerns make it inconvenient for devices to share potentially sensitive data with a centralized server (or any other party). For example, medical images may contain sensitive and private information about patients [3], which prohibits the collection of such data from multiple healthcare institutions for CL. Additionally, uploading bandwidth-hungry data requires a high communication cost, which is not feasible for most IoT devices with network resource constraints [1,4]. In fact, the data collected by IoT devices could be larger than the model size [4], especially when dealing with image data.

Federated learning (FL) has emerged as one of the promising candidates to address this challenge. FL is a technique that trains an algorithm across multiple edge devices

holding individual local datasets without sharing or exchanging them. First, each IoT device (client) uses its locally collected data to train a local model. After training, each IoT device uploads its locally trained models to an FL server for aggregation. A new global model is generated, which is trained using data from all participating clients without actually sharing the sensitive and bandwidth-hungry data. Thus, FL addresses data privacy concerns by training a global model in distributed environments. FL has since been applied in various applications, ranging from mobile keyboard prediction [5] to natural disaster classification [6,7] and medical image segmentation [3]. Using FL, Google trained its mobile keyboard prediction using 600 million sentences from a surprising amount of 1.5 million clients [5]. In addition, Intel released its production-ready and open-source FL (OpenFL) framework [8]. OpenFL is used in the Federated Tumor Segmentation (FeTS) initiative, which is a program participated in by 56 clinical sites around the globe to train tumor segmentation models via FL. Experiments show that FL models can reach 99% of CL model without sharing the sensitive data [3]. These large-scale real-life applications proved the huge economic value of FL.

FL coupled with the IoT has huge potential for real-world application. For instance, research works [9–11] combined FL with industrial IoT (IIoT), creating an industrial-grade hierarchical FL framework. Hierarchical FL is a three-layer architecture FL framework composed of clients, edge servers and a cloud server. Regular FL is performed between the edge server and its corresponding client device. Upon model aggregation at the edge servers, the aggregated models are then uploaded to the cloud for global model aggregation. Experiments show hierarchical FL to be superior to a regular FL, with lower training latency and better convergence [12]. This is because the model aggregation at the client edge before global model aggregation can significantly reduce the training divergence. Due to the robustness of hierarchical FL, it has also been exploited to empower digital twins [10,13]. In this study, we only focus on regular FL, which is the fundamental building block for any sophisticated FL framework.

Despite huge potential, FL still faces several challenges from practical implementation: (1) model convergence in the presence of a non-independent and identically distributed (non-IID) dataset, (2) computing efficiency and (3) communication efficiency [14,15]. First, data is usually not uniformly distributed across IoT devices. Realistically, each IoT device has a unique data distribution and can be considered non-IID, whereas the global population (if the data is centralized) would be IID. According to [16], the earth mover's distance (EMD) between the local client data distribution and the global population is the main reason the FL model diverges from the global optima solution. This is also termed weight divergence between FL and CL models, which greatly reduces the convergence rate of FL models [16]. Additionally, the heterogeneity of computing and communication resources in IoT networks hinders resource utilization efficiency. In most studies, except [17,18], the local epoch number is set to be the same for all client devices disregarding their computing constraints. As a result, devices with stronger computing power often have to wait for the straggler devices to complete their training, which drastically increases the overall training latency. Moreover, some clients may not have access to high-speed networks, making local model uploading slow or unrealistic.

Many previous studies aimed to tackle the three challenges from different viewpoints. However, optimizing one of the objectives might deteriorate the other objectives [14]. Deep reinforcement learning (DRL) has recently been exploited for FL resource optimization. However, to the best of our knowledge, none of the DRL-based FL frameworks allows dynamic local epoch adjustment. Past studies [9,10,14,19–21] only exploited DRL to select clients that fulfil the resource constraints without introducing a tuning mechanism to adjust the local training epoch for clients with limited computing power.

To this end, we present federated double deep reinforcement learning (FedDdrl). FedDdrl exploits the double DRL (DDRL) framework, which uses two DRL agents to find the optimal client selection and local epoch adjustment policies. Our objective was to maximize the global model's accuracy while minimizing the FL system's training latency

and communication cost. We first formulated the FL protocol as a Markov Decision Process (MDP). We then adopted two DRLs based on Value Decomposition Networks (VDNs) as the policy networks. To speed up the convergence speed, we adopted the recently proposed balanced-MixUp [22] augmentation technique to mitigate weight divergence. Simulation results showed that our FedDdrl algorithm improved model accuracy with lower training latency and communication cost. Note that FL facilitates edge computing, which is one of the goals of the ASEAN IVO project titled “Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network”.

We summarize our contributions as follows.

1. We modeled the FL system as an MDP. Then, we proposed to use a DDRL framework for adaptive early client termination and local epoch adjustment, to maximize the global model accuracy while minimizing the training latency and communication costs.
2. We demonstrated our proposed algorithm in a non-IID setting on MNIST, CIFAR-10 and CrisisIBD datasets. We showed that our solution could outperform existing methods in terms of global model accuracy with shorter training latency and lower communication costs.
3. We explored the influence of balanced-MixUp in the FL system. In most settings, balanced-MixUp could mitigate weight divergence and improve convergence speed.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 discusses the system model and problem formulation. Section 4 presents the proposed solution. Section 5 shows the experimental setup, followed by the simulation results and discussion. Section 6 concludes the paper and outlines future research directions.

2. Related Work

This section reviews existing works on FL and DRL-based FL to provide insights into the current trend in FL. Then, we discuss the limitation of each algorithm. Lastly, we also elaborate on the weight divergence problem in FL, which is a common problem faced by all FL algorithms.

2.1. Federated Learning and Deep Reinforcement Learning

Some of the commonly used FL algorithms include FedAvg [4], FedProx [23] and FedNova [17]. FedAvg was the first practical implementation of FL. In each communication round, the server sends the global model to N randomly selected clients. Each client trains the model using its local dataset. Then, each client uploads its locally trained model to the server, where the server averages the received local models' weights as the new global model. It has since become the de facto approach for FL and is widely used in various applications [3,5–7]. FedProx presents a reparameterization of FedAvg by introducing an additional L_2 regularization term in the local objective function. The regularization term limits the distance between the local and global models, preventing local updates from diverging from global optima. A hyperparameter μ controls the weight of the regularization term. Overall, the modification can be easily performed on the existing FedAvg algorithm while improving model accuracy on non-IID datasets. However, it introduces additional computing overhead, leading to longer training latency. On the other hand, FedNova improves FedAvg in the aggregation stage. It allows each $n \in N$ client to conduct a different number of local steps. This allows clients with higher computing resources to conduct more training while waiting for others to complete training. To ensure that the global updates are not biased, each local update is normalized and scaled according to the number of local steps conducted before they are averaged into the new global model. FedNova introduces negligible computation overhead compared to FedAvg while handling computing resources heterogeneity in FL systems. However, all the above algorithms are limited to handling statistical heterogeneity (non-IID dataset) and computing resources heterogeneity. Other heuristic algorithms have been proposed to optimize client selection in FL systems with heterogeneous network and/or energy resources [24,25]. However,

heuristic solutions could only deliver sub-optimal performance since they often rely on qualitative analysis without exploring the optimal performance [14].

DRL has been widely applied to solve optimization problems involving complex sequential decision-making, such as playing Atari games [26], multiplayer games [27] and chess [28]. Since an FL procedure can be modelled as an MDP, it can also be optimized using DRL. FAVOR is one of the first research works to optimize FL using DRL [21]. They observed an implicit connection between the distribution of a local client dataset and the model weights trained on those data. Using the model weights collected from each participating client, a DRL agent can learn to select suitable clients for the next round of training. After proving DRL success in FL optimization, multiple studies [9,10,14,19,20,29] have exploited DRL in FL resource allocation problems. For instance, [9,10,14,19] used DRL to jointly optimize computing and network resources in an FL framework while retaining the global model's accuracy. These studies employed a DRL-based client selection policy or early client termination policy. Such a policy is responsible for selecting the best subset of clients for each round of training by optimizing the tradeoff between model accuracy and resource allocation. On the other hand, [29] optimized only the network resources by quantizing the model weights before uploading them to the FL server. However, none of the DRL-based FL frameworks described above allow dynamic local epoch adjustment. These frameworks fixed the same local epochs for all clients, disregarding their computing cost and training latency. With dynamic local epoch adjustment, clients with higher computing resources can conduct more training epochs. On the contrary, clients with limited computing power can train with fewer epochs.

Table 1 summarizes the key features of the aforementioned FL and DRL-based FL algorithms. In short, we noticed an ongoing trend of utilizing DRL to optimize the computing and network resources in the FL framework. Most of them relied only on client selection or early client termination techniques. As a result, such a method often rejects clients with limited computing power to prevent these devices from dragging the overall FL training latency. Even when such devices are selected, those with stronger computing power will finish training earlier and remain idle while waiting for the slower ones to complete training. However, these devices may contain crucial training data that is essential for FL convergence. Ideally, these devices should participate in FL training but with a lower local training epoch and vice versa. To the best of our knowledge, no DRL-based FL algorithms adopt DRL for automated local epoch adjustment. On the other hand, existing FL algorithms such as FedNova rely on manual adjustment to set devices with stronger computing power with a higher local epoch. Hence, an exciting potential exists for incorporating DRL-based dynamic local epoch adjustments for automated calibration.

Table 1. Features of existing FL and DRL-based FL algorithms.

Method	Resource Optimization	Client Selection	Local Epoch Adjustment
FedAvg [4]	-	Random	Fixed
FedProx [23]	-	Random	Fixed
FedNova [17]	Computing	Random	Flexible
FAVOR [21]	Computing	DRL Agent	Fixed
TP-DDPG [9]	Computing + Communication	DRL Agent	Fixed
Research work [10]	Computing + Communication	DRL Agent	Fixed
FedMarl [14]	Computing + Communication	DRL Agent	Fixed
Research work [19]	Computing + Communication	DRL Agent	Fixed
Research work [20]	Computing + Communication	Random	Fixed
Research work [29]	Communication	Random	Fixed
Proposed FedDdrl	Computing + Communication	DRL Agent	DRL Agent

2.2. Weight Divergence in Federated Learning

Weight divergence is the difference between FL model weights w_i^{FL} and CL model weights w_i^{CL} . An ideal level of weight divergence in FL could exploit the rich decentralized data, resulting in a better performance. For instance, FL outperforms its CL counterpart in various applications, including drug discovery [30], disaster classification [7] and autonomous driving object detection [31]. However, in extreme non-IID cases where the local client data distribution p^k is far from the global data distribution p , the highly diverged local weight updates could lead to bad aggregated solutions which are far from the global optimum solution. This is especially true in IoT networks, where each IoT device has a unique data distribution and can be considered non-IID [1]. According to [16], the main source of weight divergence is the earth mover's distance (EMD) between p^k and p , denoted as $\sum_{i=1}^{n_c} \|p^k(y=i) - p(y=i)\|$, where n_c denotes the total number of classes. In general, weight divergence is inevitable since p^k and p are almost guaranteed to be different in a real-life setting. Figure 1 shows an example of weight divergence between FedAvg and CL models.

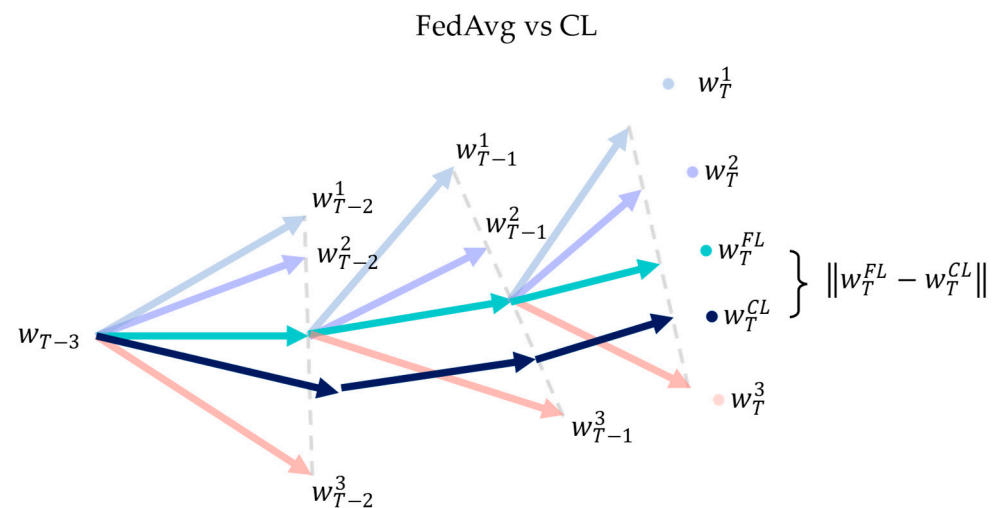


Figure 1. Weight divergence between w_i^{FL} and w_i^{CL} is inevitable even if both models have the same initialization weights.

Generally, only a fraction of the total clients is selected for training per communication round, t , to reduce the total communication cost in FL. Let C denote the total number of participating clients per round. When C is low, it is difficult to ensure the sampled data resemble the global data distribution. This also leads to high EMD between p^k and p , which again contributes to the divergence of w_i^{FL} from w_i^{CL} . Let A_t be the accuracy of the global model at communication round $t \in T$. Figure 2 shows the accuracy curve of FedAvg models trained on a non-IID CIFAR-10 dataset using $C = 5$ and $C = 10$. First, the average accuracy of the global model after $t = 15$ communication round was 62.73% and 72.79% for $C = 5$ and $C = 10$, respectively. Additionally, the fluctuation and standard deviation of the accuracy curve were higher when $C = 5$ as compared to $C = 10$. It is shown that FedAvg (or FL in general) had inferior performance when the number of participating clients per round is low.

Recent studies have contributed various solutions to mitigate weight divergence. For instance, the FedProx [23] mentioned earlier adds a regularization term to the local subproblem to prevent the local updates from diverging away from the global FL model. This method, in turn, hopes to ensure the aggregated global FL model weights w_i^{FL} are close to w_i^{CL} . Albeit effective, FedProx requires higher computing costs and a longer training time [32]. On the other hand, [16] proposed partial global sharing of local data to reduce EMD between client data distribution and the global populations. However, this induces high communication costs for data sharing and raises privacy concerns.

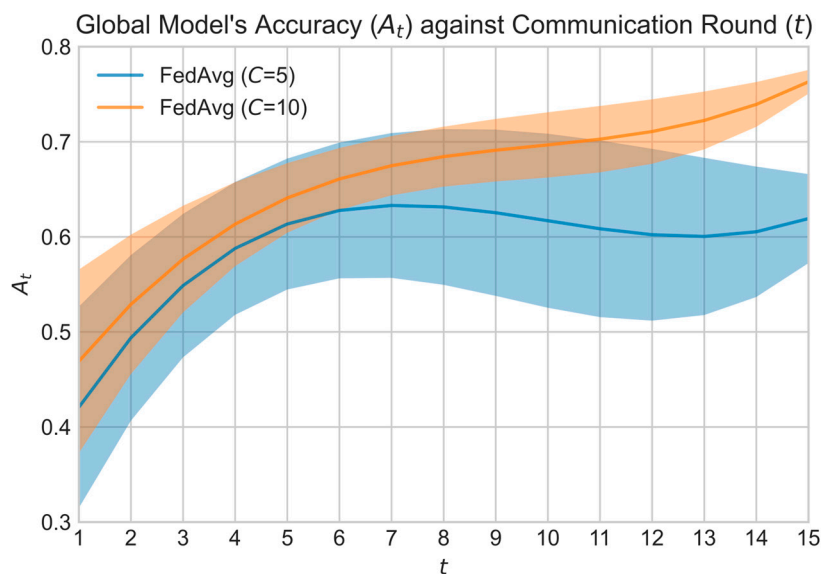


Figure 2. The global model’s accuracy curve for $C = 5$ and $C = 10$.

Meanwhile, methods that employ adaptive client selection or early client termination (i.e., FedMarl) aim to tackle weight divergence via careful client selection. For each communication round, FedMarl will only select a subset of the C clients for training. Ideally, only the selected clients are useful for training, while the rest are not. Effectively, this means that C is not constant for each communication round t . However, a lower C may lead to less steady convergence based on Figure 2. Thus, FedMarl is expected to handle the careful dropping of clients considering the EMD between global and local populations while taking care of other optimizing objectives, such as the training latency and communication cost of each client. Dropping the wrong clients may lead to large weight divergence, as shown in Figure 3.

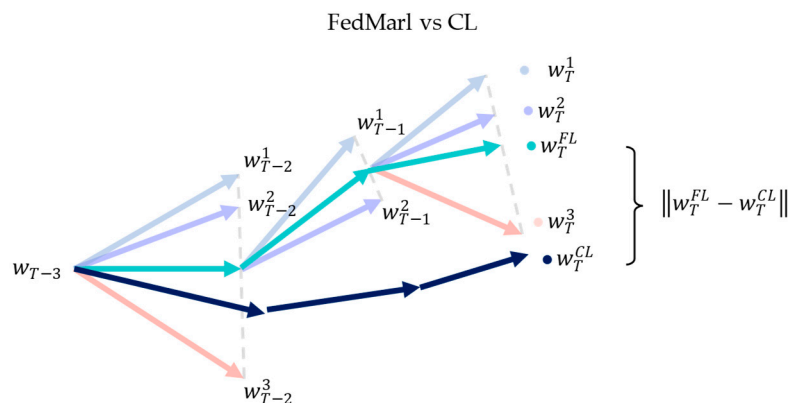


Figure 3. Ineffective client dropping by FedMarl will lead to large weight divergence. Client 3 (denoted by peach) is dropped from training at communication round $T - 2$. This causes the aggregated FL weights w_t^{FL} (denoted by green) to converge toward Client 1 and 2 while diverging away from the w_t^{CL} .

3. System Model and Problem Formulation

In this section, we present the system model for our FL system and discuss the problem formulation. Our commonly used symbols are listed in Table 2 for ease of reference.

Table 2. List of key variables defined in the system model.

Notation	Definition
t	Index of communication round
K	The total number of client devices (IoT devices)
N	The total number of client devices selected at each communication round
n	Index of selected IoT devices at communication round t
$H_{t,n}^b$	Model broadcasting latency from server to client n
$H_{t,n}^p$	Probing training latency for client n
$H_{t,n}^m$	Metadata uploading latency from client n to server
$H_{t,n}^u$	Model uploading latency from client n to server
H_t	Complete training latency for communication round t
B_n^t	Communication cost of client n
B_t	Total communication cost for communication round t
A_t	Accuracy of the global model at communication round t
ΔA_t	Global model's accuracy improvement
ϕ_t	Client selection matrix at communication round t
E_t	Local epoch count matrix at communication round t

3.1. System Model

We considered an FL system with K number of client devices. At communication round $t \in T$, N number of clients were randomly selected from the K number of clients. Each communication round consisted of four phases, which are: (1) model broadcasting, (2) probing training, (3) client dropping and (4) completion of training.

1. Model broadcasting: If $t = 1$, the FL server will initialize a global model, whereas at $t \geq 2$, the FL server will collect the client models trained at round $t - 1$ and aggregate them into a new global model. Then, the FL server will broadcast the global model to N randomly selected clients.
2. Probing training: Each selected client $n \in N$ will perform one epoch of local training called probing training. The purpose of probing training is to acquire the metadata of each client. The metadata consist of the client's states, which will be fed to the DRL agents for adaptive early client termination and local epoch adjustment. The details of the client states will be defined later together with the specification of the DRL agents. After probing training, each client will upload its metadata to the server and proceed to the next phase.
3. Early client termination: Based on the collected client states, the DRL agents at the FL server will drop non-essential clients to reduce total latency H_t and total communication cost B_t for round t . The decision made by DRL agents will be sent to each client.
4. Completion of training: Only the remaining C clients that are not dropped by the DRL agent will resume training. Each client n will complete the remaining local training until E_n^t epochs are reached. Each locally trained model will be uploaded to the FL server for model aggregation.

Let $H_{t,n}^p$ denote the probing training latency for client $n \in N$ at round $t \in T$. Let $H_{t,n}^c$ be the complete local training latency for client n at round t , while $H_{t,n}^u$ denotes the time taken for client n to upload its local model to the FL server. Let $\phi_n^t \in \{0, 1\}$ denote if client n is selected by the DRL agent to complete full local training. The total processing latency H_t and the total communication cost B_t at communication round t can be expressed by Equations (1) and (2):

$$H_t = \max_{1 \leq n \leq N} (H_{t,n}^c + H_{t,n}^u) a_n^t \quad (1)$$

$$B_t = \sum_{n=1}^N B_n^t \phi_n^t \quad (2)$$

Figure 4 depicts the system model for the proposed FL protocol. Note that the model broadcasting latency $H_{t,n}^b$ was not included into H_t since it is not part of the FL optimization problem. Additionally, the time latency to upload client metadata to the server, $H_{t,n}^m$, was ignored. This is because the metadata file size was only 278 bytes, while even a lightweight MobileNetV2 file is 24.5 megabytes. Thus, $H_{t,n}^m$ is negligible since $H_{t,n}^m \ll H_{t,n}^u$.

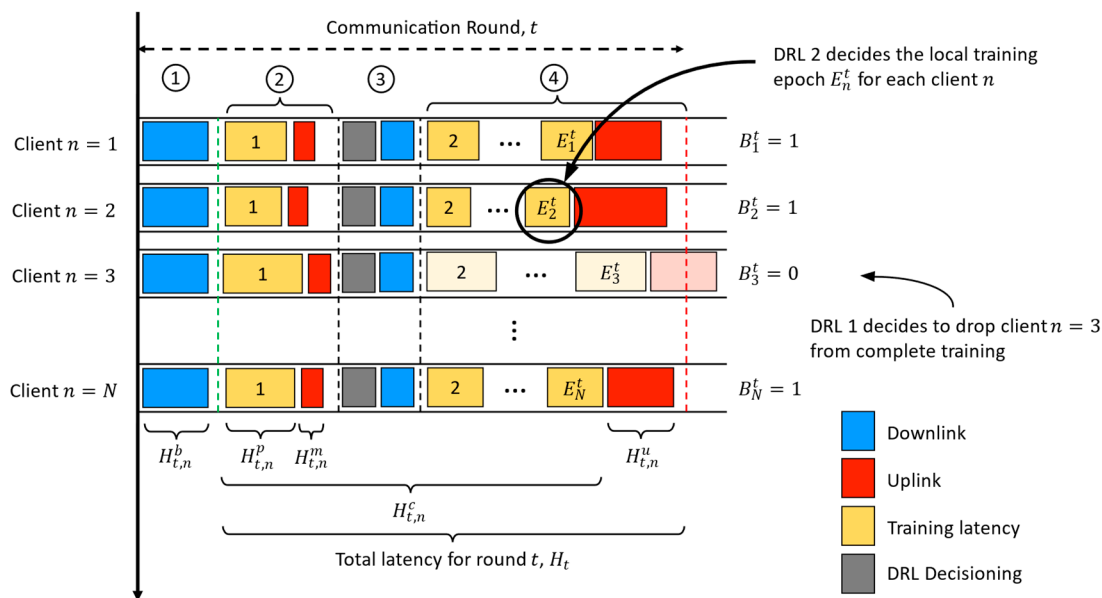


Figure 4. The proposed FL protocol’s system model consists of four phases in each communication round: (1) model broadcasting, (2) probing training, (3) early client termination and (4) completion of training.

3.2. Problem Formulation

Our objective was to maximize the cumulative A_t improvement while minimizing the H_t and B_t . Let $\phi_t = [\phi_n^t]$ and $E_t = [E_n^t]$ be a $T \times N$ matrix for client termination and local epoch adjustment decided by the DRL agents, respectively. We formulated the problem as a weighted sum optimization problem, as formulated below:

$$\max_{\phi_t, E_t} \mathbb{E} \left[\sum_{t=1}^T w_1 [U(A_t) - U(A_{t-1})] - (w_2 B_t + w_3 H_t) \right] \tag{3}$$

where w_1 , w_2 and w_3 are the weights to control the importance of each objective. To ensure A_t can improve even if it is small near the end of the FL process, a utility function denoted as $U(\cdot)$ was used to reshape the A_t of the global model. In FedMarl, $U(A_t)$ is defined in Equation (4):

$$U(A_t) = \frac{20}{1 + e^{0.35(1-A_t)}} - 10 \tag{4}$$

One problem with the original $U(A_t)$ is that it only tells us the transformed value of A_t . The entire $w_1 [U(A_t) - U(A_{t-1})]$ can be reparametrized into a single $w_1 U(\Delta A_t)$ expression, which could directly tell us the gain/penalty for ΔA_t . First, the $U(A_t)$ equation is simplified in the given range $0 \leq A_t \leq 1$ since A_t is bounded between 0 and 100%. In this range, $U(A_t)$ can be approximated as a straight line, as shown in Figure 5.

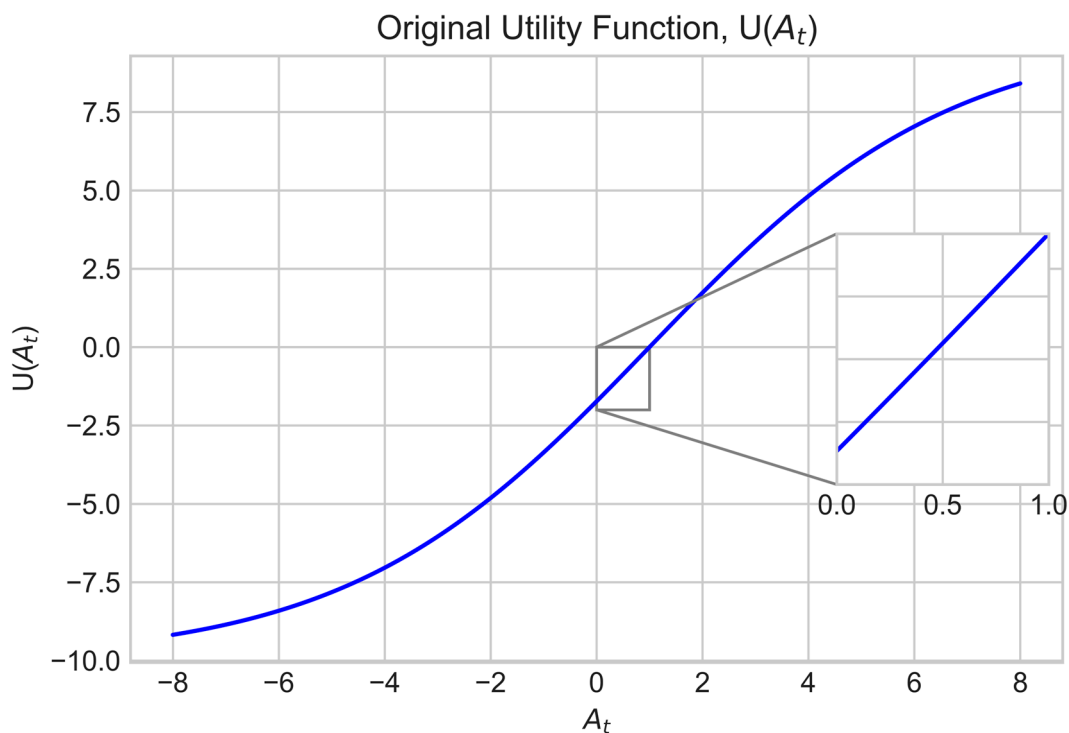


Figure 5. The original utility function $U(A_t)$ can be approximated as a straight line when A_t falls in the range $[0, 1]$.

To approximate $U(A_t)$ as a straight line in the given range, the gradient and y-intercept of the graph are required. The gradient is denoted as $U'(A_t)$, which is the first derivative of the $U(A_t)$ function. $U'(A_t)$ can be written as in Equation (5):

$$U'(A_t) = \frac{7e^{0.35(1-A_t)}}{(1 + e^{0.35(1-A_t)})^2} \tag{5}$$

The mean of the gradient, $\overline{U'(A_t)}$, within the range can be formulated as in Equation (6):

$$\begin{aligned} \overline{U'(A_t)} &= \frac{1}{1-0} \int_0^1 U'(A_t) dt \\ &= \int_0^1 \frac{7e^{0.35(1-A_t)}}{(1+e^{0.35(1-A_t)})^2} dt \\ &= 1.732 \end{aligned} \tag{6}$$

The y-intercept of $U(A_t)$, denoted as $U(A_t = 0)$, can be written as:

$$\begin{aligned} U(A_t = 0) &= \frac{20}{1+e^{0.35(1-0)}} - 10 \\ &= -1.732 \end{aligned} \tag{7}$$

Hence, $U(A_t)$ can be simplified into:

$$\begin{aligned} (A_t) &= \overline{U'(A_t)} A_t + U(A_t = 0) \\ &= 1.732 A_t - 1.732 \end{aligned} \tag{8}$$

Thus, $U(\Delta A_t)$ can be defined as:

$$\begin{aligned}
 U(\Delta A_t) & \\
 &\cong U(A_t) - U(A_{t-1}) \\
 &= (1.732 A_t - 1.732) - (1.732 A_{t-1} - 1.732) \\
 &= 1.732(A_t - A_{t-1}) \\
 &= 1.732 \Delta A_t
 \end{aligned} \tag{9}$$

$U(\Delta A_t)$ is more analyzable than $U(A_t) - U(A_{t-1})$ since the two expressions have been collapsed into one equation. At this end, we can define our optimization problem as:

$$\max_{\phi_t, E_t} \mathbb{E} \left[\sum_{t=1}^T w_1 U(\Delta A_t) - (w_2 B_t + w_3 H_t) \right] \tag{10a}$$

s.t.

$$\mathbb{E} \left(w_3 \sum_{t=1}^T H_t \right) > \Omega_1 \mathbb{E} \left(w_2 \sum_{t=1}^T B_t \right) \tag{10b}$$

$$w_1 U(\Delta A_t = 0.01) > \Omega_2 \mathbb{E}(w_2 B_t + w_3 H_t) \tag{10d}$$

$$\mathbb{E} \left(w_1 \sum_{t=1}^T U(\Delta A_t) \right) > \Omega_3 \left(w_2 \sum_{t=1}^T B_t + w_3 \sum_{t=1}^T H_t \right) \tag{10e}$$

where (10b–e) are the constraints for our optimization problem. Constraint (10b) is to make sure the sign of $U(\Delta A_t)$, B_t and H_t are not inverted. Meanwhile, constraint (10c) is to control the ratio of $\sum_{t=1}^T B_t$ to $\sum_{t=1}^T H_t$. Furthermore, constraint (10d) is to make sure $w_1 U(\Delta A_t)$ gain will not be outweighed ($w_2 B_t + w_3 H_t$) penalties when ΔA_t is as small as 0.01. Lastly, constraint (10e) makes sure $\sum_{t=1}^T w_1 U(\Delta A_t)$ is at least Ω_3 greater than the penalty terms. Note that (10c–e) are additional constraints that are not imposed on the original FedMarl optimization problem.

In FedMarl, the w_1 , w_2 and w_3 are treated as hyperparameters. FL engineers have to manually adjust the weightage of each objective until the desired outcome is achieved. However, the weightage w_1 , w_2 and w_3 does not directly translate to the weightage of each objective $\sum_{t=1}^T U(\Delta A_t)$, $\sum_{t=1}^T B_t$ and $\sum_{t=1}^T H_t$. For instance, the ratio of $w_2 B_t$ to $w_3 H_t$ does not directly equate to the ratio of $w_2 \sum_{t=1}^T B_t$ to $w_3 \sum_{t=1}^T H_t$. This is because the values of H_t and B_t are instantaneous and stochastic, which means that the ratio of $w_3 H_t$ to $w_2 B_t$ at two different t is most likely different. On the other hand, $\mathbb{E} \left(\sum_{t=1}^T H_t \right)$ and $\mathbb{E} \left(w_2 \sum_{t=1}^T B_t \right)$ are more consistent. Taking the ratio of these two components is more reliable.

We can find the best w_1 , w_2 and w_3 by setting the desired Ω_1 , Ω_2 and Ω_3 . We set $\Omega_1 = 0.2$, $\Omega_2 = 0.3$ and $\Omega_3 = 1.0$. We needed to run one iteration of FL using FedAvg to get the traces value of ΔA_t , B_t and H_t for $t \in T$ since these values are dependant on the target IoT environment setup. Based on the traces value, we could follow Algorithm 1 to acquire the suitable w_1 , w_2 and w_3 . In our experiment setup, we found the desired hyperparameters to be ($w_1 = 2.9$, $w_2 = 0.1$ and $w_3 = 0.2$).

Algorithm 1 Search for the best w_1, w_2 and w_3

```

1. Input: Set  $\Omega_1 = 0.2, \Omega_2 = 0.3, \Omega_3 = 1.0$ 
2. Output: The best  $w_1, w_2, w_3$ 
3: Run one complete iteration of FedAvg with  $T = 15$  communication rounds and record the
   traces value of  $\Delta A_t, B_t$  and  $H_t$  for  $t \in T$ .
4: Initialize an empty set  $cache = \{\}$  to store all  $((w_1, w_2, w_3), R)$  that satisfied constraints
    $(10c-e)R$  is the weighted-sum optimization goal  $w_1U(\Delta A_t) - (w_2B_t + w_3H_t)$ 
5: for  $w_1 = 0, 0.1, \dots, 3.0$  do
6:   for  $w_2 = 0, 0.1, \dots, 1.0$  do
7:     for  $w_3 = 0, 0.1, \dots, 1.0$  do
8:       Compute  $\mathbb{E}\left[w_1 \sum_{t=1}^T U(\Delta A_t)\right], \mathbb{E}\left[w_2 \sum_{t=1}^T B_t\right], \mathbb{E}\left[w_3 \sum_{t=1}^T H_t\right],$ 
        $\mathbb{E}(w_2B_t + w_3H_t)$  based on the recorded traces value, where
       we assume  $\mathbb{E}(x) \triangleq x$ 
9:       if (10c-e) are satisfied:
10:         Compute
11:          $R = \sum_{t=1}^T w_1U(\Delta A_t) - (w_2B_t + w_3H_t)$ 
12:         from the traces value
13:         Record  $((w_1, w_2, w_3), R)$  in  $cache$ 
14:       end if
15:     end for
16:   end for
17: end for

```

From $cache$, find out which combination of (w_1, w_2, w_3) results in the smallest R . This can be treated as finding the worse-case $\max \mathbb{E}[R]$.

4. Proposed Method

We propose FedDdrl, which exploits two DRL policy networks for FL optimization. Specifically, we adopted VDNs as the DRL policy networks for FedDdrl. We elaborate in detail on how we formulated the problem as MDP, including the design of state space, action space and reward of the algorithm. In addition, we also exploited the recently proposed balanced-MixUp to mitigate the impact of weight divergence and speed up the FL convergence speed.

4.1. Deep Reinforcement Learning for Federated Learning Optimization

The proposed optimization problem in Equation (10a–e) is a 0–1 Multidimensional Knapsack Problem (MKP). The items to be put in knapsacks are the client devices n with complete training latency $H_{t,n}^c$, model uploading latency $H_{t,n}^u$, communication cost B_n^t and data size D_n . The total capacity of the knapsack equals the total communication cost $B_t = \sum_n B_n^t a_n^t$, where a_n^t is the binary indicator of item (client) n . When a_n^t is set to 1, item (client) n is selected. Otherwise, a_n^t is set to 0. The total weight of the knapsack has a lower bound which has to fulfil the minimum requirement of accuracy constraint(10d). Our goal is to select a subset of clients C ($1 < C \leq N$) for complete training in each communication round to maximize the total accuracy gain $\sum_{t=1}^T \Delta A_t$ while minimizing the total latency $\sum_{t=1}^T H_t$ and total communication cost $\sum_{t=1}^T B_t$ of the entire FL training. Thus, the proposed optimization is NP-hard.

To solve problem (10), our FedDdrl algorithm adopted a double DRL framework for our optimization problem. Specifically, we formulated the DRL policy network for both tasks using a multi-agent reinforcement learning (MARL) approach. In particular, VDN has proven itself in the recent literature [33] to be a promising candidate for MARL problems. A VDN network consists of N agents, in which each agent $n \in N$ uses a deep neural network (DNN) parametrized with θ to implement the Q-function $Q_n^\theta(s, a) = \mathbb{E}[R_t | s = s_n^t, a = a_n^t]$. At each timestep t , each agent n observes its states s_n^t and selects the optimal action a_n^t with the maximum Q-value. Let $s_t = \{s_n^t\}$ and $a_t = \{a_n^t\}$ represent the states and actions

collected from all agents $n \in N$ at timestep t , respectively. The joint Q-function $Q_{tot}(\cdot)$ for the multi-agent VDN system can be represented by the elementwise summation of all the individual Q-functions, where $Q_{tot}(s_t, \mathbf{a}_t) = \sum_n Q_n^\theta(s_n^t, a_n^t)$. In FedDdrl, we set each agent in both VDN as a simple two-layer multi-layer perceptron (MLP), which is cheap to implement. All MLPs in each VDN share their weights to prevent the lazy agent problem [33].

As illustrated in Figure 6, the first VDN network takes the client states s_t (which will be detailed later) to obtain the optimal client termination matrix ϕ_t . The second VDN network takes the same client states s_t to obtain the optimal local training epoch per client E_t .

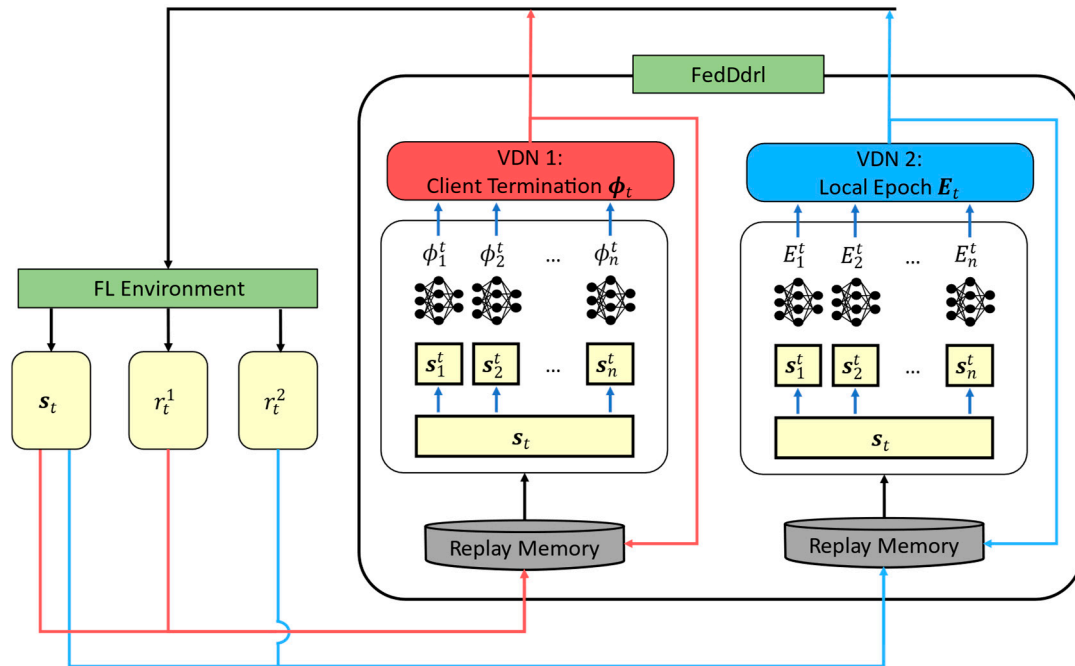


Figure 6. Structure of the proposed FedDdrl algorithm.

4.1.1. Early Client Termination

Inspired by the work of FedMarl [14], the first VDN network was employed to learn the optimal policy for early client termination matrix ϕ_t at round t . We reformulated the problem as an MDP with the following state, action and reward to train a VDN network with $N = 10$ agents.

1. **State s_t** State $s_t = \{s_n^t\}$ consisted of the client states for each VDN agent. Each agent n consisted of six components: (i) the probing loss L_n^t , (ii) probing training latencies $H_{t,n}^p$, (iii) model uploading latencies $H_{t,n}^u$, (iv) communication cost from client to server B_n^t , (v) local training dataset size $|D_n|$ and (vi) current communication round index t . The state vector for agent c can be written as Equation (11):

$$s_n^t = [L_n^t, H_{t,n}^p, H_{t,n}^u, B_n^t, |D_n|, t] \quad (11)$$

It is noteworthy that since each agent in the VDN only has access to its own local observation instead of the full observed environment, the policy has to incorporate past agent observations from history [33]. Thus, the historical values of probing latencies $H_{t,n}^p = [H_{t-\Delta T_p, n}^p, \dots, H_{t,n}^p]$ and model uploading latencies $H_{t,n}^u = [H_{t-\Delta T_u, n}^u, \dots, H_{t,n}^u]$ were included in the state vector to mitigate the limitation of local observation. Note that ΔT_p and ΔT_u are the sizes of the historical information of probing latencies and model uploading latencies, respectively.

2. **Action ϕ_t :** Action $\phi_t = \{\phi_n^t\}$ comprised the client termination decision for each VDN agent. The action space for client termination was $\phi_n^t = \{0, 1\}$, where 0 indicates the termination of the client and 1 indicates the client is selected for complete training.
3. **Reward r_t^1 :** A vanilla reward for VDN 1, denoted as r_t^1 , can be adopted from the FL optimization problem as described in Equation (12):

$$r_t^1 = w_1 U(\Delta A_t) - (w_2 B_t + w_3 H_t) \quad (12)$$

where the system is rewarded with accuracy improvement ΔA_t and penalties for B_t and H_t . However, Equation (12) has one obvious limitation. When $\Delta A_t \rightarrow 0$, the $\lim_{\Delta A_t \rightarrow 0} w_1 U(\Delta A_t) = 0$, regardless of the magnitude of w_1 . If $w_1 U(\Delta A_t) \rightarrow 0$, the reward $r_t^1 \approx -(\overline{w_2 B_t + w_3 H_t})$. This causes the optimization problem to diverge from improving accuracy with the constraint of B_t and H_t to merely the reduction of B_t and H_t . To show the severeness of this problem, we trained the VDN agents using the r_t^1 as defined by Equation (12). Let $\mathbb{E}[w_1 U(\Delta A_t)]$ and $\mathbb{E}[w_2 B_t + w_3 H_t]$ denote the expected values of accuracy improvement ΔA_t and penalties $(w_2 B_t + w_3 H_t)$, respectively. For MNIST dataset, the expected values of both components for the last $R = 5$ communication rounds can be computed in Equations (13) and (14):

$$\begin{aligned} & \mathbb{E}[w_1 U(\Delta A_t)]|_{t=\{T-R, T-R-1, \dots, T\}} \\ &= \frac{1}{5} \sum_{t=T-5}^T w_1 U(\Delta A_t) \\ &= 0.0273 \end{aligned} \quad (13)$$

$$\begin{aligned} & \mathbb{E}[w_2 B_t + w_3 H_t]|_{t=\{T-R, T-R-1, \dots, T\}} \\ &= \frac{1}{5} \sum_{t=T-5}^T (w_2 B_t + w_3 H_t) \\ &= 0.279 \end{aligned} \quad (14)$$

It is observed that $\mathbb{E}[w_1 U(\Delta A_t)] \ll \mathbb{E}[w_2 B_t + w_3 H_t]$ for the last five communication rounds. This is because as training approach the end, the accuracy improvement is often smaller compared to the earlier stage. Consequently, the VDN agents start to terminate more clients from complete training, giving way to the reduction of B_t and H_t . To make sure the agents are motivated to learn even when $\Delta A_t \rightarrow 0$, we can introduce a bias term b to r_t^1 . Let $b = \frac{3}{10} \mathbb{E}[w_2 B_t + w_3 H_t]$. Hence, the reward function r_t^1 can be reformulated as shown in Equation (15):

$$r_t^1 = \begin{cases} r_t + b, & \Delta A_t > 0 \\ r_t, & \Delta A_t \leq 0 \end{cases}, \quad r_t = w_1 U(\Delta A_t) - (w_2 B_t + w_3 H_t) \quad (15)$$

Note that we only added the bias term b to the reward r_t when $\Delta A_t > 0$ since it is intended to encourage accuracy improvement. We did not subtract the bias term b from the reward r_t when $\Delta A_t \leq 0$ since the penalty terms are sufficient to penalize the inferior actions.

4.1.2. Local Epoch Adjustments

The second DRL network was employed to learn the optimal policy for local epoch adjustments E_t at round t . A VDN algorithm with $N = 10$ agents can be formulated by defining the state, action and reward as follows:

1. **State s_t :** The second VDN shared the same state in Equation (11) since both VDNs required the same local observation for decision making.
2. **Action E_t :** Action $E_t = \{E_n^t\}$ comprises the local epoch counts for each VDN agent. The action space is $E_n^t = \{3, 5, 7\}$. This action aims to exploit client devices with stronger computation power for more training epochs and vice versa.

3. **Reward r_t^2** : We adopted Equation (15) as the starting point for the reward function for VDN 2. However, the communication cost B_t was not part of the optimizing objectives of VDN 2 since local epoch adjustment is only bounded by the H_t constraint. Hence, the reward function r_t^2 for this VDN networks can ignore the B_t penalty. As such, the r_t^2 can be defined in Equation (16):

$$r_t^2 = \begin{cases} r_t + b, & \Delta A_t > 0 \\ r_t, & \Delta A_t \leq 0 \end{cases}, \text{ where } r_t = w_1 U(\Delta A_t) - w_3 H_t \quad (16)$$

where we used the same bias term b from (15) for the simplicity's sake.

As the training converges, VDN 1 will deliver the optimal client selection, and VDN 2 will impart the optimal local epoch number for each client. The overall algorithm for solving the problem in Equation (10a–e) is summarized in Algorithm 2.

Algorithm 2 FedDdrl Algorithm

1: Input:	Initialize VDN 1 Q_{tot}^1 and its target network $Q_{tot}^{1'}$ for client selection ϕ_t policy
	Initialize VDN 2 Q_{tot}^2 and its target network $Q_{tot}^{2'}$ for local epoch adjustment E_t policy
2: Output:	Trained Q_{tot}^1 and Q_{tot}^2 networks
3: Set $\varepsilon = 1.0$	
4: for Episode $n_{ep} = 1, 2, \dots, N_{ep}$ do	
5: Reset the FL environment	
6: Initialize a global model w_0	
7: for communication round $t = 1, 2, \dots, T$ do	
8: Randomly select N clients from all K clients	
9: Broadcast the global model w_t to each selected client	
10: for each client $n \in N$ in parallel do	
11: $w_t^n \leftarrow w_t$; Copy the global model as each client model	
12: Update the client model w_t^n using the local training dataset D_n	
13: Upload client states s_t^n to the FL server	
14: end for	
15: Each agent n in VDN 1 selects the optimal action $\phi_{t,n}^* = \text{argmax } Q_n^1(s_t^n, \phi_n^t)$ with a $(1 - \varepsilon) \times 100\%$ probability, else randomly output actions	
16: Each agent n in VDN 2 selects the optimal action $E_{t,n}^* = \text{argmax } Q_n^2(s_t^n, E_n^t)$ with a $(1 - \varepsilon) \times 100\%$ probability, else randomly output actions	
17: Send action $\phi_{t,n}^*$ and $E_{t,n}^*$ to each client $n \in N$	
18: for each client $n \in N$ in parallel do	
19: if $\phi_{t,n}^* = 1$:	Continue updating w_t^n using D_n until $E_{t,n}^*$ is reached
20:	Return updated w_t^n
21: end if	
22: end for	
23: end for	
24: Aggregate global model $w_{t+1} \leftarrow \sum_{i=1}^N \frac{ D_i }{\sum_{i=1}^N D_i } w_t^i$ where $i \in \{n \phi_{t,n}^* = 1\}$	
25: Reward r_t^1 and r_t^2 are given to Q_{tot}^1 and Q_{tot}^2 based on $\Delta A_t, B_t, H_t$	
26: $s_t = \{s_n^t\}, \phi_t = \{\phi_n^t\}, E_t = \{E_n^t\}$	
27: Store transitions 1 $[s_t, \phi_t, r_t^1]$ for Q_{tot}^1 into memory buffer 1	
28: Store transitions 2 $[s_t, E_t, r_t^2]$ for Q_{tot}^2 into memory buffer 2	
29: Sample mini-batches with size n_b from memory buffer to train Q_{tot}^1, Q_{tot}^2 and $Q_{tot}^{1'}, Q_{tot}^{2'}$	
30: Decay ε gradually from 1.0 to 0.1	
31: end for	
32: end for	

4.2. Balanced-MixUp to Mitigate Weight Divergence

In this study, we focused on the non-IID label shift, where the client dataset is heavily skewed to one of the label classes. The huge EMD between the client data distribution p^k and the global distribution p will contribute to weight divergence, deteriorating the training efficiency of FL. Thus, the highly imbalanced client dataset has to be handled wisely. MixUp [34] is a simple yet effective data augmentation technique that could shed some light on this problem.

MixUp extends the training data distribution by linearly interpolating between existing data points, filling the underpopulated areas in the data space. It generates synthetic training data (\hat{x}, \hat{y}) by simply taking the weighted combination of two random data pairs, (x_i, y_i) and (x_j, y_j) , as shown in Equations (17) and (18):

$$\hat{x} = \lambda x_i + (1 - \lambda)x_j \quad (17)$$

$$\hat{y} = \lambda y_i + (1 - \lambda)y_j \quad (18)$$

where $\lambda \sim \text{Beta}(\alpha, \alpha)$, with $\alpha > 0$. Despite its simplicity, MixUp has been proven to improve model calibration and better generalization [35]. Thus, its application has expanded from image and speech classification tasks [34] to other domains, including image segmentation [36] and natural language processing [37,38]. However, a vanilla MixUp works poorly in highly imbalanced datasets [39]. In highly imbalanced datasets, MixUp would end up sampling the data pairs (x_i, y_i) and (x_j, y_j) from the same class for most of the time since the sampling is done randomly.

Some recent studies focused on solving the data imbalance problem for MixUp. In Remix by [39], x_i and x_j are mixed in the same fashion as MixUp, but y_i and y_j are mixed such that the minority class is assigned a higher weight. This method pushes the decision boundaries away from the minority class, balancing the generalization error between the majority and minority classes. Balanced-MixUp is another variation of MixUp, where MixUp is combined with a data-resampling technique to achieve a balanced class distribution [22]. Specifically, balanced-MixUp combines instance-based sampling and class-based sampling for the majority and minority classes, respectively. This ensures that each data pair (x_i, y_i) and (x_j, y_j) always consists of instances from both the majority and minority classes.

We adopted balanced-MixUp as the augmentation into the formulation of our solution to address the class imbalanced problem. To the best of our knowledge, we are the first to integrate balanced-MixUp into FL for weight divergence mitigation. Let (x_M, y_M) and (x_m, y_m) denote the instance pair sampled from the majority and minority classes, respectively. Balanced-MixUp can be expressed as shown in Equations (19) and (20):

$$\hat{x} = \lambda x_M + (1 - \lambda)x_m \quad (19)$$

$$\hat{y} = \lambda y_M + (1 - \lambda)y_m \quad (20)$$

Balanced-MixUp guarantees that each data pair mixing consists of instances from both the majority and minority class. Unlike the original balanced-MixUp where $\lambda \sim \text{Beta}(1, \alpha)$, we adopted $\lambda \sim \text{Beta}(\alpha, \alpha)$ and found it to work better in our study. The best α value may be different depending on the datasets, which will be detailed in the results section.

5. Simulation Results

This study adopted TensorFlow as the deep learning platform. We adopted three datasets for FL benchmarking: MNIST, CIFAR-10 and CrisisIBD [40]. First, MNIST is a relatively simple task under most non-IID settings. It is mainly used to prove that a novel FL algorithm is at least working. In contrast, CIFAR-10 is a challenging dataset in non-IID settings, which is strongly encouraged to be included in FL benchmarking experiments [32]. On the other hand, the CrisisIBD dataset is the benchmark dataset for various real-world disaster-related image classification tasks. In this study, we adopted the disaster classification dataset from the dataset (hereinafter referred to as CrisisIBD) as one of our benchmark datasets. We adopted balanced-MixUp to augment all three datasets. We found the best α value used by balanced-MixUp was 0.05, 0.4 and 0.2 for MNIST, CIFAR-10 and CrisisIBD, respectively. We used all three datasets to train a lightweight

MobileNetV2 [41], which aligned with our goal of developing the FL framework for low-powered IoT devices. All clients adopted $B_n^t = 1 \forall n, t$, similar to the setting in [14].

In this study, we focused on the non-IID label shift as demonstrated in [14,21]. Similarly, we divided each dataset into K clients. For each client, a fraction $\sigma = 0.8$ of the local training dataset was sampled from one random label (which is the majority label), while the rest of the training data were sampled uniformly from the remaining labels (which are the minority labels). We compared our proposed method with FedAvg, FedProx and FedMarl. The first two algorithms were shown to be robust baselines in non-IID label shift [32] and are prebuilt in many existing FL frameworks, including Tensorflow Federated [42] and Intel OpenFL [8]. On the other hand, FedMarl is one of the state-of-the-art FL algorithms [14]. Our FedDdr1 aims to outperform all three of the algorithms. The hyperparameters are listed in Table 3.

Table 3. List of hyperparameters.

Parameters	Values
Number of agents in each VDN network, N	10
Total number of clients, K	100
Local training dataset distribution, σ	0.8
Learning rate for VDN network	1×10^{-3}
Target network update interval	5
Number of episodes, N_{ep}	40
Number of clients selected for training in each round, C	10
Default number of local epochs (before adjustment by FedDdr1), E_t	5
Number of communication rounds, T	15
Batch size to update VDN agents, N_b	32
Initial ϵ -greedy exploration value	1
Final ϵ -greedy exploration value	0.1
Replay memory size	300
VDN 1 agent (MLP) size	$10 \times 256 \times 256 \times 2$
VDN 2 agent (MLP) size	$10 \times 256 \times 256 \times 3$

Due to limited resources, we only had two hardware devices: (i) an Intel NUC with an Intel core i7-10710U processor with 4.70 GHz and (ii) a workstation equipped with an Intel core i7-10875H processor with 2.30 GHz and NVIDIA RTX 2070 SUPER. In total, this yielded two TensorFlow CPU operators and one TensorFlow GPU operator. However, this was far from enough to simulate a heterogeneous FL environment with $K = 100$ clients if each operator only represents one client. Thus, we carefully devised our experimental setup, as shown in Figure 7.

To simulate an FL environment with $K = 100$ clients (IoT devices) and $C = 10$ selected clients (before early termination), we set up the experiment as shown below:

1. We created $K = 100$ client configurations, each consisting of the (i) client's computing latency per data, (ii) model upload latency and (iii) local dataset identity (ID) number. To closely simulate the heterogeneity of resources in an IoT network as in [14], the computing latency per data in each client configuration can be any of $\{0.25, 0.50, 0.75\}$ seconds, while the model upload latency can be any of $\{1.00, 1.25, 1.75, 2.00\}$ seconds.
2. CPU 1, CPU 2 and GPU simulated three, three and four clients, respectively. The simulated clients represent the $C = 10$ randomly selected clients from the total $K = 100$ clients in each communication round t .
3. In each communication round t , 10 client configurations were randomly sampled out from the configuration pools. The 10 simulated clients (in CPU 1, CPU 2 and GPU) were configured according to the selected client configuration. This entire process (3) is equivalent to the FL process of randomly selected 10 clients with unique local datasets and resources.
4. After step (3), each simulated client proceeded with its training. If the FL algorithm was FedAvg or FedProx, all 10 simulated clients underwent complete training of

$E_t = 5$ local epochs. On the contrary, if the FL algorithm was FedMarl or FedDdrl, only the simulated clients that were not terminated by the FedMarl/FedDdrl completed their local training based on $E_{t,n}^* = \operatorname{argmax} Q_n^2(s_n^t, E_n^t)$ by VDN 2.

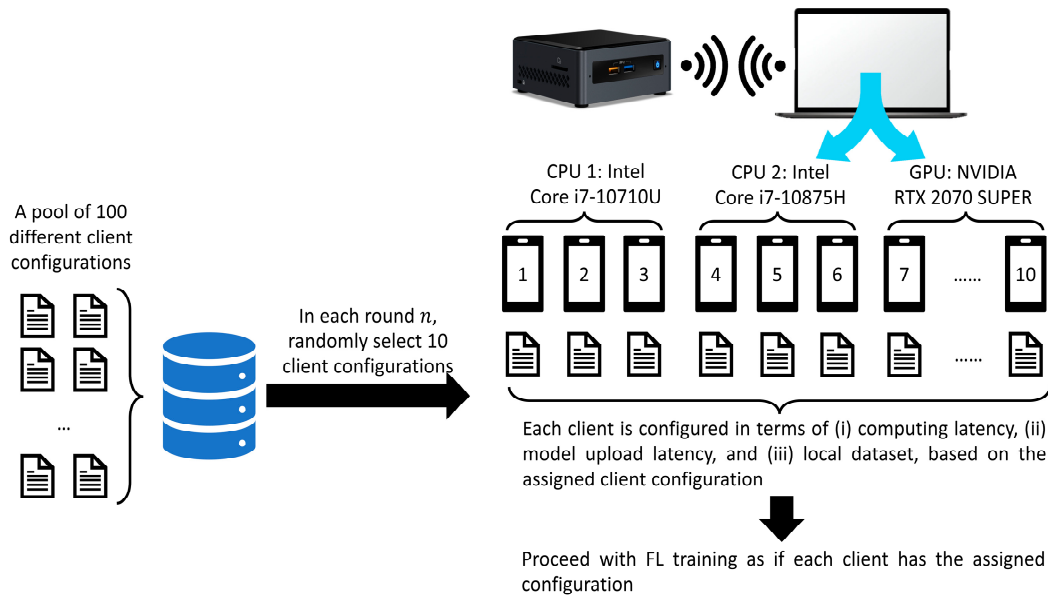


Figure 7. Experiment setup.

5.1. Results and Ablation Study

We compared the performance of FedDdrl with other baselines in all three objectives of the optimization problem, which are the (i) model accuracy, (ii) training latency and (iii) communication efficiency.

5.1.1. Model Accuracy

Table 4 shows the model accuracy trained using each FL setting after $T = 15$ communication rounds. We also conducted an ablation study showing how FedDdrl improves beyond FedMarl.

Table 4. Model accuracy for each FL setting. Bolded indicates the best score, while underlined indicates the second-best score.

Method	MNIST (K=100)	CIFAR-10 (K=100)	CrisisIBD (K=98)
FedAvg	94.6% ± 2.1%	72.8% ± 3.9%	43.2% ± 5.5%
FedAvg with Balanced-MixUp	93.2% ± 2.0%	76.5% ± 1.7%	60.2% ± 1.5%
FedProx ($\mu = 0.01$)	95.6% ± 0.5%	74.5% ± 0.2%	48.1% ± 2.9%
FedProx ($\mu = 0.01$) with Balanced-MixUp	<u>95.4% ± 0.7%</u>	<u>77.8% ± 0.5%</u>	60.7% ± 2.0%
A: FedMarl ($w_1=1.0$, $w_2=0.1$, $w_3=0.2$)	91.5% ± 1.1%	65.5% ± 2.3%	42.4% ± 3.6%
B: A + Optimized ($w_1=2.9$, $w_2=0.1$, $w_3=0.2$)	93.2% ± 1.4%	71.7% ± 2.9%	44.4% ± 3.9%
C: B + Balanced-MixUp	93.3% ± 1.2%	75.0% ± 2.6%	<u>63.3% ± 2.0%</u>
D: C + Local Epoch Adjustment (FedDdrl)	94.9% ± 1.1%	78.2% ± 2.4%	64.2% ± 1.4%

Setting A in Table 4 was our implementation of FedMarl with the original hyperparameters ($w_1 = 1.0$, $w_2 = 0.1$, $w_3 = 0.2$). In Setting B, we proved that FedMarl performance

could be improved using our hyperparameters ($w_1 = 2.9$, $w_2 = 0.1$, $w_3 = 0.2$). However, we found that its accuracy was still far behind both FedAvg and FedProx. This is because MobileNetV2 is a lightweight model which is easy to overfit [41]. In FedAvg and FedProx, the total number of clients selected for training is always $C = 10$ in each communication round t . During aggregation, there is a sufficient amount of client models overfitted for different classes, which, when aggregated, can generate a regularization effect, thus mitigating the weight divergence caused by overfitting. This is not the case for FedMarl, which does not have a fixed C for each round. We argue that since the original FedMarl was not tested on MobileNetV2, it was able to perform better than FedAvg and FedProx. Applying balanced-MixUp could significantly mitigate this problem, as shown in Setting C. The reasoning on how balanced-MixUp helps weight divergence mitigation is detailed in Section 5.5.

Setting D was our FedDdrl, where we added another VDN for local epoch adjustment. FedDdrl allows client devices to train for more epochs when required and vice versa. This allows FedDdrl to converge faster than FedAvg and FedProx for most cases, even when it is not utilizing all clients at each communication round. FedDdrl outperformed other FL algorithms in both the challenging CIFAR-10 and CrisisIBD datasets, and it was the second-best for MNIST. We suspect FedDdrl is slightly overengineered for an easy task like MNIST. Nevertheless, it was still very robust considering that real-world data is often not as simple as MNIST and is instead more challenging like the CIFAR-10 and CrisisIBD datasets.

5.1.2. Training Latency

Figure 8 shows the normalized training latency for each FL algorithm (with balanced-MixUp) on all three datasets. Our FedDdrl outperformed all three other algorithms in all datasets. This is promising since FedDdrl allows dynamic local adjustment. The FedDdrl will sometimes increase the local epoch from five to seven. However, the extra training latency is balanced when FedDdrl decreases the local epoch from five to three, especially in the early communication round when the MobileNetV2 is still learning lower-level features. FedMarl followed closely behind FedDdrl. This is mainly because both FedDdrl and FedMarl can terminate clients with longer probing latencies. On the other hand, FedAvg had a moderate performance in terms of training latency. It was not as fast as FedMarl and FedDdrl, but it was still significantly faster than FedProx. As expected, FedProx had the longest training latency compared to other FL algorithms, which is aligned with the observation by [32]. This is mainly due to the extra computing cost required to compute the L2 distance between the client and the global model. Hence, applying FedProx in low-powered devices (i.e., IoT devices) with limited computation power is not feasible.

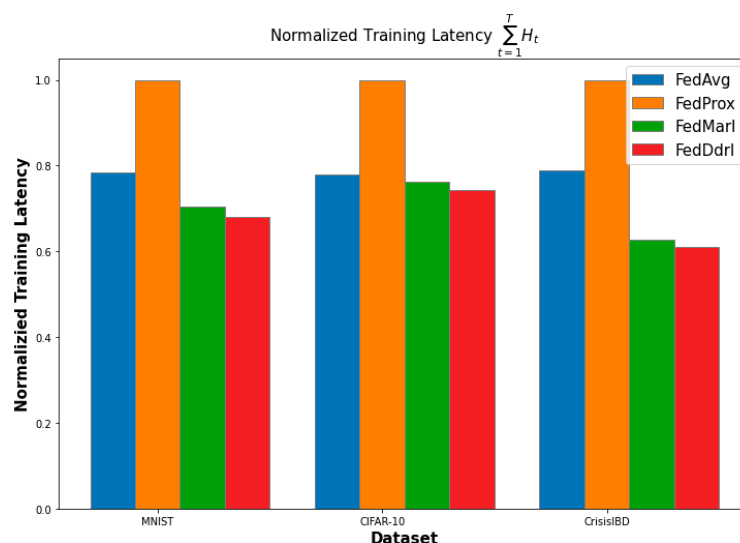


Figure 8. Normalized training latency of each FL algorithm.

5.1.3. Communication Efficiency

Figure 9 shows the normalized communication costs of all FL algorithms in the three datasets. FedDdrl and FedMarl were significantly more efficient than FedAvg and FedProx in total communication costs. There was no clear winner between FedDdrl and FedMarl regarding communication efficiency. However, FedDdrl outperformed FedMarl in CIFAR-10 and CrisisIBD datasets, which are significantly harder tasks compared to MNIST. Thus, we argue that FedDdrl is the best algorithm. On the other hand, FedAvg and FedProx have a fixed number of clients selected in each round. Since we assume $B_n^t = 1 \forall n, t$, both algorithms have the same total communication costs.

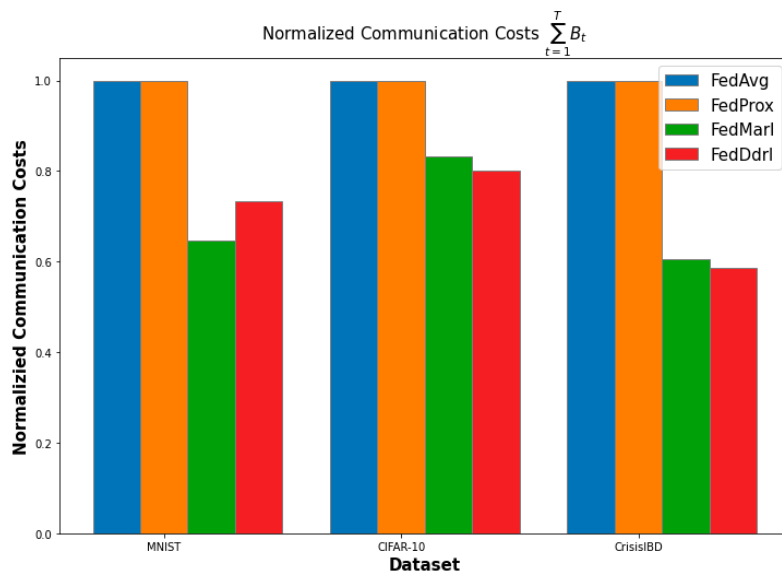


Figure 9. Normalized communication cost of each FL algorithm.

5.2. Strategy Learned by FedDdrl

In this section, we analyze the strategies learned by the FedDdrl algorithm to fully utilized its computing resources while reducing communication costs. Figures 10 and 11 show the early client termination strategy learned by FedDdrl. Blue dots indicate that the client was chosen for complete training, while red dots indicate early client termination.

First, FedDdrl generally picked lesser clients for complete training in the early phase of FL training. From Figures 10 and 11, it is noticed that only three clients were selected for complete training in the first two communication rounds $t = \{1, 2\}$. This is because DNNs usually learn the low-complexity features before learning the higher-complexity features. The former is more robust to noises [43] and can be learned with fewer data [14]. This allows FedDdrl to reduce communication costs by terminating most clients from training in the early phase, where the MobileNetV2 is still learning low-level features. Starting from round $t = 3$ to $t = 6$, the number of clients that underwent complete training increased from 6 to 10. This indicates that MobileNetV2 was beginning to learn higher-level features that require more training data. For the remaining rounds, the number of selected clients was roughly five. Second, FedDdrl preferred clients with a lower probing loss for complete training, which is aligned with the findings in FedMarl [14]. Third, FedDdrl tended to pick clients with shorter probing latency for complete training to reduce the total latency of FL training.

As mentioned earlier, conventional FL training sets the same local epoch E_t^n for all clients, disregarding their computing resources. Hence, one of the contributions of FedDdrl is to learn the optimal strategy to adjust the local epoch count for each client dynamically. In Figure 12, we plotted the local epoch count corresponding to each selected client from the scenario in Figure 11. Bigger dots indicate that a higher local epoch count was assigned for the corresponding clients. It was found that FedDdrl tended to set a lower epoch count

(smaller dots) for clients with higher probing latency. This method could reduce the total training latency since clients with limited computing power did not have to participate in long training epochs. On the other hand, clients with lower probing latency tended to have a higher epoch count. This strategy can fully utilize the computing resource of clients with stronger computing power since they can continue training while waiting for other clients to finish. However, this was not always the case, as shown in Figure 13. On certain occasions, FedDdrl set a high epoch count for clients with long probing latency if the data in these clients were crucial for FL convergence. In any case, FedDdrl was superior to FedMarl, where the FedMarl could either select or terminate a client without the third option of selecting the clients and dynamically tuning the local epoch.

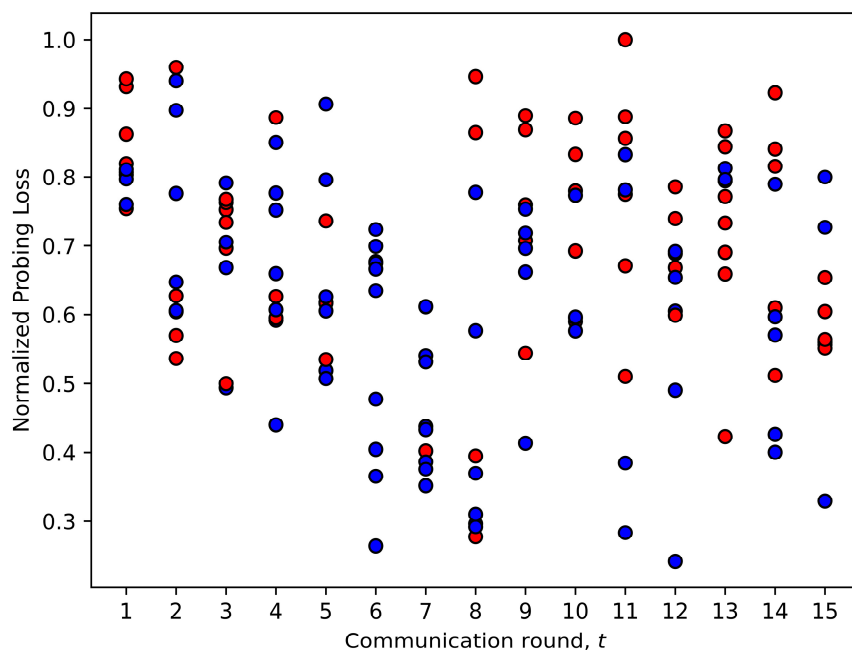


Figure 10. Decisions made by Fiddly based on probing loss.

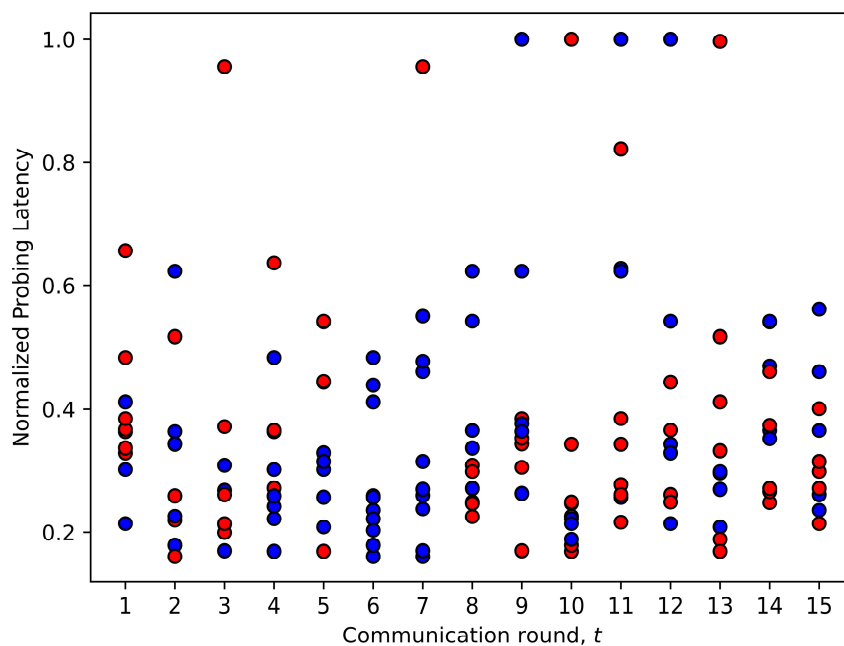


Figure 11. Decisions made by FedDdrl based on probing latency.

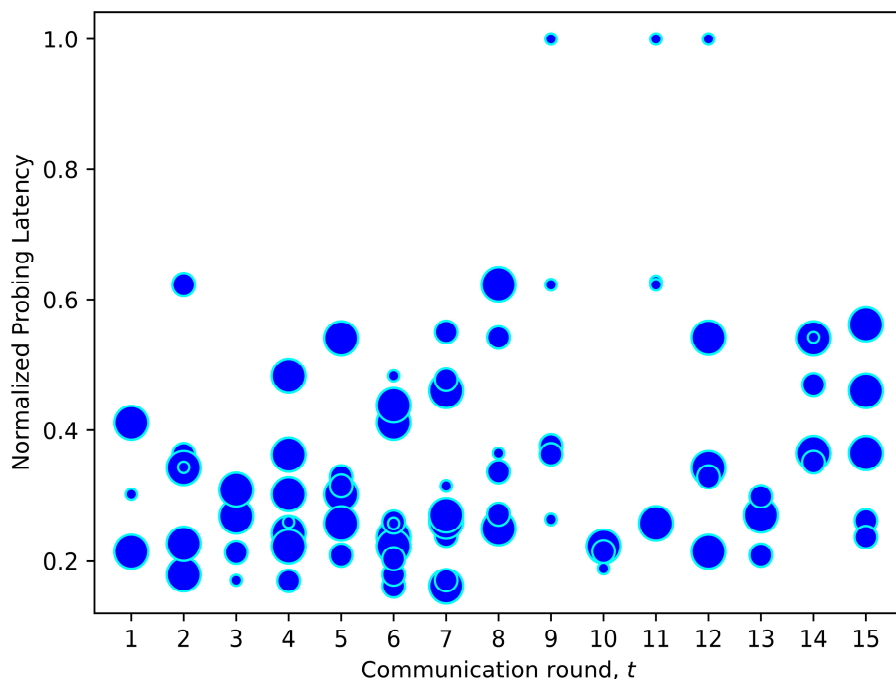


Figure 12. Local epoch count for each selected client in Figure 10. Clients with high probing latency were assigned smaller epoch counts so that the clients could finish local training earlier.

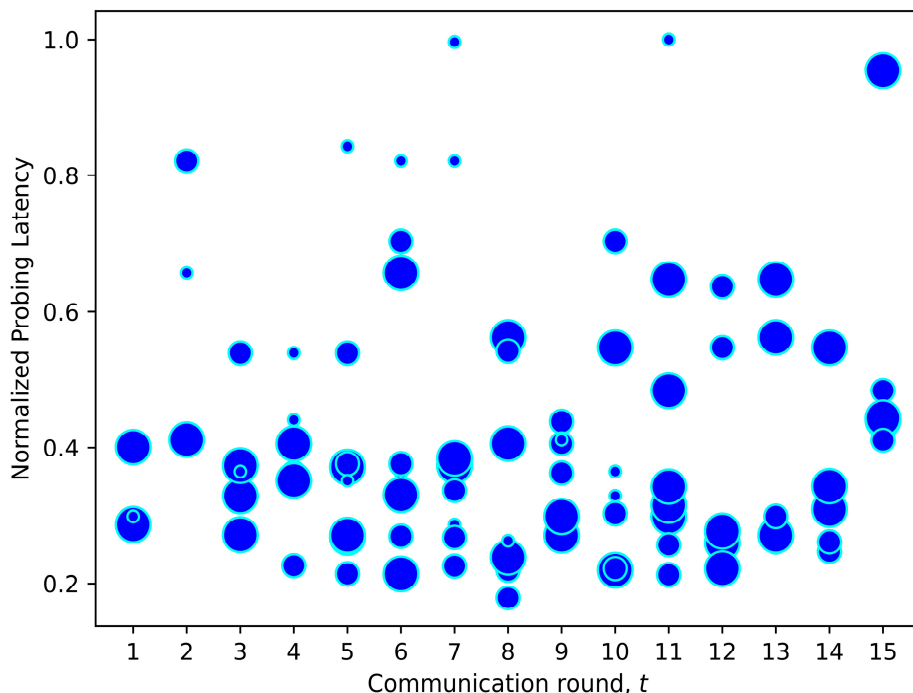


Figure 13. Another example of local epoch adjustment strategy learned by FedDdrl. Selected clients with high probing latency were occasionally assigned large epoch counts to assist the FL convergence.

5.3. How FedDdrl Optimizes the Three Objectives Simultaneously

The objectives of FedDdrl are to (i) maximize the global model’s accuracy while minimizing the (ii) FL system’s training latency and (iii) communication cost. First, VDN 1 will perform early client termination to terminate clients who are not essential for training. By doing so, we can reduce the total communication cost. Additionally, VDN 1 prefers clients with lower probing latency (which also translates to lower training latency). Thus, VDN 1 plays a huge role in reducing both communication costs and training latency. Second,

VDN 2 will dynamically adjust the local epoch count. Clients with limited computing power only have to train with a lesser epoch, so they can finish training earlier. Meanwhile, VDN 2 assigns a higher epoch count to clients with stronger computing power, so they can continue training while waiting for the slower clients. Hence, VDN 2 reduces the training latency and fully utilizes clients with stronger computing power.

Lastly, the global model's accuracy must be retained. When VDN 1 performs client termination, it is essentially reducing C . Intuitively, reducing C seems counter-productive since it reduces the total number of clients participating in training for each communication round t . Fewer clients translate to fewer training data. However, [44] showed that in a non-IID setup, the convergence rate of FedAvg had a weak dependence on C . This makes sense, as some clients may have local datasets with a huge EMD distance from the global distribution. Training the FL model using these clients may hinder the convergence rate. Additionally, [14] showed that using DRL in client selection (or early client termination) can positively affect the convergence rate. This is because selecting useful clients (with useful data) can improve the quality of the overall FL data, which is more crucial than increasing the quantity of data.

In short, FedDdrl can reduce communication cost and training latency without sacrificing model accuracy via early client termination due to the weak correlation between convergence rate and C .

5.4. Computational Complexity Analysis

FedDdrl is composed of a finite number of MLPs. In MLP, let L , n_0 and n_i denote the layer numbers, the size of the input layer (which corresponds to the client state's size) and the number of neurons in i -th layer, respectively. During training mode, the computational complexity for an MLP to update its weight in each step can be expressed as $O(N_b(n_0n_1 + \sum_{i=1}^{L-1} n_in_{i+1}))$ [45]. In total, it takes $N_{ep} \times T$ steps for the FedDdrl algorithm to finish training. Hence, the total training computational complexity of FedDdrl is $O(N_{ep}TN_b(n_0n_1 + \sum_{i=1}^{L-1} n_in_{i+1}))$. The high computation complexity of the MLP can be performed offline using a powerful device (i.e., the FL server). In the online deployment mode, the computational complexity in each step is dramatically reduced to $O(n_0n_1 + \sum_{i=1}^{L-1} n_in_{i+1})$. This is done by cutting off the training procedure, which requires feedforward and backpropagation of N_b data points. Thus, the computational complexity is retained at a favorable level.

5.5. Why Balanced-MixUp Helps in Federated Learning

Without loss of generality, we explored how balanced-MixUp mitigates weight divergence in FL assuming the amount of training data for each class is uniform in the global population. Under this setting, we can express the global distribution $p(y = i)$ for all labels $i = \{1, 2, 3, \dots, n_c\}$ as shown in Equation (21):

$$p(y = i) = \frac{1}{n_c} \quad (21)$$

In this study, a fraction σ of the local training dataset is sampled from one random label, while the remaining $1 - \sigma$ fraction is sampled uniformly from the remaining labels. Following this assumption, let $i = 1$ be the majority class in each client and $i = \{2, 3, \dots, n_c\}$ be the minority classes (whichever i can be the majority class since the ordering does not affect the approximation of client distribution). Without balanced-MixUp, we can express the client dataset distribution $p^k(y = i)$ as Equation (22):

$$p^k(y = i) = \begin{cases} \sigma, & i = 1 \\ \frac{1-\sigma}{n_c-1}, & i = \{2, 3, \dots, n_c\} \end{cases} \quad (22)$$

Based on Equations (21) and (22), the EMD between local and global distribution for FedAvg without balanced-MixUp, denoted as EMD_{ori} , can be written as Equation (23):

$$\begin{aligned} EMD_{ori} &= \sum_{i=1}^{n_c} \| p^k(y=i) - p(y=i) \| \\ &= \left\| \sigma - \frac{1}{n_c} \right\| + (n_c - 1) \left\| \frac{1-\sigma}{n_c-1} - \frac{1}{n_c} \right\| \end{aligned} \quad (23)$$

On the other hand, the client dataset distribution with balanced-MixUp $p_{MixUp}^k(y=i)$ can be expressed as shown in Equation (24):

$$p_{MixUp}^k(y=i) = \begin{cases} \mathbb{E}(\lambda), & i = 1 \\ \frac{1-\mathbb{E}(\lambda)}{n_c-1}, & i = \{2, 3, \dots, n_c\} \end{cases} \quad (24)$$

where $\mathbb{E}(\lambda)$ is the expected value of $\lambda \sim \text{Beta}(\alpha, \beta)$. $\mathbb{E}(\lambda)$ can be written as Equation (25):

$$\mathbb{E}(\lambda) = \frac{\alpha}{\alpha + \beta} \quad (25)$$

Based on Equations (21) and (24), the EMD between FedAvg with balanced-MixUp denoted as EMD_{MixUp} can be written as Equation (26):

$$\begin{aligned} EMD_{MixUp} &= \sum_{i=1}^{n_c} \| p_{MixUp}^k(y=i) - p(y=i) \| \\ &= \left\| \mathbb{E}(\lambda) - \frac{1}{n_c} \right\| + (n_c - 1) \left\| \frac{1-\mathbb{E}(\lambda)}{n_c-1} - \frac{1}{n_c} \right\| \end{aligned} \quad (26)$$

Take our experiments using CIFAR-10 as example, where $\sigma = 0.8$, $n_c = 10$, $\lambda \sim \text{Beta}(0.4, 0.4)$ and $\mathbb{E}(\lambda) = 0.5$. Based on Equations (23) and (26), the EMD_{ori} and EMD_{MixUp} can be computed as 0.777 and 0.444, respectively. This shows that balanced-MixUp could significantly reduce the weight divergence caused by the EMD between p^k and p .

The above proposition is aligned with our experiments. Table 5 shows the performance of FedAvg with and without balanced-MixUp in different C . Balanced-MixUp provided a drastic accuracy boost to FedAvg in all datasets. The accuracy improvement was as high as 17.0% for the CrisisIBD dataset when $C = 10$. This shows that balanced-MixUp can effectively mitigate the weight divergence caused by the EMD, especially when a low number of clients participate in a communication round t . Note that for the MNIST dataset ($C = 10$), the accuracy of FedAvg with balanced-MixUp was slightly poorer than its counterpart, lagging behind by merely 1.4%. This is reasonable, as MNIST is considered a simple task [32] in which FedAvg could perform similarly in certain non-IID settings. Another notable observation is that FedAvg with balanced-MixUp significantly outperformed its counterpart in both the challenging CIFAR-10 and CrisisIBD datasets. The observation is consistent in both $C = 5$ and $C = 10$ experiments. This is encouraging because it proves that balanced-MixUp is useful in mitigating non-IID label shifts, especially for algorithms like FedMarl and FedDdrl which do not have a fixed value of C .

Table 5. Performance of FedAvg with and without Balanced-MixUp.

	Method	MNIST (K=100)	CIFAR-10 (K=100)	CrisisIBD (K=98)
C = 5	FedAvg	78.9% ± 9.3%	62.7% ± 2.9%	42.0% ± 3.4%
	FedAvg with Balanced-MixUp	88.1% ± 3.6%	69.4% ± 2.4%	52.9% ± 4.2%
C = 10	FedAvg	94.6% ± 2.1%	72.8% ± 3.9%	43.2% ± 5.5%
	FedAvg with Balanced-MixUp	93.2% ± 2.0%	76.5% ± 1.7%	60.2% ± 1.5%

6. Conclusions

In this paper, we proposed a DDRL-based FL framework (FedDdrl) for adaptive early client termination and local epoch adjustment. FedDdrl can terminate clients with high probing latency to reduce total training latency and communication costs, and it can automatically adjust the local epoch to fully utilize clients' computing resources. We also showed that balanced-MixUp is a useful augmentation technique to mitigate the impact of weight divergence arising from non-IID label shifts in FL. The simulation results on MNIST, CIFAR-10 and CrisisIBD confirmed that FedDdrl outperformed the comparison schemes in terms of the model's accuracy, training latency and communication costs of FL under extreme non-IID settings. As a future work, we would explore the performance of FedDdrl on other types of non-IID settings, such as feature distribution skew and quantity skew.

Author Contributions: Conceptualization, Y.J.W. and M.-L.T.; methodology, Y.J.W.; software, Y.J.W.; validation, Y.J.W., M.-L.T., B.-H.K. and Y.O.; formal analysis, Y.J.W. and M.-L.T.; resources, M.-L.T.; writing—original draft preparation, Y.J.W.; writing—review and editing, M.-L.T., B.-H.K. and Y.O.; visualization, Y.J.W. and M.-L.T.; supervision, M.-L.T. and B.-H.K.; project administration, M.-L.T.; funding acquisition, M.-L.T. All authors have read and agreed to the published version of the manuscript.

Funding: National Institute of Information and Communications Technology (NICT): ICT Virtual Organization of ASEAN Institutes and NICT (ASEAN IVO).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The ASEAN IVO (http://www.nict.go.jp/en/asean_ivo/index.html) (accessed on 22 December 2022) project, Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network, was involved in the production of the contents of this work and financially supported by NICT (<http://www.nict.go.jp/en/index.html>) (accessed on 22 December 2022). Also, we gratefully appreciate the anonymous reviewers' valuable reviews and comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Al-Maslmani, N.; Abdallah, M.; Ciftler, B.S. Secure Federated Learning for IoT Using DRL-Based Trust Mechanism. In Proceedings of the 2022 International Wireless Communications and Mobile Computing, IWCMC 2022, Dubrovnik, Croatia, 30 May–3 June 2022; pp. 1101–1106. [\[CrossRef\]](#)
2. Reinsel, D.; Gantz, J.; Rydning, J. The Digitization of the World from Edge to Core. *Fram. Int. Data Corp.* **2018**, *16*, 16–44.
3. Sheller, M.J.; Edwards, B.; Reina, G.A.; Martin, J.; Pati, S.; Kotrotsou, A.; Milchenko, M.; Xu, W.; Marcus, D.; Colen, R.R.; et al. Federated Learning in Medicine: Facilitating Multi-Institutional Collaborations without Sharing Patient Data. *Sci. Rep.* **2020**, *10*, 12598. [\[CrossRef\]](#) [\[PubMed\]](#)
4. McMahan, B.H.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, FL, USA, 20–22 April 2017. [\[CrossRef\]](#)
5. Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; Ramage, D. Federated Learning for Mobile Keyboard Prediction. *arXiv* **2018**, arXiv:1811.03604. [\[CrossRef\]](#)
6. Ahmed, L.; Ahmad, K.; Said, N.; Qolomany, B.; Qadir, J.; Al-Fuqaha, A. Active Learning Based Federated Learning for Waste and Natural Disaster Image Classification. *IEEE Access* **2020**, *8*, 208518–208531. [\[CrossRef\]](#)
7. Wong, Y.J.; Tham, M.-L.; Kwan, B.-H.; Gnanamuthu, E.M.A.; Owada, Y. An Optimized Multi-Task Learning Model for Disaster Classification and Victim Detection in Federated Learning Environments. *IEEE Access* **2022**, *10*, 115930–115944. [\[CrossRef\]](#)
8. Reina, G.A.; Gruzdev, A.; Foley, P.; Perepelkina, O.; Sharma, M.; Davidyuk, I.; Trushkin, I.; Radionov, M.; Mokrov, A.; Agapov, D.; et al. OpenFL: An Open-Source Framework for Federated Learning. *arXiv* **2021**, arXiv:2105.06413. [\[CrossRef\]](#)
9. Chen, X.; Li, Z.; Ni, W.; Wang, X.; Zhang, S.; Xu, S.; Pei, Q. Two-Phase Deep Reinforcement Learning of Dynamic Resource Allocation and Client Selection for Hierarchical Federated Learning. In Proceedings of the 2022 IEEE/CIC International Conference on Communications in China, ICC 2022, Foshan, China, 11–13 August 2022; pp. 518–523. [\[CrossRef\]](#)
10. Yang, W.; Xiang, W.; Yang, Y.; Cheng, P. Optimizing Federated Learning with Deep Reinforcement Learning for Digital Twin Empowered Industrial IoT. *IEEE Trans. Industr. Inform.* **2022**, *19*, 1884–1893. [\[CrossRef\]](#)

11. Zhang, W.; Yang, D.; Wu, W.; Peng, H.; Zhang, N.; Zhang, H.; Shen, X. Optimizing Federated Learning in Distributed Industrial IoT: A Multi-Agent Approach. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3688–3703. [[CrossRef](#)]
12. Liu, L.; Zhang, J.; Song, S.H.; Letaief, K.B. Client-Edge-Cloud Hierarchical Federated Learning. In Proceedings of the IEEE International Conference on Communications 2020, Dublin, Ireland, 7–11 June 2020. [[CrossRef](#)]
13. Song, Q.; Lei, S.; Sun, W.; Zhang, Y. Adaptive Federated Learning for Digital Twin Driven Industrial Internet of Things; Adaptive Federated Learning for Digital Twin Driven Industrial Internet of Things. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021. [[CrossRef](#)]
14. Zhang, S.Q.; Lin, J.; Zhang, Q. A Multi-Agent Reinforcement Learning Approach for Efficient Client Selection in Federated Learning. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 9091–9099. [[CrossRef](#)]
15. Abdulrahman, S.; Tout, H.; Ould-Slimane, H.; Mourad, A.; Talhi, C.; Guizani, M. A Survey on Federated Learning: The Journey from Centralized to Distributed on-Site Learning and Beyond. *IEEE Internet Things J.* **2021**, *8*, 5476–5497. [[CrossRef](#)]
16. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated Learning with Non-IID Data. *arXiv* **2018**, arXiv:1806.00582. [[CrossRef](#)]
17. Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Vincent Poor, H. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, 6–12 December 2020. [[CrossRef](#)]
18. Park, J.; Yoon, D.; Yeo, S.; Oh, S. AMBLE: Adjusting Mini-Batch and Local Epoch for Federated Learning with Heterogeneous Devices. *J. Parallel. Distrib. Comput.* **2022**, *170*, 13–23. [[CrossRef](#)]
19. Zhang, H.; Xie, Z.; Zarei, R.; Wu, T.; Chen, K. Adaptive Client Selection in Resource Constrained Federated Learning Systems: A Deep Reinforcement Learning Approach. *IEEE Access* **2021**, *9*, 98423–98432. [[CrossRef](#)]
20. Zhang, P.; Wang, C.; Jiang, C.; Han, Z. Deep Reinforcement Learning Assisted Federated Learning Algorithm for Data Management of IIoT. *IEEE Trans. Industr. Inform.* **2021**, *17*, 8475–8484. [[CrossRef](#)]
21. Wang, H.; Kaplan, Z.; Niu, D.; Li, B. Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. In Proceedings of the IEEE INFOCOM 2020–IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020; pp. 1698–1707. [[CrossRef](#)]
22. Galdran, A.; Carneiro, G.; González Ballester, M.A. Balanced-MixUp for Highly Imbalanced Medical Image Classification. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021, Proceedings of the 24th International Conference, Strasbourg, France, 27 September–1 October 2021*; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2021; Volume 12905, pp. 323–333. [[CrossRef](#)]
23. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. *arXiv* **2018**, arXiv:1812.06127. [[CrossRef](#)]
24. Nishio, T.; Yonetani, R. Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019. [[CrossRef](#)]
25. Zheng, J.; Li, K.; Tovar, E.; Guizani, M. Federated Learning for Energy-Balanced Client Selection in Mobile Edge Computing. In Proceedings of the 2021 International Wireless Communications and Mobile Computing, IWCMC 2021, Harbin, China, 28 June–2 July 2021; pp. 1942–1947. [[CrossRef](#)]
26. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602. [[CrossRef](#)]
27. Vinyals, O.; Babuschkin, I.; Czarniecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature* **2019**, *575*, 350–354. [[CrossRef](#)]
28. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv* **2017**, arXiv:1712.01815. [[CrossRef](#)]
29. Han, M.; Sun, X.; Zheng, S.; Wang, X.; Tan, H. Resource Rationing for Federated Learning with Reinforcement Learning. In Proceedings of the 2021 Computing, Communications and IoT Applications (ComComAp), Shenzhen, China, 26–28 November 2021; pp. 150–155. [[CrossRef](#)]
30. Xiong, Z.; Cheng, Z.; Xu, C.; Lin, X.; Liu, X.; Wang, D.; Luo, X.; Zhang, Y.; Qiao, N.; Zheng, M.; et al. Facing Small and Biased Data Dilemma in Drug Discovery with Federated Learning. *bioRxiv* **2020**. [[CrossRef](#)]
31. Jallepalli, D.; Ravikumar, N.C.; Badarinath, P.V.; Uchil, S.; Suresh, M.A. Federated Learning for Object Detection in Autonomous Vehicles. In Proceedings of the IEEE 7th International Conference on Big Data Computing Service and Applications, BigDataService, Oxford, UK, 23–26 August 2021; pp. 107–114. [[CrossRef](#)]
32. Li, Q.; Diao, Y.; Chen, Q.; He, B. Federated Learning on Non-IID Data Silos: An Experimental Study. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Virtual, 9–12 May 2021; pp. 965–978. [[CrossRef](#)]
33. Sunehag, P.; Lever, G.; Gruslys, A.; Marian Czarniecki, W.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning. *arXiv* **2017**, arXiv:1706.05296. [[CrossRef](#)]
34. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. Mixup: Beyond Empirical Risk Minimization. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018. Conference Track Proceedings 2017. [[CrossRef](#)]

35. Thulasidasan, S.; Chennupati, G.; Bilmes, J.A.; Bhattacharya, T.; Michalak, S. On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks. *Adv. Neural. Inf. Process. Syst.* **2019**, *32*.
36. Zhou, Z.; Qi, L.; Shi, Y. Generalizable Medical Image Segmentation via Random Amplitude Mixup and Domain-Specific Image Restoration. In Proceedings of the 17th European Conference, Computer Vision–ECCV 2022, Tel Aviv, Israel, 23–27 October 2022; Springer: Cham, Switzerland, 2020; pp. 420–436. [[CrossRef](#)]
37. Sun, L.; Xia, C.; Yin, W.; Liang, T.; Yu, P.S.; He, L. Mixup-Transformer: Dynamic Data Augmentation for NLP Tasks. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020; pp. 3436–3440. [[CrossRef](#)]
38. Guo, H.; Mao, Y.; Zhang, R. Augmenting Data with Mixup for Sentence Classification: An Empirical Study. *arXiv* **2019**, arXiv:1905.08941. [[CrossRef](#)]
39. Chou, H.P.; Chang, S.C.; Pan, J.Y.; Wei, W.; Juan, D.C. Remix: Rebalanced Mixup. In Proceedings of the Computer Vision–ECCV 2020 Workshops, Glasgow, UK, 23–28 August 2020; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Volume 12540, pp. 95–110. [[CrossRef](#)]
40. Alam, F.; Alam, T.; Ofli, F.; Imran, M. Social Media Images Classification Models for Real-Time Disaster Response. *arXiv* **2021**, arXiv:2104.04184v1. [[CrossRef](#)]
41. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
42. Tensorflow Federated Using TFF for Federated Learning Research | TensorFlow Federated. Available online: https://www.tensorflow.org/federated/tff_for_research (accessed on 25 December 2022).
43. Xu, Z.-Q.J.; Zhang, Y.; Luo, T.; Xiao, Y.; Ma, Z. Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks. *Commun. Comput. Phys.* **2019**, *28*, 1746–1767. [[CrossRef](#)]
44. Li, X.; Huang, K.; Yang, W.; Wang, S.; Zhang, Z. On the Convergence of FedAvg on Non-IID Data. *arXiv* **2019**, arXiv:1907.02189. [[CrossRef](#)]
45. Yang, H.; Xiong, Z.; Zhao, J.; Niyato, D.; Xiao, L.; Wu, Q. Deep Reinforcement Learning Based Intelligent Reflecting Surface for Secure Wireless Communications. *IEEE Trans. Wirel. Commun.* **2020**, *20*, 375–388. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

Article

Mobility-Aware Resource Allocation in IoRT Network for Post-Disaster Communications with Parameterized Reinforcement Learning

Homayun Kabir ¹, Mau-Luen Tham ^{1,*} , Yoong Choon Chang ¹, Chee-Onn Chow ²  and Yasunori Owada ³

¹ Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Sungai Long Campus, Kajang 43000, Malaysia; homayun@utar.my (H.K.)

² Department of Electrical Engineering, Faculty of Engineering, Universiti Malaya, Lembah Pantai, Kuala Lumpur 50603, Malaysia

³ Resilient ICT Research Center, Network Research Institute, National Institute of Information and Communications Technology (NICT), Tokyo 184-8795, Japan

* Correspondence: thamml@utar.edu.my

Abstract: Natural disasters, including earthquakes, floods, landslides, tsunamis, wildfires, and hurricanes, have become more common in recent years due to rapid climate change. For Post-Disaster Management (PDM), authorities deploy various types of user equipment (UE) for the search and rescue operation, for example, search and rescue robots, drones, medical robots, smartphones, etc., via the Internet of Robotic Things (IoRT) supported by cellular 4G/LTE/5G and beyond or other wireless technologies. For uninterrupted communication services, movable and deployable resource units (MDRUs) have been utilized where the base stations are damaged due to the disaster. In addition, power optimization of the networks by satisfying the quality of service (QoS) of each UE is a crucial challenge because of the electricity crisis after the disaster. In order to optimize the energy efficiency, UE throughput, and serving cell (SC) throughput by considering the stationary as well as movable UE without knowing the environmental priori knowledge in MDRUs aided two-tier heterogeneous networks (HetsNets) of IoRT, the optimization problem has been formulated based on emitting power allocation and user association combinedly in this article. This optimization problem is nonconvex and NP-hard where parameterized (discrete: user association and continuous: power allocation) action space is deployed. The new model-free hybrid action space-based algorithm called multi-pass deep Q network (MP-DQN) is developed to optimize this complex problem. Simulations results demonstrate that the proposed MP-DQN outperforms the parameterized deep Q network (P-DQN) approach, which is well known for solving parameterized action space, DQN, as well as traditional algorithms in terms of reward, average energy efficiency, UE throughput, and SC throughput for motionless as well as moveable UE.

Keywords: post disaster communication; internet of robotic things (IoRT); movable and deployable resource units (MDRU); deep reinforcement learning (DRL); parameterized action space; multi-pass deep Q network (MP-DQN)



Citation: Kabir, H.; Tham, M.-L.; Chang, Y.C.; Chow, C.-O.; Owada, Y. Mobility-Aware Resource Allocation in IoRT Network for Post-Disaster Communications with Parameterized Reinforcement Learning. *Sensors* **2023**, *23*, 6448. <https://doi.org/10.3390/s23146448>

Academic Editors: Jose Manuel Molina López and Alessandra Rizzardi

Received: 10 March 2023

Revised: 4 May 2023

Accepted: 15 June 2023

Published: 17 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Due to rapid climate change, natural catastrophes, including earthquakes, floods, landslides, tsunamis, wildfires, and hurricanes, have frequently occurred worldwide, directly affecting humanity by direct and secondary deaths of human, economic, and environmental losses [1,2]. Recently, authorities have deployed various types of robots and drones such as unmanned ground vehicles (UGVs), unmanned aerial vehicles (UAVs), unmanned underwater vehicles (UUVs), mobile robots, health care robots, etc. that can be defined as user equipment (UE) for post-disaster management (PDM) because they can be dispatched to locations which cannot be accessed or too risky to be accessed by

human rescuers after a disaster has occurred and yet work nonstop [3–6]. Furthermore, the Internet of Things (IoT) and robotic technologies have recently been combined in order to expand the functional capabilities of these robots, commonly called the Internet of Robotic Things (IoRT) [7–10]. The communication between IoRT devices can be provided by 4G/LTE/5G and beyond cellular communication, which can be the heterogeneous network (HetNet) [11]. Residents in the affected area are unable to express their demands and circumstances when regular IoRT/IoT networks are substantially compromised due to the disaster [12,13]. However, victims frequently require essential services, for example, food, water, medical assistance, and shelter, which must be rapidly arranged within 72 h after the disaster to save lives and mitigate losses. As a result, the immediate requirement is for rapid and effective post-disaster network rebuilding [13].

In a post-disaster scenario, Movable and Deployable Resource Units (MDRUs) developed by Tohoku University (TU), Japan, and Nippon Telegraph and Telephone (NTT) can be adopted as SBSs to restore the network coverage and capacity due to the rapid deployment, flexibility, interoperability, and resilience [14]. On the other hand, the UAV-aided cellular network has been regarded as a crucial solution for PDM; however, UAVs can support a maximum of one hour due to the power limitation [15]. Due to the electric power unavailability for a long time in disaster-affected areas, van-type MDRU has around seven hours of battery life, similar to the battery backup of the small base station (SBS), was conducted the field test for a comprehensive solution to satisfy the demands of UEs in disaster areas [16]. In addition, the authors of [17] recommended deploying MDRUs to provide communication services where few SBSs have been interrupted due to a small disaster; in contrast, others are in working condition to reconstruct the whole cellular network. Furthermore, MDRU has been deployed to provide communication services and process the big data for minimizing the latency that is important to find injured people, animals, and damaged infrastructure in the disaster-affected area [18]. It was also used to build heterogeneous wireless IoT networks to sense, exchange, and monitor natural disasters by humanitarian organizations [19]. The authors [20] reconstructed the communication network deploying MDRU in the disaster area and found the best results in small coverage and dense area. Furthermore, the intelligent post-disaster network was developed using big crowd data. The authors deployed MDRUs connected with multiple still-alive base stations by using the virtual vertex [21]. The author in [22] recommended building a resilient IoT network by deploying MDRU, which is connected to a backbone network. In summary, all still-alive SBSs and MDRUs (replacement of damaged SBSs) are generally associated with MBS through wireless backhaul connection to handle the vast UE data generated for PDM with the quality of service (QoS). The whole system can be called two-tier HetNet, as illustrated in Figure 1.

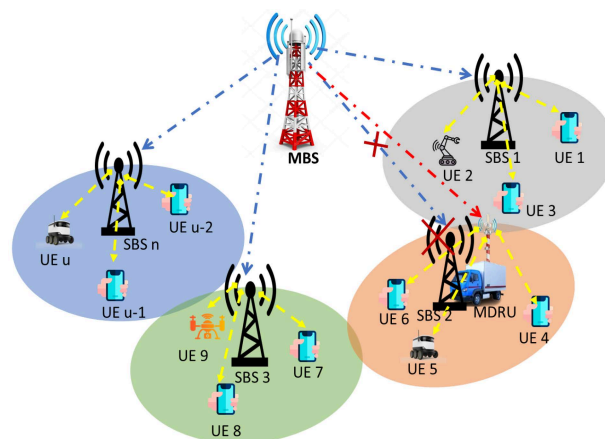


Figure 1. MDRU aided wireless communication after disaster.

The mobility of UEs is one of the critical points to collecting the data in the disaster-affected area for PDM, which impacts channel conditions, path loss, shadow effect, etc., and is a more realistic phenomenon. Our impression is that only a few studies have been conducted based on the mobility of UEs. In [23], we developed a twin delayed deep deterministic policy gradient (TD3) based power allocation algorithm considering UE mobility in one tier IoRT network; however, the major limitation of that research is UE association was not considered. In [24], power allocation optimization is conducted by convex optimization. However, most of the formulated problems, for example, dynamic PA, maximization of the coverage area, traffic offloading, traffic load balancing with user association, maximization of sum rate, etc., are strongly nonconvex as well as nondeterministic polynomial-time hardness (NP-hard) [25]. In this research, we investigate optimizing the energy efficiency and throughput of UE as well as serving cell (SC) of the MDRU-aided two-tier HetNet scenario by ensuring the QoS of mobility-aware UEs where user association and power allocation for each UE have been considered without knowing the environmental priori knowledge. Hence, this optimization problem is strongly nonconvex as well as NP-hard.

Deep Reinforcement Learning (DRL) algorithms (one of the most potent AI algorithms) can handle nonconvex and NP-hard optimization problems [25,26] by leveraging the power of deep neural networks to learn a policy that maps states to actions. The reinforcement learning framework provides a way to learn this policy by trial and error through interaction with the environment illustrated in Figure 2. By learning from experience, the agent can gradually improve its performance and find suitable solutions to complex optimization problems. Consequently, DRL has been applied in wireless communication, robots, computer vision, IoT, IoRT, etc. [27]. According to the action space, DRL is classified as discrete action space algorithms, for example, Deep Q network (DQN), Double DQN (DDQN), Rainbow DQN, dueling DQN, etc., continuous action space algorithms; for instance, Deep Deterministic Policy Gradient (DDPG), Twin delayed DDPG (TD3), Distributed Distributional DDPG (D4PG), Soft Actor-Critic (SAC), etc., that are based on policy gradient and hybrid action space algorithms, such as Q-PAMDP, PA-DDPG, Parametrized DQN (P-DQN), Multi-Pass DQN (MP-DQN), etc., that can handle discrete-continuous combinedly [28].

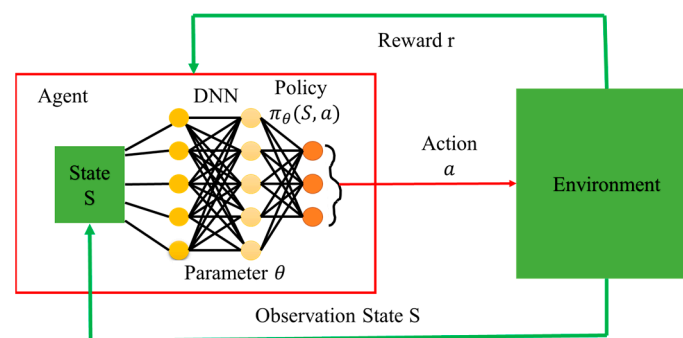


Figure 2. Architecture of deep reinforcement learning (DRL).

In [29], authors investigated a combined strategy for power allocation, which is considered continuous action and user association that is discrete action to improve downlink energy efficiency while ensuring QoS of stationary UEs under standard backhaul connection in HetNet by implementing the hybrid action space-based DRL called P-DQN. Furthermore, the architecture of P-DQN is similar to DDPG. However, discrete action is produced from the Q network, while continuous action is generated from the actor network. In P-DQN, the joint action parameter vector for all whole actions at a one-time step is taken as the input of the Q network. As a result, the gradients concerning all action parameters are calculated, which generates false gradients. In [30], the authors developed MP-DQN (similar architecture as P-DQN) and tested it in well-known Robot Soccer Goal and Half Field Offense games. They forwarded the continuous action parameter with

a standard basis vector to the Q-network. Consequently, it mitigated the effects of false gradients of P-DQN, converged the complex optimization problem, and outperformed P-DQN regarding data efficiency and converged policy performance [28,30]. After that, researchers are applying MP-DQN to solve hybrid action space-based optimization problems in various fields from robotics to communications. MP-DQN was implemented in the Golf simulator to find the best action that consisted of shot selection with height, spin, and speed [31]. In [32], the authors implemented MP-DQN to build a decision tree method for the imbalanced binary classification where the continuous attributes represented the discrete action, and the threshold values were continuous action. For intelligent traffic signal control development, MP-DQN was deployed in [33], considering selecting traffic lights (red, amber, and green) with on–off time intervals. In [34], the authors applied the MP-DQN in an actual robot setup to primitive action (translation, rotation, and insertion) with the end-effector velocity as well as the contact force limits. In wireless communication, MP-DQN has been implemented for task scheduling of the Radio Access Network [35] and joint task offloading and resource allocation in the Non-orthogonal multiple access (NoMA) system [36].

This paper explores MDRU aided two-tier HetNet scenario considering the UE mobility for post-disaster communication. It aims to optimize resource allocation by deploying the parameterized DRL called MP-DQN without knowing the environmental priori knowledge. The main contributions of this article are summarized below:

1. We investigate UE association and power allocation for maximizing the energy efficiency of downlink in MDRUs-aided two-tier HetNet for post-disaster communications by considering the backhaul links of MDRUs/SBSs with MSB where UE association as discrete action space and power allocation as continuous action space combinedly called parameterized action space of DRL has been considered when UEs are stationary.
2. Mobility-aware resource allocation (UE association and power allocation) has been formulated for parameterized DRL to optimize the energy efficiency, the throughput of SBSs/MDRUs, and the throughput of UEs in MDRUs aided two ties HetNet.
3. Model-free and parameterized action space-based MP-DQN algorithm, which utilizes several concurrent batch processing to provide action parameters to the Q network, has been proposed to maximize the energy efficiency, the throughput of SBSs/MDRUs and throughput of UEs of MDRUs aided HetNet.

Note that the proposed framework improves network robustness, which is one of the goals of the ASEAN IVO project titled “Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network”.

2. Related Work

TU, Japan, and NTT are conducting research continuously to improve the MDRU performance in terms of connectivity, serviceability, and coordination during PDM. They deployed the channel allocation algorithm in MDRU and conducted the test successfully in the Philippines and Nepal [37]. Due to the limited power after the disaster, emitting power optimization of rapidly deployable resource units by satisfying the demand of UEs (search and rescue robots, drones, smartphones, etc.) has paid great attention [38]. In [39], authors proposed radio access control based on DRL for selecting the van type MDRUs/relay and optimizing the power of MDRUs. In [40], authors investigated spectrum and energy-efficient methods for providing communication services to UE of MDRU-based networks. The authors [41] analyzed the problem of UAV deployment as MDRU in both standalone deployment scenarios to support fixed SBSs already in place where SBSs are damaged due to malfunction or disaster in HetNet. In addition, they considered that UAVs were connected with remaining SBS or MBS by wireless backhauled, which was essential to serve the UEs by fulfilling their demands.

In [24], we implemented DRL, consisting of two value-based networks for energy-efficient radio resource allocation in IoRT that outperformed the DQN [42], where the UE

demand and status (active or sleep) of BS are considered as state and action is to estimate the status of each BS. In addition, emitting power of active BS to serve UEs was optimized by a convex optimizer. In [43], value-based distributed DRL has been proposed to find user association and resource allocation by ensuring UE QoS. After that, the simulation results were improved by implementing the D3QN consisting of DDQN and dueling architecture in [44], where the degree of satisfaction of UEs was state space, and the selection of BS and transmission channels combinedly were action space. However, emitting power of BS can be adequately optimized when emitting power is conserved as continuous action of the DRL algorithm. In [23], continuous action-based DRL algorithms, TD3, have been applied to estimate the optimal emitting power of BS in the IoRT network by considering the interfering multiple access channel (IMAC). In [45], the authors developed a novel DRL based on DDPG to optimize the joint issue user association and power allocation of BS in HetNet that achieved the load balance and improved the energy efficiency of the network. In [46], a transfer learning algorithm based on DDPG has been developed to optimize the power allocation and ensure user association in HetNet. However, user association is discrete and power allocation is continuous. Hence, to solve the joint optimization problem combined with user association and power allocation, a hybrid action space-based DRL algorithm is the most suitable. In [29], the author formulated the problem of combined user association and power allocation, where user association considers as the discrete action and power allocation is expressed as the continuous action. In addition, P-DQN has been implemented to maximize energy efficiency by satisfying the QoS of each UE. Simulation results of P-DQN outperformed compared to DQN in terms of overall efficiency by satisfying the QoS of stationary UEs.

For PDM, UEs need to move in the vicinity to collect the appropriate information about victims that movement directly affects communication channel quality and throughput. This critical phenomenon has not been taken into account by many academics. The UE mobility model in non-orthogonal multiple access (NOMA), where each UE moved from one point to another with varied directions and speeds, was taken into consideration by the authors [47]. Due to UE mobility, the authors [48] suggested a conventional dynamic power allocation (DPA) method that took the channel circumstances into account and asserted that UE mobility significantly influences NOMA's performance, particularly for downlink throughput. The authors of [49] created a power control method for a wireless network where UE mobility causes changes in the communication channel. In [23], two widespread UE mobility models, (a) modified Gauss–Markov and (b) random walk, have been investigated to maximize the sum rate in dynamic power allocation problems where the TD3-based DRL algorithm has been implemented; however, user association was not considered. In [50], the authors implemented Genetic Algorithm (GA) to allocate the UEs worked to share the information after the disaster in overlapping areas among the appropriate MDRUs. The proposed GA algorithm outperformed greedy and random algorithms as well as the nearest MDRU in terms of latency and QoS. In order to maximize the expected achievable rate of UE in ultra-dense networks, the authors [51] developed a matching game algorithm, where mobility-aware user association was considered by minimizing the handovers number. The authors [52] deployed the DRL algorithm to estimate the transmit timing, routing as well as power allocation for UEs from MDRU deployed in disaster areas where UE mobility, channel states, and energy harvesting were considered.

3. System Model

In this section, we consider two-tier HetNet that consists of one MBS with M active SBSs and N deployed MDRUs (replacement of damaged SBSs due to disaster) where $\mathcal{M} = \{1, 2, \dots, M\}$ and $\mathcal{N} = \{1, 2, \dots, N\}$ are the sets of active SBSs and deployed MDRUs [17,20,50]. The total SCs for PDM is $K = M + N$ where $\mathcal{K} = \{1, 2, \dots, K\}$ is the set of SCs that serve U UEs considering $\mathcal{U} = \{1, 2, \dots, U\}$ is the set of UEs. In addition, we assume two different bands that are 6 GHz and millimeter wave bands for MBS to

SBSs/MDRUs (tier 1) and SCs to UEs (tier 2), respectively. As a result, interference between tiers is not available in this network. For tier 1 downlinks, the antenna array of MBS is larger than the total number of SCs. Furthermore, orthogonal frequency division multiple access (OFDMA) is deployed to communicate from SBSs/MDRUs to UEs where the total subchannels number is N_{sub} . To collect the data and survey the disaster-affected area, UEs need to move from one place to another. Hence, the mobility model of UE has to be considered for PDM. Modified Gauss–Markov is the well-known mobility model of UE, especially for robots and drones that are considered in our research.

3.1. Modified Gauss–Markov Mobility Model

The Modified Gauss–Markov (MGM) mobility model improves past approaches by including temporal dependence. Here, the speed and direction of a UE are updated in line with the recorded values of earlier time periods. The degree of randomness used in calculating these two figures can also be changed based on the features of the simulated wireless network. The MGM mobility paradigm is not stateless since the memory of past actions are retained. Nonetheless, the UE mobility continues to be distinct from that of other mobile terminals linked to the same network [47,53]. According to Figure 3, UE mobility makes possible u^{th} UE to move randomly with average velocity that is indicated as $\Delta\alpha_u(t-1, t)$ and $v_u(t-1, t)$, u^{th} UE are $x_u(t)$ $y_u(t)$ are presented below:

$$x_u(t) = x_u(t-1) + v_u(t-1, t) * \cos(\alpha_u(t-1, t)) * \Delta t, \quad (1)$$

$$y_u(t) = y_u(t-1) + v_u(t-1, t) * \sin(\alpha_u(t-1, t)) * \Delta t, \quad (2)$$

$$\alpha_u(t) = \alpha_u(t-1) + \Delta\alpha_u(t-1, t), \quad (3)$$

where $x_u(t-1)$, $y_u(t-1)$, and $\alpha_u(t-1, t)$ are the x -axis, y -axis, and direction of u^{th} UE at $t-1$ time slot. The distance traveled by u^{th} within Δt can be illustrated by

$$d_u(t-1, t) = \sqrt{(x_u(t-1) - x_u(t))^2 + (y_u(t-1) - y_u(t))^2}. \quad (4)$$

The distance between k^{th} SC and u^{th} UE at t time slot is presented as

$$d_{k,u}(t) = \sqrt{(x_k - x_u(t))^2 + (y_k - y_u(t))^2}, \quad (5)$$

where x_k and y_k are the coordinates of k^{th} SC.

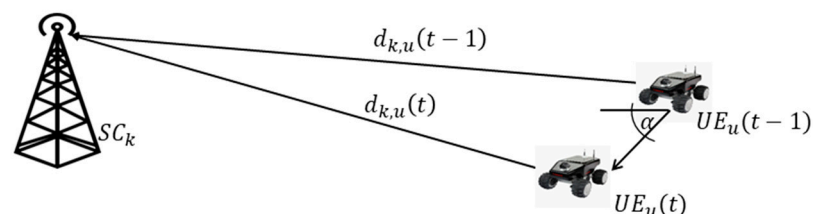


Figure 3. UE Mobility model for UE with random direction and average velocity.

3.2. Network Model

Even though each SBS/MDRU are using OFDMA to serve UEs that construct a cluster of UEs, each UE can only be connected to a single SC. Let's consider k^{th} serving cell serves to u^{th} UE by \mathcal{F}_r frequency subchannel. Here, $c_{ku}(t) = \{0, 1\}$ is represented as the status of user association where $c_{ku}(t) = 1$ denotes the u^{th} UE is associated with k^{th} SC and $c_{ku}(t) = 0$ otherwise. After that, the set of UEs in the cluster k is assumed by $C_k(t) = \{u : c_{ku}(t) = 1, u \in \mathcal{U}\}$. The SBS serves the u^{th} UE can be illustrated by $S_u(t) = \{k : c_{ku}(t) = 1, m \in \mathcal{M}\}$

where $|S_u(t)|$ is one. The set of active SC is at t time slot $\mathcal{K}^{active}(t) = \{k | |C_k(t)| > 0\}$. The channel gain between from k^{th} SC to u^{th} UE can be defined as

$$g_{k,u,f}(t) = \left| h_{k,u,f}(t) \right|^2, \quad (6)$$

where $h_{k,u,f}(t)$ is the channel coefficient when subchannel frequency is f . The signal to interference plus noise ratio (SINR) from k^{th} serving cell to u^{th} UE can be illustrated as follows:

$$SINR_{u,f}(t) = \frac{\sum_{k=1}^K c_{k,u}(t) g_{k,u,f}(t) p_{k,u,f}(t)}{\sigma^2 + I_{u,f}(t)}, \quad (7)$$

where $p_{k,u,f}(t)$ is the allocated power of k^{th} SC for u^{th} UE which must be satisfied the $0 \leq \sum_{u \in C_k} \sum_{f \in \mathcal{F}_R} p_{k,u,f}(t) \leq P_{SC_k,max}$. $P_{SC_k,max}$ is the maximum emitting power from k^{th} SC. The observed interference and noise power by u^{th} UE is $I_{u,f}(t)$ and σ^2 respectively. To ensure no intra-cluster interference in each cluster, we investigate the case in which each SC allots orthogonal subchannels to various UEs within its serving area. Every UE can receive a minimum of one subchannel to transmit the data for data transmission when the cluster size does not exceed the total number of sub-channels. When intra-cluster interference is absent, just inter-cluster interference makes up the interference term $I_{u,f}$ which may be represented as

$$I_{u,f}(t) = \sum_{w \notin C_{S_u}} \sum_{f \in F_u \cap F_w} g_{k,u,f}(t) p_{k,w,f}(t). \quad (8)$$

The spectral efficiency of the u^{th} UE is illustrated as

$$\rho_u(t) = \sum_{f \in \mathcal{F}_k} \log_2(1 + SINR_{u,f}(t)). \quad (9)$$

The user sum-rate for the k^{th} SC is calculated as

$$\rho_k^{SC}(t) = \sum_{u \in C_k} \rho_u(t) = \sum_{u=1}^U c_{k,u}(t) \rho_u(t). \quad (10)$$

The summation of data transmission power and the operating power that is defined as the minimum power requirement to maintain the SC active is defined total consumed power of our network that can be represented as

$$P_{total}(t) = |\mathcal{K}^{active}(t)| \cdot P_{o,SC} + \sum_{k \in \mathcal{K}} \sum_{u \in C_k} \sum_{f \in \mathcal{F}_k} p_{k,u,f}(t). \quad (11)$$

where $P_{o,SC}$ is the operational power of SC. Detailed notation descriptions are summarized in Table 1.

Table 1. Notation summary.

Notation	Definition
$\mathcal{M}, \mathcal{N}, \mathcal{K}$ and \mathcal{U}	set of SBSs, MRDUs, SCs and UEs
M, N, K and U	Total number of SBSs, MDRUs, SCs, and UEs
\mathcal{F}_w	Set of subchannels allocated to u^{th} UE
$S_k(t)$	The SC serving the u^{th} UE at time slot t
B_{sub}	Subchannel bandwidth
N_T	The number of antennas on MBS
$P_{total}(t)$	Total consumed power by active SCs
$g_{k,u,f}(t)$	The channel gain from k^{th} SC to u^{th} UE with f^{th} subchannel at time slot t
$h_{k,u,f}(t)$	The channel coefficient from k^{th} SC to u^{th} UE in f^{th} subchannel at time slot t
$P_{SC_k,max}$	The maximum power available of k^{th} SC
$p_{k,u,f}(t)$	Emitting power from k^{th} SC to u^{th} UE in f^{th} subchannel at time slot t

Table 1. Cont.

Notation	Definition
$ \mathcal{K}^{active}(t) $	Total quantity of active SCs at time slot t
σ^2	Noise power
$I_{u,f}(t)$	Interference observed by u^{th} UE in subchannel f at time slot t
$C_k(t)$	The set of UEs in cluster k at time slot t
$c_{ku}(t)$	Link indicator between k^{th} SC and u^{th} UE at time slot t
$SINR_{uf}(t)$	SINR for u^{th} UE in the f^{th} subchannel at time slot t
v_u	Capacity threshold for u^{th} UE
D_k^{SC}	Maximum downlink data rate for k^{th} SC

We strive for a way that results in optimizing user association and emitting power allocation to maximize the energy efficiency expressed as the achievable sum rate per utilized power in our assumed network by considering the QoS guarantee, and wireless backhaul link capacity constraints without knowing the environmental priori knowledge. The problem can be formulated as

$$c_{ku}(t), p_{k,u,f}(t) \sum_{t=0}^{t=T} \frac{1}{P_{total}(t)} \sum_{u=1}^U \rho_u(t), \quad (12a)$$

$$\text{Subject to } C_1 : \sum_k c_{k,u}(t) = 1, c_{k,u}(t) \in \{0, 1\}, \forall k \in \mathcal{K}, u \in \mathcal{U}, \quad (12b)$$

$$C_2 : 0 \leq \sum_{u \in C_k} \sum_{f \in \mathcal{F}_R} p_{k,u,f}(t) \leq P_{SC_k, max}, \forall k \in \mathcal{K}, u \in \mathcal{U}, \quad (12c)$$

$$C_3 : \rho_u(t) \geq v_u, \forall u \in \mathcal{U}, \quad (12d)$$

$$C_4 : |C_k(t)| \leq |C_k|_{max}, \forall k \in \mathcal{M}, \quad (12e)$$

$$C_5 : \rho_k^{SC}(t) \leq D_k^{SC}, \forall m \in \mathcal{M}. \quad (12f)$$

Each UE is presumed to be serviced by a single SC in C_1 in (12b), and the transmit power limit at the k^{th} SC is discussed in C_2 in (12c), where $P_{SC_k, max}$ is the maximum power that is used at the k^{th} SC. C_3 in (12d), where v_u is the capacity threshold for u^{th} UE denotes the QoS requirement for each UE. The cluster size limitation in (12e) is C_4 , and the maximum number of users in k cluster is $|C_k|_{max}$. To prevent intra-cluster interference, this makes sure that UEs in the same cluster are given distinct subchannels. D_k^{SC} is the highest feasible downlink data rate for k^{th} SC, while C_5 in (12f) is the backhaul connection capacity restriction.

By identifying the best user associations as well as power distribution, which is often a difficult task with a variety of unknowns and hybrid unknown spaces (discrete clustering and continuous power) in the network, the technique in (12a) aims to maximize energy efficiency. Additionally, the optimization issue in (12a) involves a one-shot situation at a certain time instant that must be reassessed as the network advances until the following time instant. We are consequently driven to deploy MP-DQN approaches to address the issues.

4. Deep Reinforcement Learning for Parameterized Action Space

In this section, we illustrate the DRL which can handle the parameterized action space for identifying optimal user association (discrete action) as well as emitting power allocation (continuous action) of SC by satisfying the QoS. The parameterized action space [54] combined with discrete and continuous action space represented as A_d and A_j respectively is illustrated as $\mathcal{A} = \{(j, z_j) | z_j \in A_j \text{ for all } j \in A_d\}$, where $a(t) = (j, z_j)$

is the hybrid action. A discrete action j has been chosen from the discrete action set $A_d = \{j_1, j_2, j_3, \dots, j_J\} = \{[c_{ku}(t)] : c_{ku}(t) = \{0, 1\}, k \in \mathcal{K}, u \in \mathcal{U}\}$. The continuous action parameters for that specific discrete action j is $z_j = \mathbf{p}^{UE}(t) = [p_1^{UE}(t), p_2^{UE}(t), \dots, p_u^{UE}(t)]$, where $p_u^{UE}(t) = [p_{S_u, u, f}(t)]_{f: f \in \mathcal{F}_u}$ for downlink data transmission in all sub channels allocated to u^{th} UE. Furthermore, $z_j \in \mathcal{Z}$, where \mathcal{Z} is the set of continuous actions considering all possible discrete action. According to [55], parameterized action MDP (PAMDP) is presented as $\langle \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma \rangle$. Here, \mathcal{S} represents the state space, the Markov probability of transition is illustrated as \mathcal{P} , the parameterized action space is denoted by \mathcal{A} , the reward is defined \mathcal{R} and the discount fraction is $\gamma \in [0, 1]$. At the t^{th} timeslot, the agent observes the state of environment $s(t) \in \mathcal{S}$ and chooses suitable parameterized action $a(t) \in \mathcal{A}$ based on its policy π . After applying the chosen parameterized action, the immediate reward $r(s(t), a(t))$ is received with next state of environment $s(t+1) \sim \mathcal{P}(s(t+1)|s(t), a(t))$.

To solve the non-convex, the NP-hard and joint optimization problem consists of selecting the user association and allocating the transmitted power of MDRU-aided two-tier HetNet discussed in Section 3 by parameterized DRL, state, action, reward, and experience replay are described below:

State: The data rate of each UE at t^{th} timeslot has been generated from SINR that is calculated considering the user association, emitting power allocation, channel gain, interference, and noise power observed by UE in that specific time slot. Hence, the set of data rate for all UE has been assumed as the state at t^{th} timeslot for DRL agent.

$$s(t) = [\rho_1(t), \dots, \rho_u(t)]. \quad (13)$$

Action: In this optimization problem, discrete (identification of UE association) and continuous (emitting power for each UE from SBS) action spaces at t^{th} timeslot have been combinedly considered as follows:

$$a(t) = [\mathbf{c}^{UE}(t), \mathbf{p}^{UE}(t)], \quad (14)$$

where $\mathbf{c}^{UE}(t) = [c_{ku}(t)], k = 1 : M, u = 1 : U$ with $c_{ku}(t) = \{0, 1\}, k \in \mathcal{K}, u \in \mathcal{U}$ is denoted for UE association with SC. When $c_{ku}(t) = 1$, it means u^{th} UE is associated with k^{th} SC for that specific time slot and otherwise $c_{ku}(t) = 0$. After ensuring the UE association, SC is allocation power to that UE at t^{th} timeslot. The vector of power allocation from SCs at t^{th} timeslot is defined as $\mathbf{p}^{UE}(t) = [p_1^{UE}(t), p_2^{UE}(t), \dots, p_u^{UE}(t)]$.

Reward: The maximization of the overall energy efficiency according to the Equation (12a) is the prime goal of this research by satisfying the QoS of every UE and the constraint capacity of backhaul link of each SBS. Therefore, the reward $r(t)$ at t^{th} time slot is illustrated as:

(a) Reward function one (RFO) [29]:

$$r(s(t), a(t)) = \begin{cases} r'(s(t), a(t)) & \text{if } \rho_k^{SC} \leq R_k^{SC}, \forall k \in \mathcal{K} \\ r'(s(t), a(t)) - r_{th} & \text{if } \rho_k^{SC} > R_k^{SC} \text{ for some } k \in \mathcal{K}, \end{cases} \quad (15)$$

where $r'(s(t), a(t)) \cong \lambda_1 Z_{\alpha_1(t)} - \lambda_2 Z_{\alpha_2(t)}$ with $\alpha_1(t) = \frac{1}{P_T} \sum_{u=1}^U \rho_u(t)$ that is the energy efficiency of system and $\alpha_2(t) = \sum_{u=1}^U (\rho_u(t) - v_u)^2$ which is the penalty term is deployed for discouraging the agent to take the actions, for example, the capacity of every UE huge diverges from the threshold of QoS and $Z_{\alpha_1(t)}$ and $Z_{\alpha_2(t)}$ are the Z-scores of $\alpha_1(t)$ and $\alpha_2(t)$, respectively. In addition, r_{th} is the threshold value is deployed to mitigate the likelihood of violating the backhaul capacity constraint.

(b) Reward function two (RFT):

$$r(s_t, a_t) = \lambda_1 Z_{\alpha_1(t)} + \lambda_2 \sum_{u=1}^U R_{QoS_UE}(t) + \lambda_3 \sum_{m=1}^M R_{QoS_backhaulink}(t), \quad (16)$$

$$\begin{aligned}\alpha_1(t) &= \frac{1}{P_T} \sum_{u=1}^U \rho_u(t), \\ R_{QOS_{UE}}(t) &= \text{if } \rho_u \geq v_u, r = 1 \text{ else } r = -1, \\ R_{QOS_{backhaulink}}(t) &= \text{if } \rho_k^{SC} \leq D_k^{SC}, r = 1 \text{ else } r = -1.\end{aligned}$$

Here, $\lambda_1, \lambda_2, \lambda_3$ are non-negative weights of the corresponding terms and range from 0 to 1.

Experience replay: It is a DRL strategy that utilizes replay memory to record the agent's experiences at each time step in a data set that is pooled over several episodes. After that, a minibatch of experience is selected randomly from the experience replay that is utilized for training. This process solves the problem of autocorrelation leading to unstable training.

Furthermore, three well-known DRL including our proposed method called MP-DQN which can handle parameterized action space are discussed below.

4.1. Deep Q Network

One of the most well-known DRL algorithms is DQN [56], which is value-based and utilized for discrete action space only. The goal of traditional DQN is to find optimized the action by maximizing the action value function $Q_{(s,a)}$ as follows:

$$Q(s, a) \triangleq E \left[\sum_{k=0}^{\infty} \gamma^k r(s(t+k), a(t+k)) | s(t) = s, a(t) = a \right]. \quad (17)$$

The maximization of (17) is equivalent to the Bellman equation and can be described as

$$y(t) = r(t) + \gamma \max_{a(t+1)} Q(s(t+1), a(t+1); \theta^-), \quad (18)$$

where $y(t)$ represents the optimized value of Q. The loss function is represented as

$$L = (y(t) - Q(s(t), a(t); \theta))^2, \quad (19)$$

which mitigates the correlation between current value $Q(s(t), a(t); \theta)$ and target value $y(t)$. In addition, the traditional DQN can be deployed for continuous action space when it is converted into a finite set of discrete action spaces by discretizing the process. Furthermore, DQN can also be utilized for parameterized action space by converting from continuous to discrete action space that concatenates with existing discrete action space. When continuous action has conducted the quantization to reverse discrete action, many action values are generated and those action values may round off. Consequently, the complexity of the DQN exponentially rises with the size of the action space, resulting in very massive power consumption and a delay in convergence speed. To overcome those issues, P-DQN has been deployed to handle the parameterized action space-based optimization problem [28].

4.2. Parameterized Deep Q Learning

P-DQN [57] is a DRL algorithm that handles hybrid (discrete-continuous) action spaces combined without relaxation or approximation. The structure of P-DQN is similar to DDPG, which describes a deterministic function that takes the state as input and produces continuous parameters of each discrete action. After that, generated continuous action parameters are concatenated with the state, which is utilized as input to the Q network to generate the Q values. Finally, the optimal function chooses the best discrete action from generated Q values. Let's consider one actor parameter network $z_j(s; \theta)$ with weight θ and one actor network $Q(s, j, z_j; w)$ with weight w . Furthermore, the weights θ has been estimated by optimizing the expected function of the action-value that are described as $E[Q(s, j, z_j(j; \theta); w)]$. And the weight w has been determined by optimizing the mean squared error $E[(y(t) - [Q(s(t), a(t); w)])^2]$, where $a(t) = (j, z_j)$ and the target value is described as

$$y(t) = r(t) + \gamma \max_{j' \in A_d} Q(s(t+1), j', z_{j'}(s(t+1); \theta^-); \omega^-). \quad (20)$$

In addition, the loss function of the actor parameter and the actor network can be presented as follows:

$$L^x(\theta) = \frac{1}{N} \sum_{t=1}^N Q(s(t), j, z_j(s(t); \theta); \omega), \quad (21)$$

$$L^Q(\omega) = \frac{1}{N} \sum_{t=1}^N (y(t) - Q(s(t), j, z_j(s(t); \theta); \omega))^2. \quad (22)$$

Furthermore, the weights θ and ω are updated according to

$$\theta \leftarrow \theta - \alpha_{a,p} \nabla_{\theta} L^x(\theta), \quad (23)$$

$$\omega \leftarrow \omega - \alpha_a \nabla_{\omega} L^x(\omega), \quad (24)$$

where $\alpha_{a,p}$ and α_a are the learning rate for the actor parameter and actor network.

Even if P-DQN can converge and the impact is excellent, there is still room for improvement in the theory behind discrete and continuous action selection. Updates to any action's continuous action parameter will affect all actions' Q values, not just the Q value linked to the action parameter [28,30].

4.3. Multi Pass Deep Q Learning

The issue of excessive parameterization of P-DQN is resolved by MP-DQN [30] by employing multiple concurrent batch processing to provide action parameters to the Q network. Without altering the P-DQN structure, MP-DQN isolates the continuous parameters and inputs each one into the Q network individually. They executed a forward pass once for each discrete action j where the state s and action parameter vector $z \mathbb{E}_j$ are concatenated as input and \mathbb{E}_j represents the j dimensional standard basis vector. Hence, the joint parameter vector is represented as $\mathcal{Z} \mathbb{E}_j = (0, \dots, 0, z_j, 0, \dots, 0)$ where each $z_i, i \neq j$ is set to zero. As a consequence, the impact of network weights is negated for unassociated action parameters z_j from the input layer where all false gradients are set to zero. Furthermore, Q is only depended on associated z_j where

$$Q(s, j, \mathcal{Z} \mathbb{E}_j) \cong Q(s, j, z_j). \quad (25)$$

To forecast all Q values, c forward passes are necessary as opposed to just one. To perform the multi pass, the capacity of parallel minibatch processing by PyTorch or TensorFlow library can be deployed. A multi-pass with j actions is processed in the same manner as a minibatch of size j :

$$\begin{pmatrix} Q(s, \cdot, \mathcal{Z} \mathbb{E}_1; \theta_Q) \\ \cdot \\ \cdot \\ Q(s, \cdot, \mathcal{Z} \mathbb{E}_j; \theta_Q) \end{pmatrix} = \begin{pmatrix} Q_{11} & \cdots & Q_{1j} \\ \vdots & \ddots & \vdots \\ Q_{j1} & \cdots & Q_{jj} \end{pmatrix}, \quad (26)$$

where the Q-value for action b produced on the a^{th} pass is Q_{ab} . Furthermore, the diagonal elements Q_{aa} is pivotal and deployed in the final output $Q_a \leftarrow Q_{aa}$ as shown in Figure 4. According to [33], MP-DQN makes it easier to choose the best hybrid action by reducing the impact of a single discrete action on other continuous action parameters.

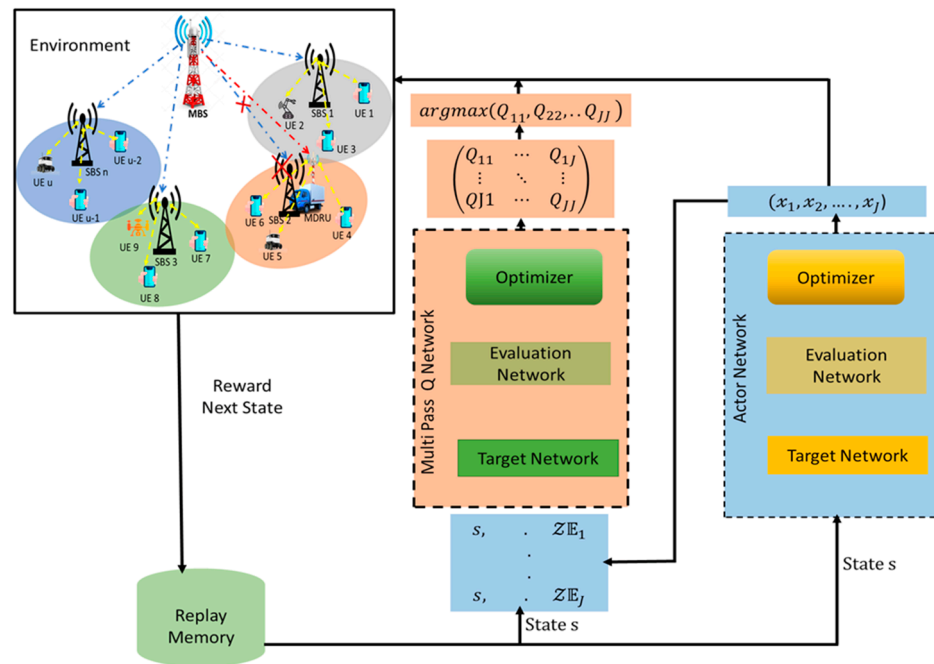


Figure 4. Architecture of multi pass deep Q learning (MP-DQN).

5. Performance Evaluations

In this section, we utilize TensorFlow 1.14.0 on Spyder IDE 3.3.6 in an 11th Gen inter-core i7, 16 GB RAM, and RTX 3060 laptop GPU to demonstrate the simulation scenario. In addition, a HetNet has been considered which consists of one MBS with 100 antennas and 20 beamforming groups, three SBSs connected with MBS through backhaul transmission model [58], and five UEs presented in Figure 5a. We consider the non-line-of-sight path-loss model for urban MBSs and SBSs [59] and slow Rayleigh fading channels $h \sim \mathcal{CN}(0, 1)$. We followed the same system configuration as [29] for ensuring a fair comparison, tabulated in Table 2. All simulation results have been standardized by using the Z-score.

Table 2. Simulation Parameters.

Parameter	Value
Carrier frequency	2 GHz
Subchannel bandwidth	15 kHz
Number of subchannels	3
Number of subchannels per user	1
MBS antenna array size	100
MBS beamforming group size	20
The radius of the entire network	500 m
Number of SBS	2
Number of MDRU	1
Number of UE	5
SINR threshold of UE	1 for each UE
Rayleigh channel coefficient	$h \sim \mathcal{CN}(0, 1)$
Path loss model from MBS to SBS	$19.77 + 3.91 \times \log_{10} d_k$ in dB and d_k in km
Path loss model for SBS to UEs	$30.53 + 36.71 \times \log_{10} d_{k,u,t}$ in dB and $d_{k,u,t}$ in km at time t
Noise power spectral density	174 dBm/Hz
Maximum transmit power of SBS	24 dBm
Maximum cluster size	3
Transmit power of MBS	43 dBm
Operational power of SBS	0.5 W
Operational power of MBS	130W

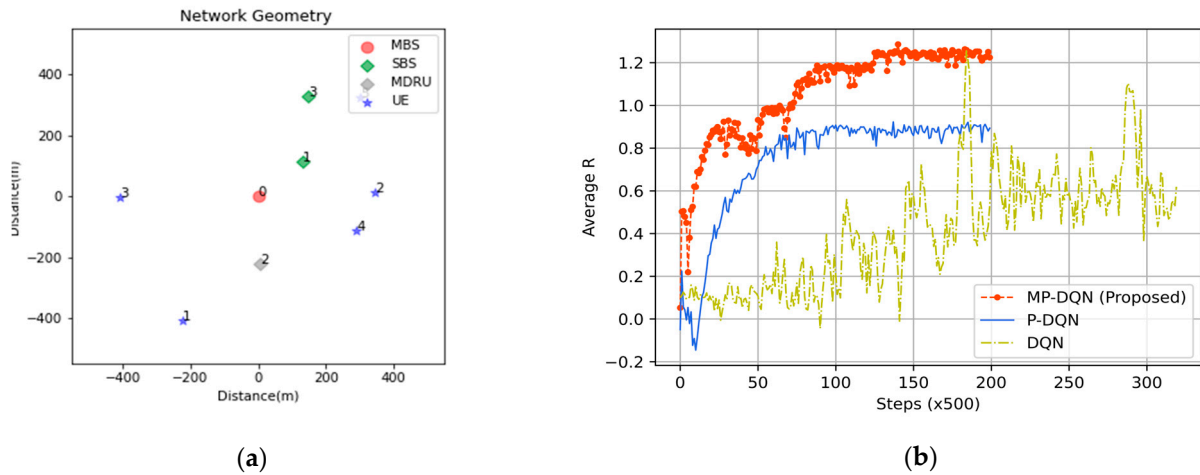


Figure 5. (a) Network geometry of HetNet based IoRT network and (b) Average reward (normalized) of proposed method (MP-DQN), P-DQN and DQN during the training session.

We compare the proposed MP-DQN presented in Algorithm 1 for the optimization problem, where UE association and emitting power allocation of SBSs/MDRUs have been considered jointly as the hybrid action space with two DRL based algorithms P-DQN and DQN. For DQN, continuous action space is converted into discrete by quantization process with $\frac{P_{SCm,max}}{10^{\mathcal{L}}}$, where \mathcal{L} is the discrete power levels ($\mathcal{L} = 5$ is considered in our simulation). In addition, the simulation results of the proposed method are compared to a well-known method called Nearest SBS/MDRU with Random Power. Each UE is connected to the nearby SC, which generates random power to serve every UE in its SC's cluster by fulfilling the conditions (1) the total power for all UE must be less or equal to the maximum power and (2) the total sum rate cannot exceed backhaul capacity of each SBS/MDRU. Furthermore, we consider the size of replay memory is 20,000, mini batch is 128, and the discount factor is 0.95 for all DRL algorithms. The total episodes for MP-DQN and P-DQN are 2000 while each episode has 50 timesteps. However, 3200 episodes are considered to simulate DQN. It takes more episodes to converge in hybrid action space-based optimization problem. In addition, other hyperparameters for MP-DQN, P-DQN, and DQN are tabulated in Table 3.

Algorithm 1: Multi pass DQN (MP-DQN) Algorithm.

Input: Probability distribution ξ , mini batch size B , exploration parameter ϵ , learning rates $\{\alpha_a, \alpha_{a,p}\}$.

Initialization: actor weights ω, ω^- and actor parameter weights (θ, θ^-)

For $t = 1, 2, 3, T$ **do**

Estimate the action parameters $z_j(s(t); \theta(t))$ by actor network

Choose the action $a(t) = (j, z_j)$ based on the ϵ greedy policy:

$$a(t) = \begin{cases} \text{random sample according to probability distribution } \xi, \text{ with } \epsilon \\ (j, z_j) : j = \operatorname{argmax}_{c \in A_d} Q(s(t), j, z_{\oplus j}; \omega), \text{ with } (1 - \epsilon) \end{cases}$$

Execute action $a(t)$, receive immediate reward $r(s(t), a(t))$ and next state $s(t+1)$

Save the experience $(s(t), a(t), r(t), s(t+1))$ into replay memory

Select mini batch size B randomly from the replay memory

Define the target $y(t)$ by

$$y(t) = r(t) + \gamma \max_{j' \in A_d} Q(s(t+1), j', z_{j'}(s(t+1); \theta^-); \omega^-)$$

Select the diagonal element from $\begin{pmatrix} Q_{11} & \cdots & Q_{1c} \\ \vdots & \ddots & \vdots \\ Q_{c1} & \cdots & Q_{cc} \end{pmatrix}$

Choose the best action j by argmax from diagonal elements

Use the $(y(t), s(t), a(t))$ to estimate the gradients $\nabla_{\omega} L^x(\omega)$ and $\nabla_{\theta} L^x(\theta)$

Update the weights parameters $\omega, \omega^-, \theta, \theta^-$

Table 3. The hyperparameter of MP-DQN.

Parameters	MP-DQN Q Network	MP-DQN Actor	P-DQN Q Network	P-DQN Actor	DQN
Learning rate	10^{-4}	10^{-5}	10^{-5}	10^{-5}	10^{-3}
Exploration	e-greedy	Ornstein-Uhlenbeck noise	e-greedy	Ornstein Uhlenbeck noise	e-greedy
Number of Outputs	$ A_d $	$U \cdot A_d $	$ A_d $	$U \cdot A_d $	$ A_d * \mathcal{L}^U$
Hidden layer	ReLu, 1024	ReLu, 1024	ReLu, 512	ReLu, 256	ReLu, 512
	ReLu, 512	ReLu, 512	ReLu, 128		ReLu, 128
	ReLu, 256	ReLu, 256			
	Relu 128	Relu 128	Relu 16		Relu 16
Number of Inputs	$U + U \cdot A_d $	U	$U + U \cdot A_d $	U	U

5.1. Simulation Results for Stationary UEs

We illustrate the average normalized results versus step over 500 realizations of proposed MP-DQN, P-DQN, and DQN algorithms during the training session. Figure 5b presents the average normalized reward of proposed MP-DQN, P-DQN and traditional DQN. Due to the complexity of the traditional DQN for discretization issues and the size of action space, the average reward is not properly converged. In P-DQN, the results are perfectly converged and saturated after-time steps. The final value of the average normalized reward is around 0.91. In comparison, the results of the proposed MP-DQN have converged perfectly but are a saturated bit later than P-DQN. The saturated value of the proposed method is around 1.25, which is clearly best compared with P-DQN and DQN algorithms.

We compare the average normalized test results of our proposed MP-DQN method with P-DQN, DQN, and Nearest SBS+ Random Power in Figure 6a,b and Figure 7 by considering the total time steps with 100 realizations. In Figure 6a, the average standardized test reward has been shown for all methods where maximum average results (around 1.26) for all timesteps are generated by our proposed method MP-DQN. The second height test reward is produced from P-DQN while the nearest SBS with random power for the UE method gives the worst results. In addition, the average normalized energy efficiency for test sessions has been depicted in Figure 6b for all discussed methods. The energy efficiency of our proposed method is approximately 9.89%, 94.7%, and 160.44% better than P-DQN, traditional DQN, and distance-based association methods, respectively during the whole test period. In addition, the average UE throughput by the normalized process has been illustrated in Figure 7 for all methods. According to Figure 7, the average normalized system throughput of our proposed method MP-DQN is approximately 4.27 which is 12.36%, 44.74%, and 19.607% better results compared to P-DQN, DQN, and the nearest distance with random power allocation algorithms. The summary of test results for all methods including the proposed method is presented in Table 4.

Table 4. The average normalized value of implemented methods.

Methods	Average Reward	Average Energy Efficiency	Average UE Throughput
MP-DQN (Proposed)	1.26	6.83	4.27
P-DQN	0.92	6.21	3.80
Nearest SBS+ Random Power	-0.75	2.62	3.57
DQN	0.118	3.51	2.95

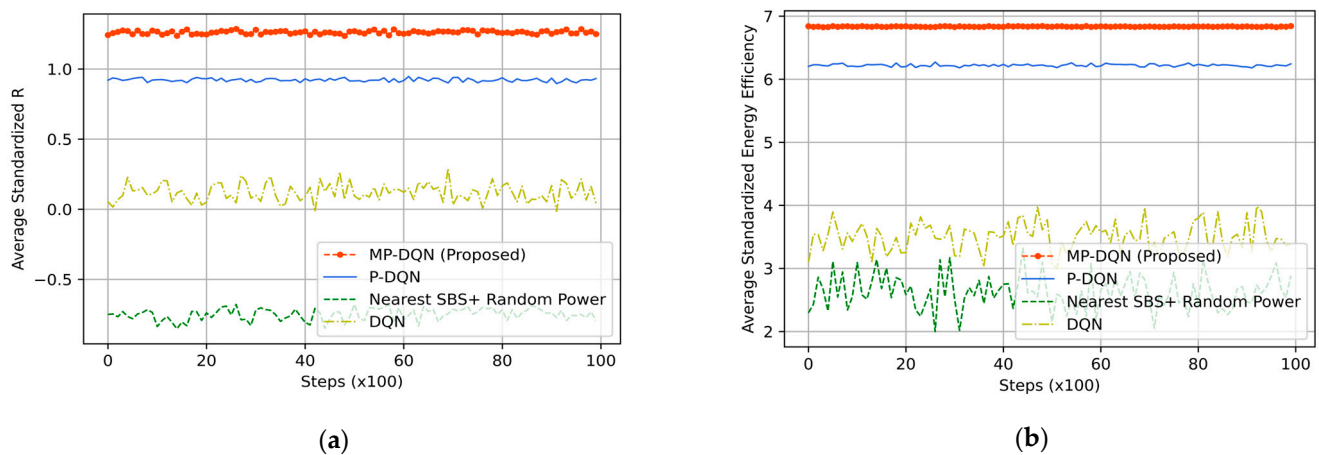


Figure 6. (a) Average reward (normalized) and (b) Average energy efficiency (normalized) of proposed method (MP-DQN), P-DQN, and DQN during the test session.

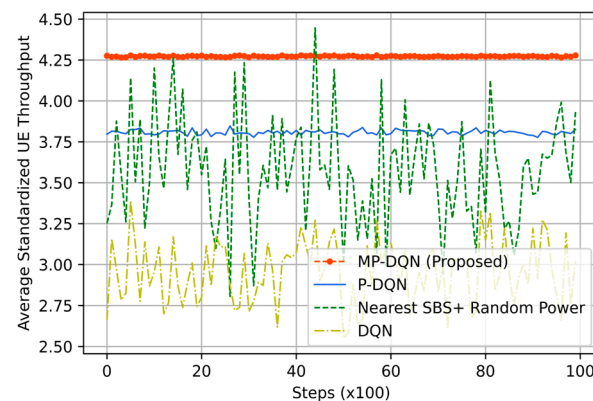
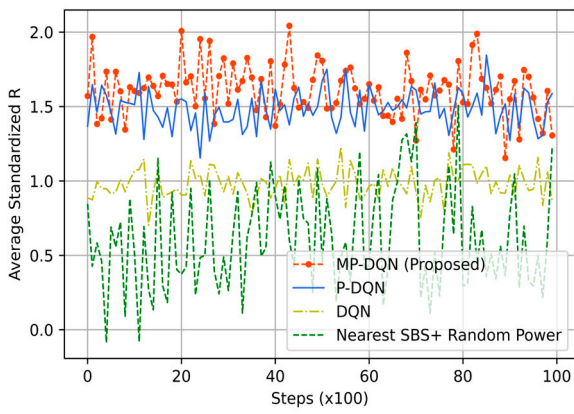


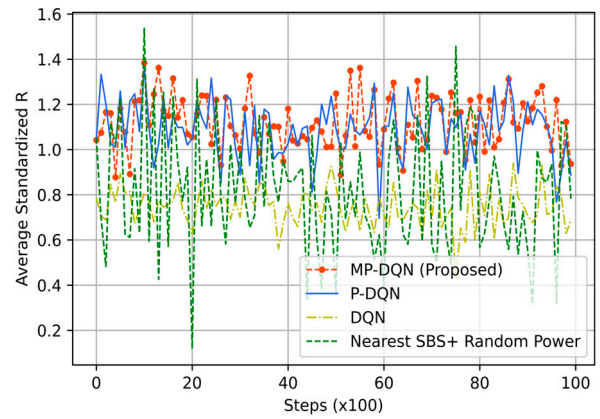
Figure 7. The average normalized system throughput proposed method (MP-DQN), P-DQN, and DQN during the test session.

5.2. Simulation Results Considering UE's Mobility

Due to its ability to remember previous activities, the GM mobility model is not stateless. It is appropriate for moveable UEs such as robots, cars, UGV, etc. We have illustrated average standardized reward, emergency efficiency, UE throughput, and SBS/MDRU throughput in Figure 8, Figure 9, Figure 10 and Figure 11, respectively. In addition, simulation results based on RFO and RFT have been illustrated in Figure Xa and Figure Xb, respectively, where X is within 8 to 11. Each UE's average velocity has been considered 10 km/h with a random direction. In Figure 8a, the average standardized rewards mean is 0.6, 0.98, 1.48, and 1.61 from Nearest SBS + random power, DQN, PDQN, and MPDQN, respectively, based on the RFO. When the simulation is run with the RFT, the MPDQN generates (mean) 1.12, while PDQN and DQN produce 1.08 and 0.74, respectively illustrated in Figure 8b. Average standardized emergency efficiencies for all algorithms have been illustrated in Figure 9a,b according to RFO and RFT, respectively. In Figure 9a, MPDQN gives 5.03 while PDQN, DQN, and Nearest SBS + random power produce 4.81, 3.52, and 3.13, respectively. Furthermore, the MPDQN and PDQN generate almost similar energy efficiency that is better than the DQN and Nearest SBS + random power illustrated in Figure 9b.

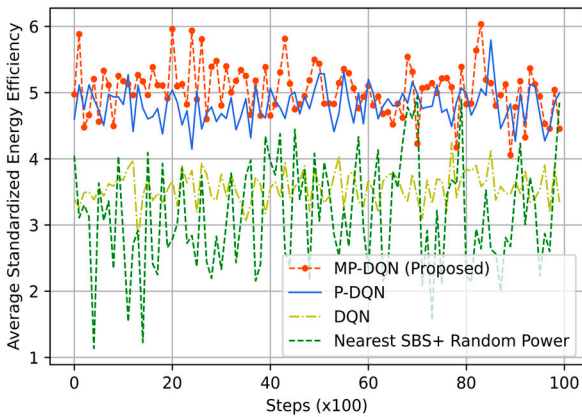


(a)

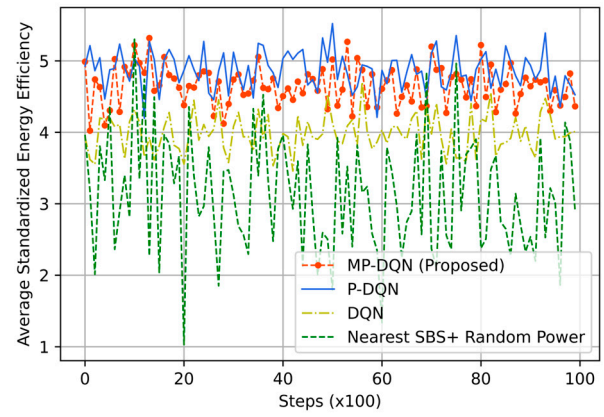


(b)

Figure 8. Average standardized reward based on (a) RFO and (b) RFT from proposed method (MP-DQN), P-DQN, and DQN and Nearest SBS + random power.

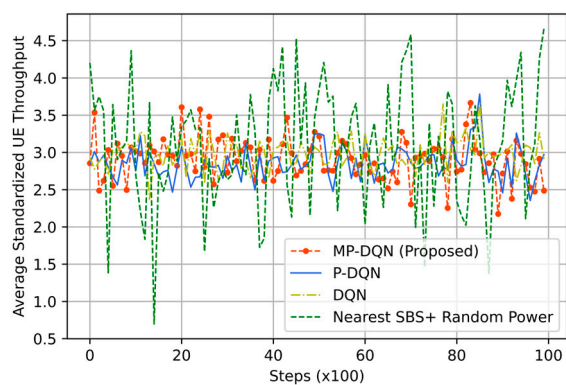


(a)

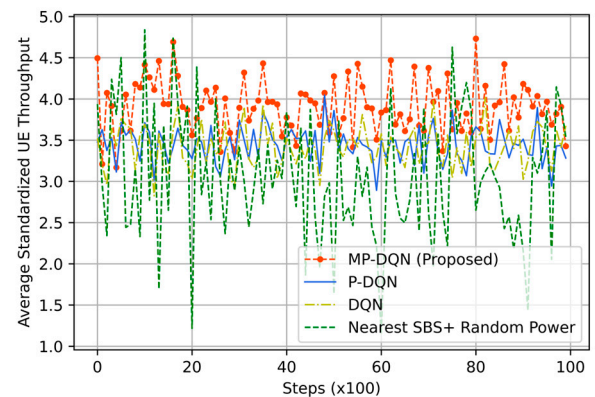


(b)

Figure 9. Average standardized energy efficiency based on (a) RFO and (b) RFT from proposed method (MP-DQN), P-DQN, and DQN and Nearest SBS + random power.



(a)



(b)

Figure 10. Average standardized UE throughput based on (a) RFO and (b) RFT from proposed method (MP-DQN), P-DQN, and DQN and Nearest SBS + random power.

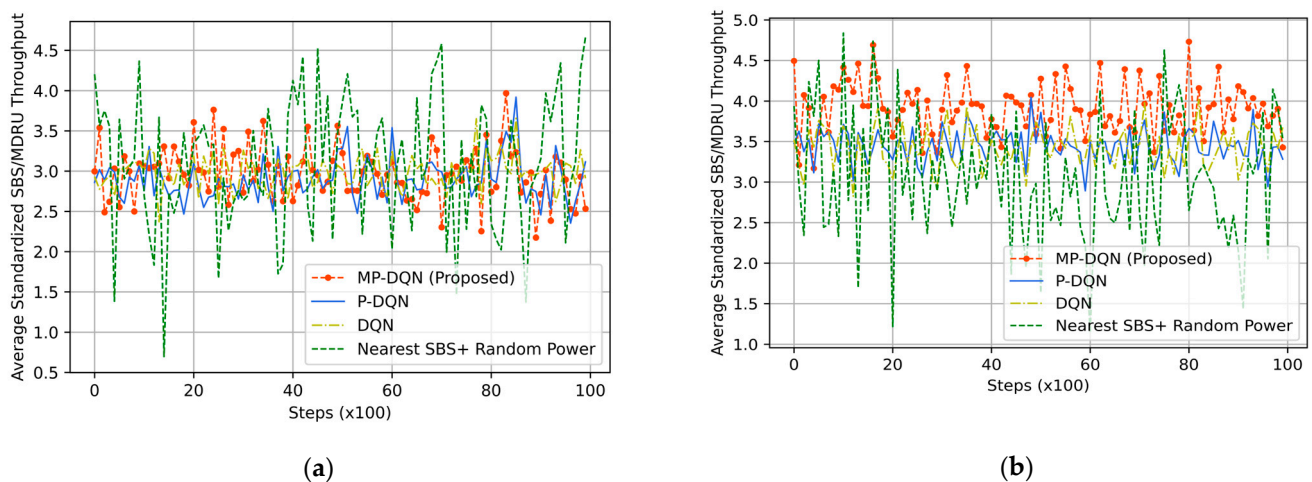


Figure 11. Average standardized SBS throughput based on (a) RFO and (b) RFT from proposed method (MP-DQN), P-DQN, and DQN and Nearest SBS + random power.

For evaluating the IoRT network, the QoS of UE is the crucial parameter that directly depends on the downlink throughput of UE in each time slot. In Figure 10, we have depicted the average standardized UE throughput. When we have utilized the RFO, the means of average standardized UE throughputs are 2.92, 2.85, and 2.97 from MPDQN, PDQN, and DQN, respectively. However, the Nearest SBS + Random power generates 3.08, as shown in Figure 10a. We have illustrated the simulation results using the RFT in Figure 10b. The mean of average standardized UE throughput is 3.05 (similar to Figure 10a) by Nearest SBS + Random power, while DRL-based algorithms generate better results. Hence, the design of an appropriate reward function is the key factor in DRL-based problem formulation. The proposed method (MPDQN) gives 3.91, which is the best UE throughput compared to PDQN (3.44) and DQN (3.40). Another key factor of two-tier HeNet is the backhaul connection from MBS to SBS/MDRU, which depends on the throughput of SBS/MDRU illustrated in Figure 11. The proposed method with the RFT outperforms others that are clearly shown in Figure 11.

In Figure 12, the mean of average standardized UE throughput has been presented concerning the velocity range from 10 Km/h to 60 Km/h. With RFO, the average standardized UE throughput means approximately 3.10, 2.95, 2.92, and 2.84 from Nearest SBS + random power, DQN, MPDQN, and PDQN, respectively for all velocities until 60 Km/hour. In contrast, those are around 3.10, 3.40, 3.44, and 3.93 for Nearest SBS + random power, DQN, PDQN, and MPDQN, respectively, when adopting the RFT. The results are varied for the Nearest SBS + random power method due to the random power allocations. In our simulations, the discrete action (user association) selects the SC, and the continuous action allocates the power from SBS according to user association in every time step. As a result, the increment of velocity does not impact the simulation results. It is shown that the proposed method with the proposed reward function RFT gives a better result compared to others (Nearest SBS + random power, DQN, and PDQN).

The proposed reward function RFT consists of three main factors of a two-tier IoRT network (i) the energy efficiency, (ii) the QoS of UE, and (iii) the QoS of SBS/MDRU, while the original reward function RFO mainly depends on the average standardized energy efficiency and throughput of UE. As a result, DRL algorithms with the proposed reward function produce better results in contrast to DRL algorithms with the original reward function. The proposed method (MP-DQN) performs better than other algorithms due to the solution of excessive parameterization of P-DQN. In summary, MP-DQN with the proposed reward function RFT outperforms PDQN, DQN, and Nearest SBS + random power in reward, average energy efficiency, average system throughput, and average SBS throughput for various velocities of UE.

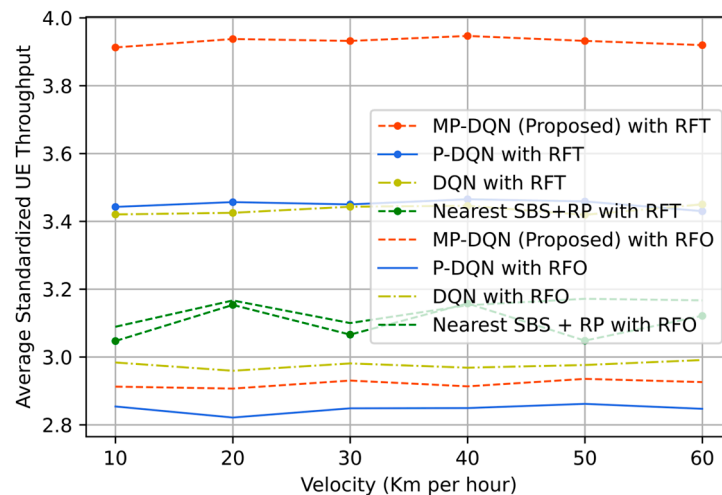


Figure 12. The mean of average standardized UE throughput based with proposed and old reward function from proposed method (MP-DQN), P-DQN, and DQN and Nearest SBS + random power.

6. Conclusions

For the PDM, authorities deploy various UE such as UGVs, UAVs, UUVs, health care robots, and smartphones via IoRT to collect information in affected areas, where wireless network, especially 4G/LTE/5G and beyond, works as a backbone. Few SBSs of HetNet can be damaged due to the disaster. Hence, the deployment of MDRU to replace malfunctioning SBS is well-established nowadays. In addition, the electric power crisis is a big challenge for PDM. Therefore, power optimization of HetNet by satisfying all UE demands has paid great attention to research. In this article, we have examined UE association and power allocation of SBS/MDRU to optimize the energy efficiency, UE throughput, and SC throughput of the downlink without knowing the environmental priori knowledge while taking into account the backhaul link and QoS guarantee for stationary and movable UE in MDRU aided two-tier HetNet, which are nonconvex, NP-hard, as well as a hybrid action space problem. We have proposed MP-DQN, which is model-free as well as hybrid action space-based DRL algorithm. The simulation results of the proposed method (MP-DQN) have been compared with two DRL-based algorithms (P-DQN and DQN) and the nearest distance-based SBS with random allocation power. During the whole test period considering the stationary UE, our suggested method's energy efficiency was around 9.89%, 94.7%, and 160.44% better than P-DQN, standard DQN, and distance-based association approaches, respectively. When the problem formulation by considering the modified Gauss–Markov UE mobility model has been investigated, we have proposed a new reward function RFT that is dependent on (i) the average standardized energy efficiency, (ii) the QoS of UE, and (iii) the QoS of SC; however, the original reward function RFO consists of average standardized energy efficiency and throughput of UE. Hence, DRL algorithms with the RFT are superior outcomes to those with the RFO. At various velocities, MP-DQN with the RFT outperforms PDQN, DQN, and Closest SBS + random power regarding reward, average energy efficiency, average system throughput, and average SC throughput.

Author Contributions: Conceptualization, H.K.; Investigation, H.K., M.-L.T. and Y.C.C.; Methodology, H.K., M.-L.T. and Y.C.C.; Resources, C.-O.C. and Y.O.; Writing—original draft, H.K., M.-L.T. and Y.C.C.; Software, H.K.; Funding acquisition, M.-L.T.; Writing—review and editing, H.K., C.-O.C. and Y.O. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Universiti Tunku Abdul Rahman (UTAR), Malaysia, under UTAR Research Fund (UTARRF) (IPSR/RMC/UTARRF/2021C1/T05). The ASEAN IVO (http://www.nict.go.jp/en/asean_ivo/index.html) project, “Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network”, was also involved in the production of the contents of this work.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lawry, L.; Burkle, F.M. Measuring the true human cost of natural disasters. *Disaster Med. Public Health Prep.* **2008**, *2*, 208–210. [[CrossRef](#)] [[PubMed](#)]
2. Shimada, G. The impact of climate-change-related disasters on Africa's economic growth, agriculture, and conflicts: Can humanitarian aid and food assistance offset the damage? *Int. J. Environ. Res. Public Health* **2022**, *19*, 467. [[CrossRef](#)]
3. Kamegawa, T.; Akiyama, T.; Sakai, S.; Fujii, K.; Une, K.; Ou, E.; Matsumura, Y.; Kishutani, T.; Nose, E.; Yoshizaki, Y.; et al. Development of a separable search-and-rescue robot composed of a mobile robot and a snake robot. *Adv. Robot.* **2020**, *34*, 132–139. [[CrossRef](#)]
4. Vera-Ortega, P.; Vázquez-Martín, R.; Fernandez-Lozano, J.J.; García-Cerezo, A.; Mandow, A. Enabling Remote Responder Bio-Signal Monitoring in a Cooperative Human–Robot Architecture for Search and Rescue. *Sensors* **2023**, *23*, 49. [[CrossRef](#)] [[PubMed](#)]
5. Paravisi, M.; Santos, D.H.; Jorge, V.; Heck, G.; Gonçalves, L.M.; Amory, A. Unmanned Surface Vehicle Simulator with Realistic Environmental Disturbances. *Sensors* **2019**, *19*, 1068. [[CrossRef](#)] [[PubMed](#)]
6. AlAli, Z.T.; Alabady, S.A. A survey of disaster management and SAR operations using sensors and supporting techniques. *Int. J. Disaster Risk Reduct.* **2022**, *82*, 103295. [[CrossRef](#)]
7. Lee, M.-F.R.; Chien, T.-W. Artificial intelligence and Internet of Things for robotic disaster response. In Proceedings of the 2020 International Conference on Advanced Robotics and Intelligent Systems (ARIS), Taipei, Taiwan, 19–21 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
8. Kamilaris, A.; Botteghi, N. The penetration of Internet of Things in robotics: Towards a web of robotic things. *J. Ambient. Intell. Smart Environ.* **2020**, *12*, 491–512. [[CrossRef](#)]
9. Villa, D.; Song, X.; Heim, M.; Li, L. Internet of Robotic Things: Current Technologies, Applications, Challenges and Future Directions. *arXiv* **2021**, arXiv:2101.06256.
10. Ray, P.P. Internet of robotic things: Concept, technologies, and challenges. *IEEE Access* **2016**, *4*, 9489–9500. [[CrossRef](#)]
11. Vermesan, O.; Bahr, R.; Ottella, M.; Serrano, M.; Karlsen, T.; Wahlstrøm, T.; Sand, H.E.; Ashwathnarayan, M.; Gamba, M.T. Internet of Robotic Things Intelligent Connectivity and Platforms. *Front. Robot. AI* **2020**, *7*, 104. [[CrossRef](#)]
12. Rengaraju, P.; Sethuramalingam, K.; Lung, C.H. Providing internet access for post-disaster communications using balloon networks. In Proceedings of the 18th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor & Ubiquitous Networks, Alicante, Spain, 22–26 November 2021; pp. 111–117.
13. Panda, K.G.; Das, S.; Sen, D.; Arif, W. Design and Deployment of UAV-Aided Post-Disaster Emergency Network. *IEEE Access* **2019**, *7*, 102985–102999. [[CrossRef](#)]
14. Sakano, T.; Fadlullah, Z.M.; Ngo, T.; Nishiyama, H.; Nakazawa, M.; Adachi, F.; Kato, N.; Takahara, A.; Kumagai, T.; Kasahara, H.; et al. Disaster-resilient networking: A new vision based on movable and deployable resource units. *IEEE Netw.* **2013**, *27*, 40–46. [[CrossRef](#)]
15. Zeng, Y.; Zhang, R.; Lim, T.J. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Commun. Mag.* **2016**, *54*, 36–42. [[CrossRef](#)]
16. Sakano, T.; Kotabe, S.; Komukai, T.; Kumagai, T.; Shimizu, Y.; Takahara, A.; Ngo, T.; Md Fadlullah, Z.; Nishiyama, H.; Kato, N. Bringing movable and deployable networks to disaster areas: Development and field test of MDRU. *IEEE Netw.* **2016**, *30*, 86–91. [[CrossRef](#)]
17. Matraccia, M.; Saeed, N.; Kishk, M.A.; Alouini, M.-S. Post-Disaster Communications: Enabling Technologies, Architectures, and Open Challenges. *IEEE Open J. Commun. Soc.* **2022**, *3*, 1177–1205. [[CrossRef](#)]
18. Wang, J.; Sato, K.; Guo, S.; Chen, W.; Wu, J. Big Data Processing With Minimal Delay and Guaranteed Data Resolution in Disaster Areas. *IEEE Trans. Veh. Technol.* **2018**, *68*, 3833–3842. [[CrossRef](#)]
19. Porte, J.; Briones, A.; Maso, J.M.; Pares, C.; Zaballos, A.; Pijoan, J.L. Heterogeneous wireless IoT architecture for natural disaster monitorization. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 184. [[CrossRef](#)]
20. Wang, Y. Models and Algorithms for Efficient Data Processing in Fog Computing Supported Disaster Areas. Ph.D. Dissertation, University of Aizu, Aizuwakamatsu, Japan, 2019.
21. Wang, X.; Jiang, F.; Zhong, L.; Ji, Y.; Yamada, S.; Takano, K.; Xue, G. Intelligent Post-Disaster Networking by Exploiting Crowd Big Data. *IEEE Netw.* **2020**, *34*, 49–55. [[CrossRef](#)]
22. Xu, K.; Qu, Y.; Yang, K. A tutorial on the internet of things: From a heterogeneous network integration perspective. *IEEE Netw.* **2016**, *30*, 102–108. [[CrossRef](#)]
23. Kabir, H.; Tham, M.-L.; Chang, Y.C. Twin Delayed DDPG based Dynamic Power Allocation for Mobility in IoRT. *J. Commun. Softw. Syst.* **2023**, *19*, 19–29. [[CrossRef](#)]

24. Kabir, H.; Tham, M.-L.; Chang, Y.C. DRL based Energy-Efficient Radio Resource Allocation Algorithm in Internet of Robotic Things. In Proceedings of the 2022 IEEE Symposium on Wireless Technology & Applications (ISWTA), Kuala Lumpur, Malaysia, 17–18 August 2022; pp. 104–109.
25. Nguyen, H.T.; Nguyen, M.T.; Do, H.T.; Hua, H.T.; Nguyen, C.V. DRL-based intelligent resource allocation for diverse QoS in 5G and toward 6G vehicular networks: A comprehensive survey. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 5051328. [[CrossRef](#)]
26. Abbasi, M.; Shahraki, A.; Piran, J.; Taherkordi, A. Deep Reinforcement Learning for QoS provisioning at the MAC layer: A Survey. *Eng. Appl. Artif. Intell.* **2021**, *102*, 104234. [[CrossRef](#)]
27. Xiong, Z.; Zhang, Y.; Niyato, D.; Deng, R.; Wang, P.; Wang, L.-C. Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges. *IEEE Veh. Technol. Mag.* **2019**, *14*, 44–52. [[CrossRef](#)]
28. Zhu, J.; Wu, F.; Zhao, J. An Overview of the Action Space for Deep Reinforcement Learning. In Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence, Sanya, China, 22–24 December 2021; pp. 1–10. [[CrossRef](#)]
29. Hsieh, C.-K.; Chan, K.-L.; Chien, F.-T. Energy-Efficient Power Allocation and User Association in Heterogeneous Networks with Deep Reinforcement Learning. *Appl. Sci.* **2021**, *11*, 4135. [[CrossRef](#)]
30. Bester, C.J.; James, S.D.; Konidaris, G.D. Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces. *arXiv* **2019**, arXiv:1905.04388.
31. Omstedt, F. A deep reinforcement learning approach to the problem of golf using an agent limited by human data. In *Degree Project in Computer Science and Engineering*; Kth Royal Institute of Technology: Stockholm, Sweden, 2020.
32. Wen, G.; Wu, K. Building decision tree for imbalanced classification via deep reinforcement learning. *Proc. Mach. Learn. Res.* **2021**, *157*, 1645–1659.
33. Bouktif, S.; Cheniki, A.; Ouni, A. Traffic Signal Control Using Hybrid Action Space Deep Reinforcement Learning. *Sensors* **2021**, *21*, 2302. [[CrossRef](#)]
34. Zhang, X.; Jin, S.; Wang, C.; Zhu, X.; Tomizuka, M. Learning insertion primitives with discrete-continuous hybrid action space for robotic assembly tasks. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 9881–9887. [[CrossRef](#)]
35. Yan, Y.; Du, K.; Wang, L.; Niu, H.; Wen, X. MP-DQN Based Task Scheduling for RAN QoS Fluctuation Minimizing in Public Clouds. In Proceedings of the 2022 IEEE International Conference on Communications Workshops, ICC Workshops, Seoul, Republic of Korea, 16–20 May 2022; pp. 878–884. [[CrossRef](#)]
36. Guo, L.; Jia, J.; Chen, J.; Du, A.; Wang, X. Joint Task Offloading and Resource Allocation in STAR-RIS assisted NOMA System. In Proceedings of the 2022 IEEE 96th Vehicular Technology Conference, VTC2022-Fall, London, UK, 26–29 September 2022; pp. 1–5. [[CrossRef](#)]
37. Shimizu, Y.; Suzuki, Y.; Sasazawa, R.; Kawamoto, Y.; Nishiyama, H.; Kato, N.; Yamamoto, A.; Kotabe, S. Development of Movable and Deployable ICT Resource Unit (MDRU) and its Overseas Activities. *J. Disaster Res.* **2019**, *14*, 363–374. [[CrossRef](#)]
38. Khan, A.; Mukhtar, M.; Ullah, F.; Bilal, M.; Kwak, K.-S. EA-RDSP: Energy Aware Rapidly Deployable Wireless Ad hoc System for Post Disaster Management. *Comput. Mater. Contin.* **2021**, *69*, 1725–1746. [[CrossRef](#)]
39. Zhou, H.; Wang, X.; Umehira, M.; Chen, X.; Wu, C.; Ji, Y. Deep reinforcement learning based access control for disaster response networks. In Proceedings of the GLOBECOM 2020—2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [[CrossRef](#)]
40. Ngo, T.; Nishiyama, H.; Kato, N.; Sakano, T.; Takahara, A. A Spectrum- and Energy-Efficient Scheme for Improving the Utilization of MDRU-Based Disaster Resilient Networks. *IEEE Trans. Veh. Technol.* **2014**, *63*, 2027–2037. [[CrossRef](#)]
41. Wang, H.; Zhao, H.; Wu, W.; Xiong, J.; Ma, D.; Wei, J. Deployment algorithms of flying base stations: 5G and beyond with UAVs. *IEEE Internet Things J.* **2019**, *6*, 10009–10027. [[CrossRef](#)]
42. Xu, Z.; Wang, Y.; Tang, J.; Wang, J.; Gursoy, M.C. A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
43. Zhao, N.; Liang, Y.-C.; Niyato, D.; Pei, Y.; Jiang, Y. Deep reinforcement learning for user association and resource allocation in heterogeneous networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. [[CrossRef](#)]
44. Zhao, N.; Liang, Y.-C.; Niyato, D.; Pei, Y.; Wu, M.; Jiang, Y. Deep Reinforcement Learning for User Association and Resource Allocation in Heterogeneous Cellular Networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 5141–5152. [[CrossRef](#)]
45. Li, Z.; Wen, X.; Lu, Z.; Jing, W. A General DRL-based Optimization Framework of User Association and Power Control for HetNet. In Proceedings of the 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Helsinki, Finland, 13–16 September 2021; pp. 1141–1147.
46. Li, Z.; Wen, X.; Lu, Z.; Jing, W. A DDPG-based Transfer Learning Optimization Framework form User Association and Power Control in HetNet. In Proceedings of the 2022 IEEE International Conference on Communications Workshops (ICC 750 Workshops), Seoul, Republic of Korea, 16–20 May 2022; pp. 343–348.
47. Narottama, B.; Shin, S.Y. Dynamic power allocation for non-orthogonal multiple access with user mobility. In Proceedings of the 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 17–19 October 2019; pp. 0442–0446. [[CrossRef](#)]

48. Masaracchia, A.; Nguyen, V.-L.; Nguyen, M. The impact of user mobility into non-orthogonal multiple access (NOMA) transmission systems. *EAI Endorsed Trans. Ind. Netw. Intell. Syst.* **2020**, *7*, e5. [[CrossRef](#)]
49. Neely, M.J.; Modiano, E.; Rohrs, C.E. Dynamic power allocation and routing for time varying wireless networks. In Proceedings of the IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies 756 (IEEE Cat. No. 03CH37428), San Francisco, CA, USA, 30 March–3 April 2003; Volume 1, pp. 745–755.
50. Wang, Y.; Meyer, M.C.; Wang, J. Base Station Allocation for Users with Overlapping Coverage in Wirelessly Networked Disaster Areas. In Proceedings of the 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCOM/CyberSciTech), Fukuoka, Japan, 5–8 August 2019; pp. 954–960. [[CrossRef](#)]
51. Zhang, B.; Liu, S.; Yu, J.-L.; Han, Z. A Learning Aided Long-Term User Association Scheme for Ultra-Dense Networks. *IEEE Trans. Veh. Technol.* **2021**, *71*, 820–830. [[CrossRef](#)]
52. Zhou, H.; Wang, X.; Umehira, M.; Chen, X.; Wu, C.; Ji, Y. Wireless Access Control in Edge-Aided Disaster Response: A Deep Reinforcement Learning-Based Approach. *IEEE Access* **2021**, *9*, 46600–46611. [[CrossRef](#)]
53. Bai, F.; Helmy, A. A survey of mobility models. In *Wireless Ad hoc Networks*; University of Southern California: Los Angeles, CA, USA, 2004; Volume 206, p. 147.
54. Hausknecht, M.; Stone, P. Deep reinforcement learning in parameterized action space. *arXiv* **2015**, arXiv:1511.04143.
55. Masson, W.; Ranchod, P.; Konidaris, G. Reinforcement learning with parameterized actions. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30. No. 1.
56. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Found. Trends@Mach. Learn.* **2018**, *11*, 219–354. [[CrossRef](#)]
57. Xiong, J.; Wang, Q.; Yang, Z.; Sun, P.; Han, L.; Zheng, Y.; Fu, H.; Zhang, T.; Liu, J.; Liu, H. Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv* **2018**, arXiv:1810.06394.
58. Wang, N.; Hossain, E.; Bhargava, V.K. Joint Downlink Cell Association and Bandwidth Allocation for Wireless Backhauling in Two-Tier HetNets with Large-Scale Antenna Arrays. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 3251–3268. [[CrossRef](#)]
59. 3rd Generation Partnership Project (3GPP). *Further Advancements for E-UTRA Physical Layer Aspects (Release 9)*; 3rd Generation Partnership Project (3GPP): Sofia Technology Park, France, 2016.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.



Malaysia

**PROPOSED INCLUSION OF NEW USE CASE TO APT REPORT ON
LOCAL-AREA RESILIENT INFORMATION SHARING AND COMMUNICATION
SYSTEMS**

In the Expert Group on Disaster Risk Management and Relief System (EG DRMR) of ASTAP Working Group Network and System (WG NS), an ATP report, entitled “APT Report on local-area resilient information sharing and communication systems,” has been discussing since ASTAP-33, June 2021. In this report, editorial efforts are being made by NICT. In the latest draft shown below, some systems including “NerveNet” have been described.

ASTAP-34/TMP-22

Draft of APT Report on local-area resilient information sharing and communication systems

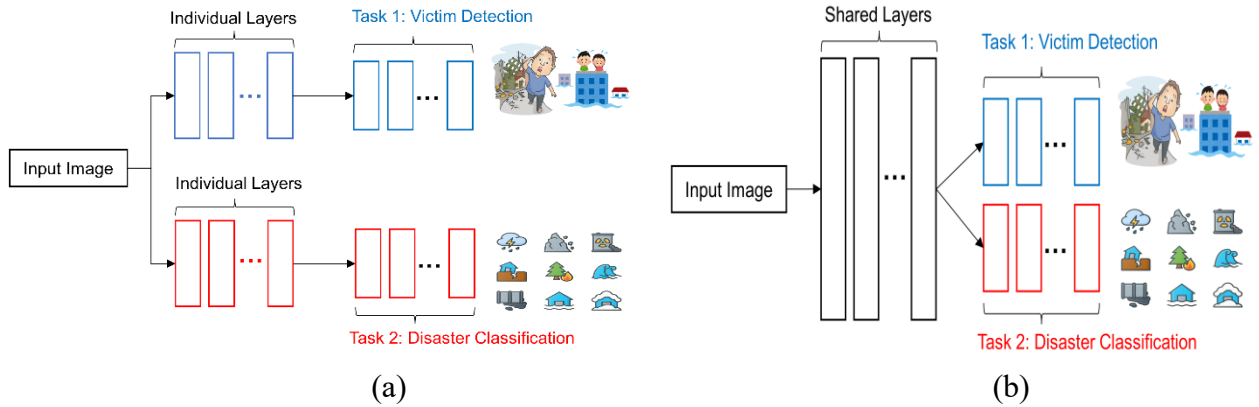
https://www.apr.int/sites/default/files/2022/04/ASTAP-34-TMP-22_LocalAreaResilientSystem_EG_DRMRS-20220421.docx

Universiti Tunku Abdul Rahman (UTAR), Malaysia has collaborated with NICT in using NerveNet for disaster monitoring. Thus, Malaysia would like to include a new use case of NerveNet to Section 5.1.3.4 of the APT report.

Proposed text:

5.1.3.4. Disaster Monitoring using Artificial Intelligence (AI) and NerveNet

When disaster events happen, an efficient rescue operation requires the detected disaster type and number of victims. A straightforward approach would be deploying two single-task AI models that perform the disaster classification and victim detection separately, as shown in Figure 5-6(a). Such approach is ill-suited for IoT applications due to high memory footprint and computing power. A better solution would be using a multi-task learning (MTL) model, as displayed in Figure 5-6(b). The advantages of using the MTL model are listed in Figure 5-6(c).



- (c)
- ✓ LESSER memory requirements
 - ✓ BETTER disaster classification accuracy
 - ✓ SAME victim detection performance
 - ✓ FASTER inference speed

Figure 5-6: (a) Conventional AI model. (b) Multi-task learning AI model. (c) Advantages of using multi-task learning model.

Figure 5-7 shows the implementation of MTL model in a low-powered IoT device (Raspberry Pi).



Figure 5-7: Raspberry Pi (a) Front View. (b) Rear View.

Figures 5-8 shows the inference output of the MTL model at different areas.



Figure 5-8: (a) Flood. (b) Earthquake. The joint disaster classification and victim count prediction are labeled at the top left corner of the input images.

Considering that communications infrastructure may be destroyed during disaster, resilient networking is attained by implementing the NerveNet testbed as shown in Figure 5-9.

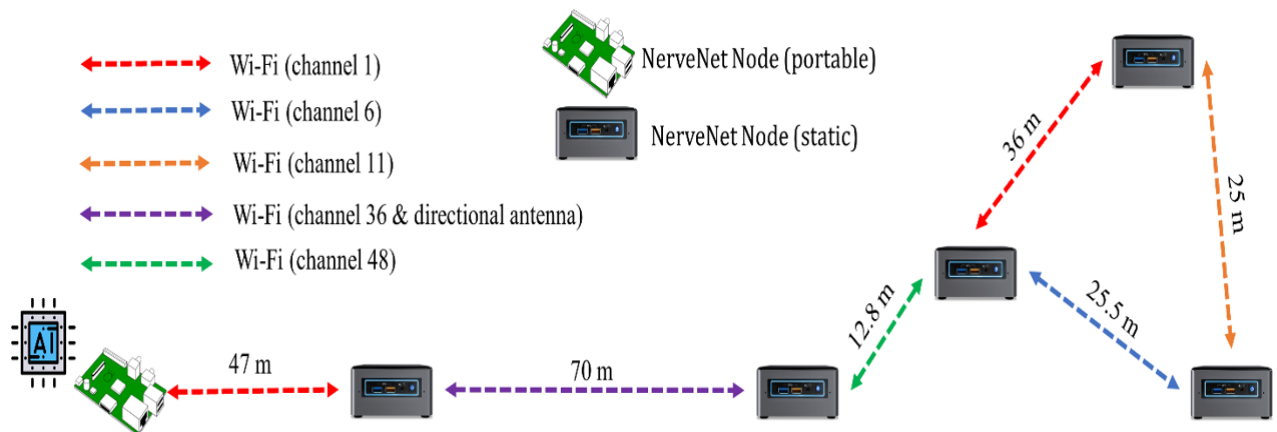


Figure 5-9: NerveNet testbed.

Figure 5-10 displays the working flow of the disaster monitoring solution. Once the detection result for every particular video frame is obtained, rolling average prediction is applied to reduce prediction flickering.

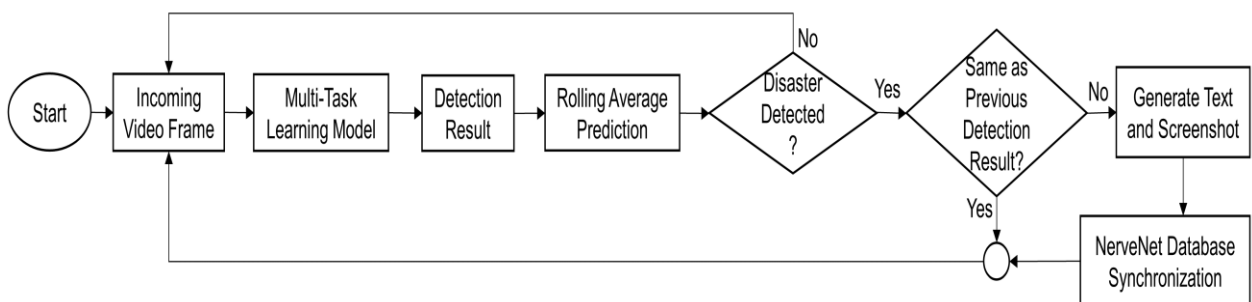


Figure 5-10: Working flow of disaster monitoring.