Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

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

YI JIE WONG[1], MAU-LUEN THAM[1], BAN-HOE KWAN[2], EZRA MORRIS ABRAHAM GNANAMUTHU[1], 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, Selangor 43000, Malaysia
[2]Department of Mechatronics and Biomedical Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Sungai Long Campus, Selangor 43000, Malaysia
[3]Resilient ICT Research Center, National Institute of Information and Communications Technology (NICT), Tokyo, 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 titled "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

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 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 [8-9] 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 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 [13,16,27-29] and victim detection [18–21, 31–33] 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 [34-38], 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 [23–26] 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 & 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 CrisisMMD was released. To facilitate benchmarking purpose, the authors in [16] consolidated the aforementioned datasets into a dataset called Crisis Image Benchmarks Dataset (CrisisIBD), 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 investigate 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, 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, sematic 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], [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 [18–21, 31–33] 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 [34–35]. 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) & 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 [8–9] do not use production-ready tools for FL implementation.

D. INFERENCE OPTIMIZATION

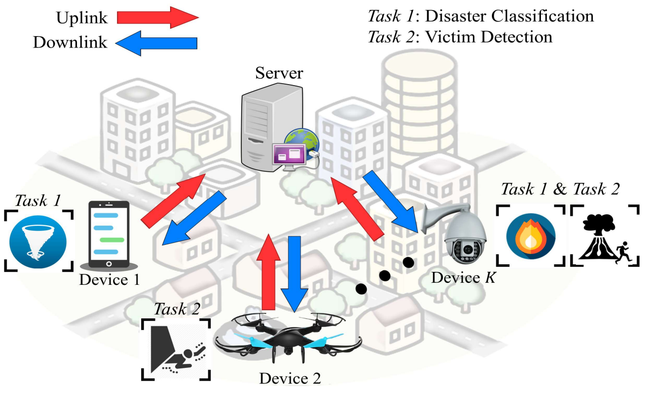
Efficient execution of a CNN model is undoubtedly another important criterion for implementing production-ready DL solutions. This is especially true for deploying heterogeneous IoT devices of different hardware constraints. How to enable fast inference on low-powered embedded platforms remains an open 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.

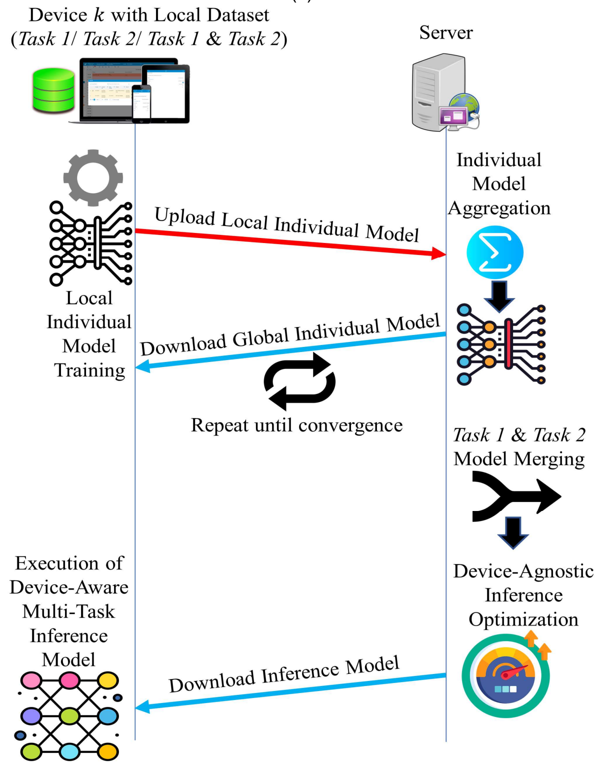
III. PROPOSED APPROACH

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. Secondly, the local model weights are transmitted to the



(a)



(b)

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

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.

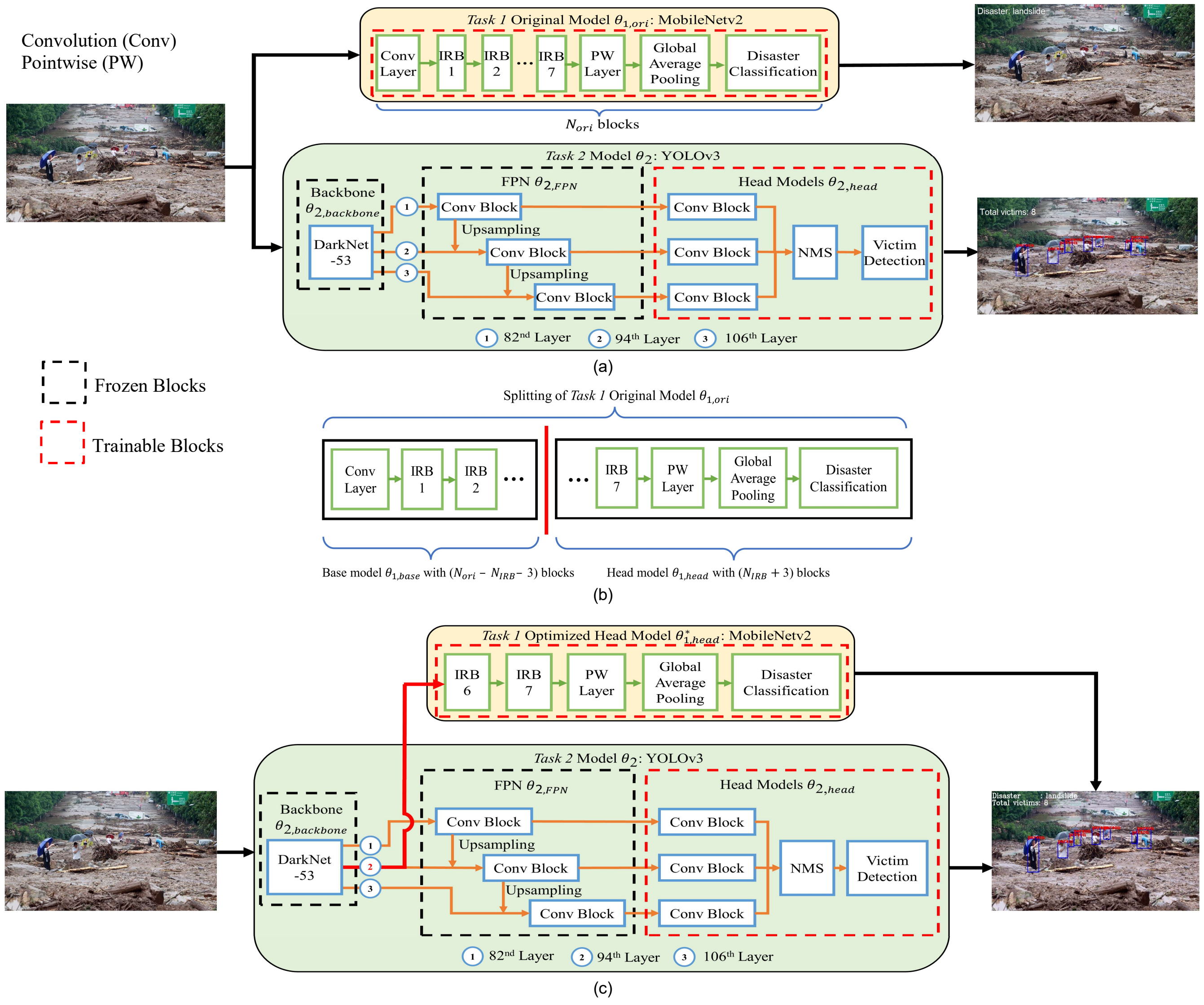
A. MTL MODEL

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


Convolution (Conv)
Pointwise (PW)
Task 1 Original Model $\theta_{1,ori}$: MobileNetv2
Conv Layer
IRB 1
IRB 2
IRB 7
PW Layer
Global Average Pooling
Disaster Classification
N_{ori} blocks
Disaster: landslide
Task 2 Model θ_2: YOLOv3
Backbone $\theta_{2,backbone}$
DarkNet -53
FPN $\theta_{2,FPN}$
Conv Block
Upsampling
Head Models $\theta_{2,head}$
NMS
Victim Detection
Total victims: 8
82nd Layer
94th Layer
106th Layer
(a)
Frozen Blocks
Trainable Blocks
Splitting of Task 1 Original Model $\theta_{1,ori}$
Base model $\theta_{1,base}$ with (N_{ori} − N_{IRB}− 3) blocks
Head model $\theta_{1,head}$ with (N_{IRB} + 3) blocks
(b)
Task 1 Optimized Head Model $\theta^*_{1,head}$: MobileNetv2
IRB 6
Disaster : landslide
Total victims: 8
(c)
FIGURE 2. **Network architecture of *Task 1* and *Task 2*. (a) Conventional model. (b) *Task 1* splitting. (c) Proposed model.**

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.

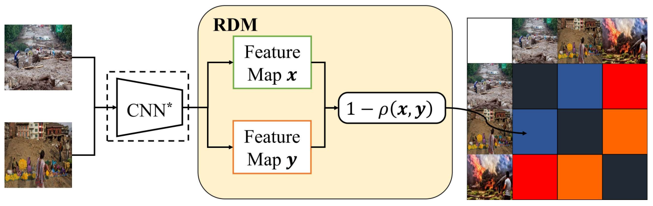
Firstly, a subset of images is randomly selected from CrisisIBD as the conditions for dissimilarity computation. We can acquire the representation or feature map of each image at any layers of a CNN by forward passing the image until the target layer. The dissimilarity score of a pair of images can be 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:

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

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

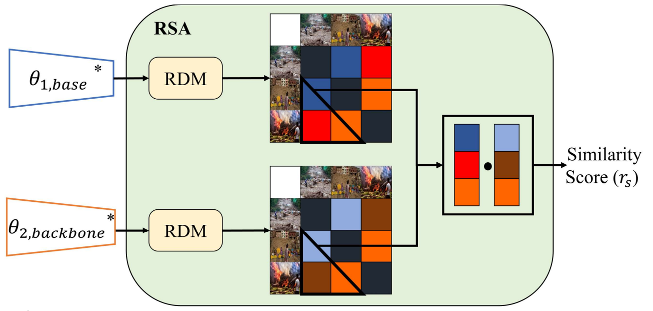
Secondly, the similarity between the RDMs of two CNNs can be computed with the Spearman's correlation (r_s) between the upper or lower triangular part of the RDMs, as shown in (2):

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


RDM
CNN*
Feature Map $\boldsymbol{x}$
Feature Map $\boldsymbol{y}$
$1 - \rho(\boldsymbol{x},\boldsymbol{y})$

*CNN will be $\theta_{1,base}$ (where N_{IRB}=1, 2, 3) and $\theta_{2,backbone}$ (82nd, 94th and 106th Layer)

(a)



*Repeat with all combinations of $\theta_{1,base}$ (where N_{IRB}=1, 2, 3) and $\theta_{2,backbone}$ (82nd, 94th and 106th Layer)

(b)

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

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* 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

Algorithm 1 Training Strategy for *Task 1*

```
Input:   Labeled Dataset, $\mathcal{L}$
         Number of Epoch, $\mathcal{N}_{t1}$
         Learning rate, $\alpha$
         Task 1 Head Model, $\theta^*_{1,head}$
         Categorical Cross-Entropy Loss Function, $\mathcal{J}_{CCE}$
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, $\boldsymbol{f}$
07         $f \leftarrow \theta_{2,backbone}(l)$
08         // Compute gradients and update model
09         $\nabla \leftarrow \frac{\delta}{\delta x} \mathcal{J}_{CCE}(\theta^*_{1,head}, f)$
10         $\theta^{*\,t+1}_{1,head} \leftarrow \theta^{*\,t}_{1,head} - \alpha \nabla$
11     end for
12 end for
```

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, $\alpha$
         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         $\nabla \leftarrow \frac{\delta}{\delta x} \mathcal{J}_{Y3}(\theta^t_{2,head}, l)$
10         $\nabla_{accumulate} \leftarrow \nabla_{accumulate} + \nabla$
11         // Update model
12         if $c$ mod $\mathcal{B} = 0$ then
13             $\nabla_{accumulate} \leftarrow \nabla_{accumulate} / \mathcal{B}$
14             $\theta^{t+1}_{2,head} \leftarrow \theta^t_{2,head} - \alpha \nabla$
15             $\nabla_{accumulate} \leftarrow 0$
16         end if
17         // Increase $\alpha$ after 10 epochs of warm up training
18         if $t$ mod $10 = 0$ then
19             $\alpha \leftarrow \alpha \times 10$
20         end if
21     end for
22 end for
```

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

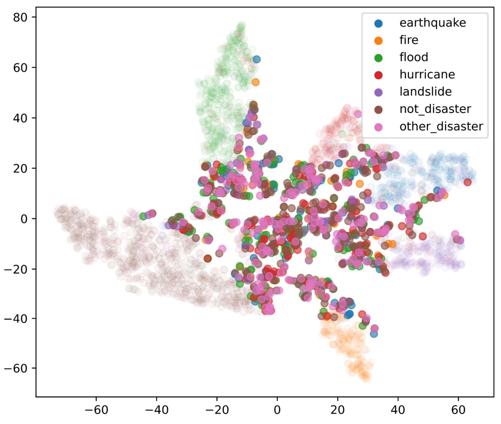
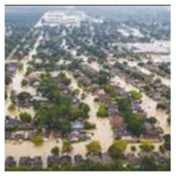


FIGURE 4. t-SNE results for the test dataset. Bold color corresponds to 33 % of hard samples.

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, $\mathcal{N}_a$
          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 $\theta^{*0}_{1.head} \leftarrow$ train $\theta^*_{1.head}$ for 5 epochs using $\mathcal{L}_0$
04 for $t = 1: \mathcal{N}_a$ 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     $\mathcal{Q}_t^{easy} \leftarrow \{\mathcal{U}_{t,i} \mid \underset{unc}{i} \in \text{argbtmK}(\mathcal{Q}, \mathcal{K}_{easy})\}$; Pick the
       last corresponding $\mathcal{K}_{easy}$ samples from $\mathcal{Q}$
09     $\mathcal{Q}_t^{mod} \leftarrow \{\mathcal{U}_{t,i} \mid \underset{unc}{i} \in \text{argmidK}(\mathcal{Q}, \mathcal{K}_{mod})\}$; Pick the
       corresponding $(1 + |\mathcal{U}_t|/2 \ to\ \mathcal{K}_{mod} + |\mathcal{U}_t|/2)$
       samples from $\mathcal{Q}$
10     $\mathcal{Q}_t^{hard} \leftarrow \{\mathcal{U}_{t,i} \mid \underset{unc}{i} \in \text{argtopK}(\mathcal{Q}, \mathcal{K}_{hard})\}$; 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     $\mathcal{Y}_t \leftarrow annotate\ \mathcal{Q}_t$
14     $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1} \cup \{(\mathcal{X}, \mathcal{Y}) \mid \mathcal{X} \in \mathcal{Q}_t,\ \mathcal{Y} \in \mathcal{Y}_t\}$
15     // Phase 4: Update Model
16     $\theta^{*t+1}_{1.head} \leftarrow$ fine-tuning $\theta^{*t}_{1.head}$ 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 $\mathcal{L}_{t+1}$
```

C. FEDERATED LEARNING (FL)

To minimize the effort of implementation, we choose the simple Federated Averaging (FedAvg) algorithm as in [52-54]. 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 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 $\mathcal{N}_e$. The entire FL framework 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

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, $\mathcal{N}_c$
          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, $\theta^{global}$
4  for $t = 1: \mathcal{N}_c$ do
5      Operations on the server side:
6      // Select a fraction of Clients, C
7      $m \leftarrow max\ (C \cdot K\ ,\ 1)$
8      $S_t \leftarrow \{random\ set\ of\ m\ clients\}$
9      // Train each selected client, $\theta^k$
10     for each client $k \in S_t$ in parallel do
11         $\theta^{global}_{t+1} \leftarrow$ ClientUpdate($\theta^{global}_t$)
12     end for
13     $\theta^{global}_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} \theta^k_{t+1}$
14 end for
15 ClientUpdate($\theta^{global}$):
16     // Train the client model using local dataset
17     $\theta \leftarrow \theta^{global}$
18     update $\theta$ using any preferred strategy
19     return $\theta$
```

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 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.

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

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.

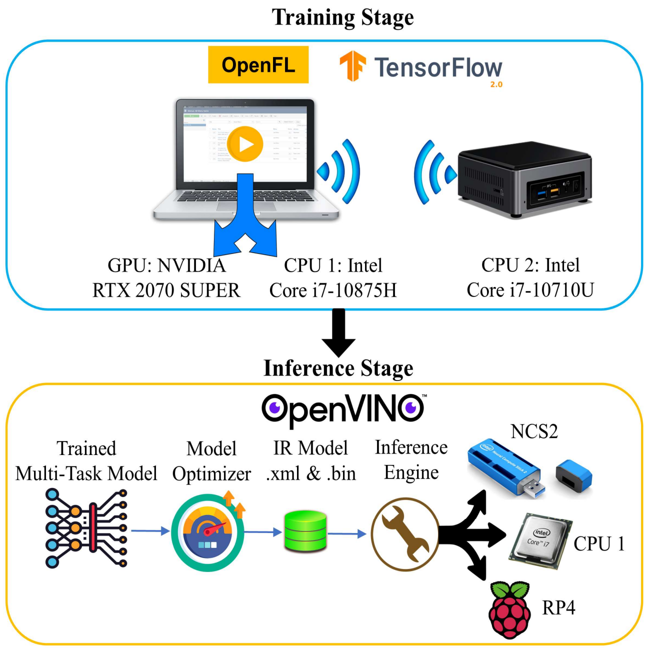


FIGURE 5. Experimental setup.

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.

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

	Parameters	Values
Task 1	Max AL round, $\mathcal{N}_a$	32
	AL Seed, $\lvert\mathcal{L}_0\rvert$	700
	Size of AL Labeled Dataset in Each Client, $\lvert\mathcal{L}_t\rvert$	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 x 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 x 10^{-3}
	Size of Local Dataset in Each Client (2-Client Setup)	2997
	Size of Local Dataset in Each Client (3-Client Setup)	1998

D. RESULTS AND DISCUSSIONS

1) RSA SIMILARITY

Firstly, we validate our hypothesis that the optimized head model $\theta^*_{1,hea}$ 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.

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.

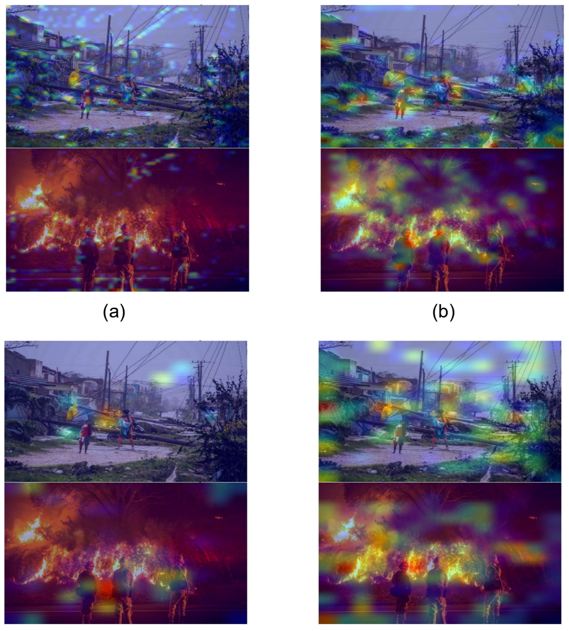


FIGURE 6. Grad-CAM visualization of activation maps. (a) $\theta_{2,backbone}$ (82nd Layer). (b) $\theta_{2,backbone}$ (94th Layer). (c) $\theta_{2,backbone}$ (106th Layer). (d) $\theta_{1,base}$ ($N_{IRB} = 2$).

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

$\theta_{2,\text{backbone}}$ / $\theta_{1,base}$	82nd Layer	94th Layer	106th 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 $\theta_{2,\text{backbone}}$ after retraining.

$\theta_{2,\text{backbone}}$ / $\theta_{1,head}$	82nd Layer	94th Layer	106th 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

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

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

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 1.64% and 1.04% in F1 score, 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, 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 4.31% 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 35%. These results again highlight the importance of sharing model weights for better performance.

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

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 6.44 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

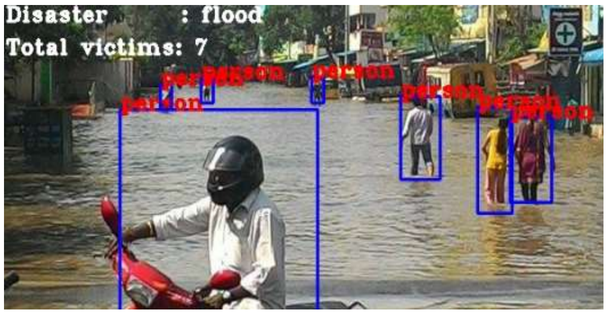
TABLE 10. **Network model size comparison.**

	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

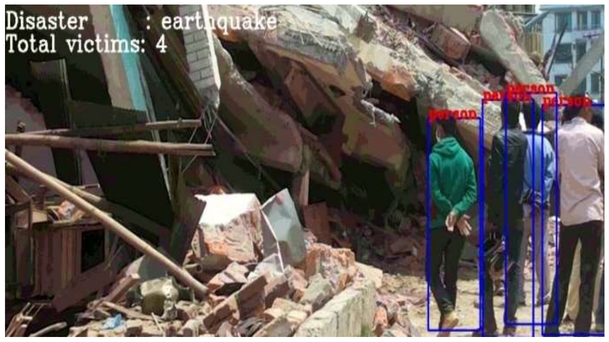
stemming from plugging the NCS2 into less powerful RP4 further drops to 1.8. However, this is acceptable since NCS2 consumes power of only 1.5 W [64], which is important in establishing sustainable IoT solutions. To reveal more insight, we also convert the models in conventional approach into two separate OpenVINO models. A sequential execution of these two models on RP4 results in another FPS slowdown of 28%.

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 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.

Fig. 7 shows some examples of inference output of the multi-task model.



(a)



(b)

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.

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

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

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 28% 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

This work is the output of the ASEAN IVO (http://www.nict.go.jp/en/asean_ivo/index.html) project titled "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).

REFERENCES

[1] S. Evans, "Claims paid for Japan's M7 quake in Feb 2021 nearing $900m," *Artemis.bm*, 2021. https://www.artemis.bm/news/claims-paid-for-japans-m7-quake-in-feb-2021-nearing-900m/ (accessed Oct. 14, 2021).

[2] United Nations Office for the Coordination of Humanitarian Affair, "Five essentials for the first 72 hours of disaster response," *OCHA*, 2017. https://www.unocha.org/story/five-essentials-first-72-hours-disaster-response (accessed Aug. 20, 2021).

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Sep. 2014, doi: 10.48550/arxiv.1409.1556.

[4] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, Accessed: Aug. 22, 2021. [Online]. Available: https://arxiv.org/abs/1704.04861.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Jun. 2015, Accessed: Aug. 16, 2021. [Online]. Available: https://arxiv.org/abs/1506.02640.

[6] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *European Conf. on Computer Vis.*, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[7] A. Saeed, F. D. Salim, T. Ozcelebi, 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.

[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.

[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.

[10] Intel, "Intel® Distribution of OpenVINO^{TM} Toolkit," *intel.com*. https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html (accessed Mar. 15, 2022).

[11] A. Demidovskij *et al.*, "OpenVINO Deep Learning Workbench: A Platform for Model Optimization, Analysis and Deployment," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, vol. 2020-November, pp. 661–668, Nov. 2020, doi: 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 *2021 IEEE 12th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, 2021, pp. 407–412, doi: 10.1109/UEMCON53757.2021.9666576.

[13] H. Mouzannar, Y. Rizk, and M. Awad, "Damage Identification in Social Media Posts Using Multimodal Deep Learning," *Proc. Int. ISCRAM Conf.*, vol. 2018-May, no. May, pp. 529–543, 2018.

[14] F. Alam, F. Ofli, and M. Imran, "CrisisMMD: Multimodal twitter datasets from natural disasters," in *12th Int. AAAI Conf. on Web and Social Media, ICWSM 2018*, 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. on World Wide Web*, 2014, pp. 159–162, doi: 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 *2020 IEEE/ACM Int. Conf. on Advances in Social Netw. Analysis and Mining (ASONAM)*, 2020, pp. 151–158, doi: 10.1109/ASONAM49781.2020.9381294.

[17] F. Alam, T. Alam, F. Ofli, and M. Imran, "Social Media Images Classification Models for Real-time Disaster Response," *CoRR*, vol. abs/2104.0, 2021, [Online]. Available: https://arxiv.org/abs/2104.04184.

[18] D. R. Hartawan, T. W. Purboyo, and C. Setianingsih, "Disaster victims detection system using convolutional neural network (CNN) method," *Proc. - 2019 IEEE Int. Conf. Ind. 4.0, Artif. Intell. Commun. Technol. IAICT 2019*, pp. 105–111, Jul. 2019, doi: 10.1109/ICIAICT.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," *2020 Int. Symp. Community-Centric Syst. CcS 2020*, Sep. 2020, doi: 10.1109/CCS49175.2020.9231433.

[20] F. B. Jaradat and D. Valles, "A Victims Detection Approach for Burning Building Sites Using Convolutional Neural Networks," *2020 10th Annu. Comput. Commun. Work. Conf. CCWC 2020*, pp. 280–286, Jan. 2020, doi: 10.1109/CCWC47524.2020.9031275.

[21] Naveen K, Lokesh Kumar N, Kumaresh PM, Mallikarjun SC, and Prachetha K, "Early Flood Detection and Disaster Victim Detection," *Int. J. Sci. Technol. Eng.*, vol. 7, no. 1, pp. 11–17, Aug. 2020, Accessed: Jul. 05, 2022. [Online]. Available: http://ijste.org/Article.php?manuscript=IJSTEV7I1003.

[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.

[23] M. L. Tham and W. K. Tan, "IoT Based License Plate Recognition System Using Deep Learning and OpenVINO," *ACM Int. Conf. Proceeding Ser.*, pp. 7–14, Oct. 2021, doi: 10.1145/3502814.3502816.

[24] E. Izutov, "Fast and Accurate Person Re-Identification with RMNet," *CoRR*, vol. abs/1812.0, 2018, [Online]. Available: http://arxiv.org/abs/1812.02465.

[25] D. Brown, "Mobile Attendance based on Face Detection and Recognition using OpenVINO," *Proc. - Int. Conf. Artif. Intell. Smart Syst. ICAIS 2021*, pp. 1152–1157, Mar. 2021, doi: 10.1109/ICAIS50930.2021.9395836.

[26] S. Bernabe, C. Gonzalez, A. Fernandez, and U. Bhangale, "Portability and Acceleration of Deep Learning Inferences to Detect Rapid Earthquake Damage from VHR Remote Sensing Images Using Intel OpenVINO Toolkit," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 14, pp. 6906–6915, 2021, doi: 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 *2017 IEEE/ACM Int. Conf. on Advances in Social Netw. Analysis and Mining (ASONAM)*, 2017, pp. 569–576.

[28] A. Sharma and U. Verma, "Flood Magnitude Assessment from UAV Aerial Videos Based on Image Segmentation and Similarity," *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2021-December, pp. 476–481, 2021, doi: 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.

[30] M. Hong and R. Akerkar, "Victim detection platform in IoT paradigm," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 3, pp. 1–14, Feb. 2021, doi: 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.

[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.

[33] C. Dorn, A. Depold, F. Lurz, S. Erhardt, and A. Hagelauer, "UAV-based Localization of Mobile Phones for Search and Rescue Applications," in *IEEE Annu. Conf. on Wireless and Microw. Technol. (WAMICON), 2022,* pp. 1–4, Jun. 2022, doi: 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, 2020, doi: 10.1109/TITS.2019.2943777.

[35] C. Wen *et al.*, "Multi-scene citrus detection based on multi-task deep learning network," in *2020 IEEE Int. Conf. on Systems, Man, and Cybern. (SMC)*, 2020, pp. 912–919, doi: 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, 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. (Ny).*, vol. 432, pp. 559–571, 2018, doi: 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 *2019 IEEE/CVF Conf. on Computer Vis. and Pattern Recog. (CVPR)*, 2019, pp. 4979–4988, doi: 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," Feb. 2022, doi: 10.48550/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, doi: 10.48550/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.*, 2021, doi: 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.

[43] G. Wang, J. N. Hwang, C. Rose, and F. Wallace, "Uncertainty sampling based active learning with diversity constraint by sparse selection," *2017 IEEE 19th Int. Work. Multimed. Signal Process. MMSP 2017*, vol. 2017-January, pp. 1–6, Nov. 2017, doi: 10.1109/MMSP.2017.8122269.

[44] Z. Xiong *et al.*, "Facing small and biased data dilemma in drug discovery with federated learning," *bioRxiv*, p. 2020.03.19.998898, Jan. 2020, doi: 10.1101/2020.03.19.998898.

[45] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *CoRR*, vol. abs/1706.0, 2017, [Online]. Available: http://arxiv.org/abs/1706.05098.

[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conf. on Computer Vis. and Pattern Recog.*, 2018, pp. 4510–4520, doi: 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," *Aquac. Eng.*, vol. 91, p. 102117, 2020, doi: 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," *2020 IEEE Int. Conf. Consum. Electron. - Asia, ICCE-Asia 2020*, Nov. 2020, doi: 10.1109/ICCE-ASIA49877.2020.9277040.

[49] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul. 2022, doi: 10.48550/arxiv.2207.02696.

[50] K. Dwivedi and G. Roig, "Representation similarity analysis for efficient task taxonomy & transfer learning," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 12379–12388, Jun. 2019, doi: 10.1109/CVPR.2019.01267.

[51] N. Grimova and M. Macas, "Query-By-Committee Framework Used for Semi-Automatic Sleep Stages Classification," *Proc. 2019, Vol. 31, Page 80*, vol. 31, no. 1, p. 80, Nov. 2019, doi: 10.3390/PROCEEDINGS2019031080.

[52] S. Wang *et al.*, "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.

[53] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," Jun. 2018, doi: 10.48550/arxiv.1806.00582.

[54] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated Optimization in Heterogeneous Networks," Dec. 2018, doi: 10.48550/arxiv.1812.06127.

[55] G. A. Reina *et al.*, "OpenFL: An open-source framework for Federated Learning," *CoRR*, vol. abs/2105.0, 2021, [Online]. Available: https://arxiv.org/abs/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," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11383 LNCS, pp. 92–104, 2019, doi: 10.1007/978-3-030-11723-8_9.

[57] M. J. Sheller *et al.*, "Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data," *Sci. Reports 2020 101*, vol. 10, no. 1, pp. 1–12, Jul. 2020, doi: 10.1038/s41598-020-69250-1.

[58] M. Hamzah, "Auto-Annotate: Automatically annotate your entire image directory by a single command," *GitHub repository*. Github, 2020, [Online]. Available: https://github.com/mdhmz1/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," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-October, pp. 618–626, Dec. 2017, doi: 10.1109/ICCV.2017.74.

[60] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, doi: 10.48550/arxiv.1905.11946.

[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, p. 103306, Sep. 2022, doi: 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.

[63] M. Asad, A. Moustafa, and T. Ito, "Federated Learning Versus Classical Machine Learning: A Convergence Comparison," Jul. 2021, doi: 10.48550/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 on Accelerated Mach. Learning (AccML)*, pp. 1–15, 2020.

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

MAU-LUEN THAM received his Bachelor of Engineering and Doctor of Philosophy in the field of Telecommunication Engineering from University of Malaya. He is currently an Assistant Professor with Universiti Tunku Abdul Rahman. His research interests include IoT, machine learning/deep learning/deep reinforcement learning and beyond-5G communications. He has been a principal investigator (PI) and co-investigator of more than 15 research and development projects. This includes 5 international grants, two of which are simultaneously led by him as the PI/Co-PI under the support of ASEAN IVO and British Council. He has published 2 IEEE Transactions papers as a principal author.

BAN-HOE KWAN has obtained his degree for Bachelor of Engineering (Electrical), Master of Engineering Science and PhD in Engineering from 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, Internet of Things and robotics.

MORRIS EZRA received his B. Eng degree from Bharathiar University, India, his M.E degree from Anna University, India and his Ph.D degree from Multimedia University, Malaysia. He joined Karunya Institute of Technology as a lecturer in 1993, before moving to Malaysia in 1998. In 2008 he joined as assistant professor with University Tunku Abdul Rahman (UTAR), Kuala Lumpur Malaysia and became an associate professor in 2014. His research area includes digital signal processing, wireless adhoc networks, optimization using PSO, GA/IGA and mobile communication. He has published over 40 papers in international journals and conferences.

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 system 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, FY2019 the Commendation for Science and Technology by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.