Effective Evacuation Route Strategy for Emergency Vehicles

Myint Myint Sein
University of Computer Studies, Yangon (UCSY), Myanmar
myint@ucsy.edu.mm

K-zin Phyo
University of Computer Studies, Yangon (UCSY), Myanmar
kzinphyo@ucsy.edu.mm

Mau Luen Tham
Universiti Tunku Abdul Rahman, UTAR, Malaysia
thamml@utar.edu.my

Yasunori Owada
NICT, Miyagi, Japan
yowada@nict.go.jp

Nordin Bin Ramli
MIMOS, Malaysia
nordin.ramli@mimos.my

Suvit Poomrittigul
PIT, Thailand
suvit@pit.ac.th

2021 IEEE 10th Global Conference on Consumer Electronics (GCCE) | 978-1-6654-3676-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GCCE53005.2021.9621935

Abstract— **Rapid response and recovery efforts are critical to emergency and disaster management. It is important that rescue teams need to arrive at incident location within a short time to minimize the risk and damage. The effective evacuation route estimating algorithm is developed for complicated unstructured road network based on the Dijkstra algorithm. The proposed system will estimate the optimal evacuation route for emergency services and vehicles such as fire service vehicle, ambulance and police car. It is also provided to search the near relief area and guiding to move the safe place through the optimal evacuation route assessment. The nearest emergency service is estimated by extracting surround services system. The haversine distance is applied for measuring between two locations. The optimal route is computed from service center to the incident location after estimating the nearest emergency services. Yangon's complex road network is selected to implement the proposed web-based application system.**

Keywords: Rapid response, Disaster management, Evacuation route, Advanced Dijkstra algorithm, Emergency Services and Vehicles.

I. Introduction

Emergencies can occur at any time without notice or message. Emergencies may be caused by accidents, fires, explosions and natural disaster. An emergency is a terrible situation that can endanger people and the environment and requires an immediate response from emergency services and rescue teams. In many developing countries, the rate of emergency damage is still high due to lack of emergency respond scheme and unstructured road grids. Because of disaster and unfinished reconstruction, some roads are damaged, and some roads are impassable. It is a critical task of medical service centers, fire stations and emergency rescue teams, to move the injured people to the hospitals within a short time. Weak road network infrastructure makes it difficult for emergency vehicles to reach the incident place quickly. To address this situation, an effective evacuation route strategy for emergency vehicles is proposed to get to the incident location and destination quickly. Yangon road network is selected for implement the proposed system. The location of three emergency services, fire stations, hospitals and police stations in Yangon region are collected and stored in database. Inadditing, the location of road, street and road condition are added in database.

There are some researches related to optimal part finding algorithm [1-5]. The proposed evacuation route findinding system is extened based on previous research works [1-3]. The geospecial parth optimization algorithem for hospital was implemented for a case study of Allahabad city[4]. S.Sivakumaret al.[5] proposed the Modified Dijkstra's shortest path algorithm with multiple features such as cost, time and congestion. Most traditional route search methods are often used as the criteria for choosing the shortest route with the travel distance.

In this study, a unique optimization algorithm based on Dijkstra's algorithm has been upgraded, including the addition of certain parameters and conditional statement of roads to find the effective evacuation routes and better implementation of the safest and convenient routes. Yangon's road network is used to evaluate the proposed system of intricate, narrow, one-ended roads, a complex road network. The optimal route finding strategy is improved by using the web technology for emergency vehiles such as fire truck, ampulance and police car when the accident occurs on unstructure road network.

This research is a part of the " Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network", ASEAN IVO project. One of the goals of this project is to find the rapid route or optimal route identification on complex road network for fire vehicle.

II. Generating the Road Network Data Base

The proposed optimal route detection system for the emergency vehicles is primarily to save lives and properties and to pinpoint the exact location of emergencies.

A. *Data Collection for Tested Area*

The downtown area of Yangon's road network is selected for the study area of the proposed system. It is one of the largest cities in Myanmar and which is built with complicated unstructured road like a narrow road and a closed road. The location data (latitude, longitude) of 85 hospitals and clinics, 41 fire force stations and 50 police stations are collected for emergency services.

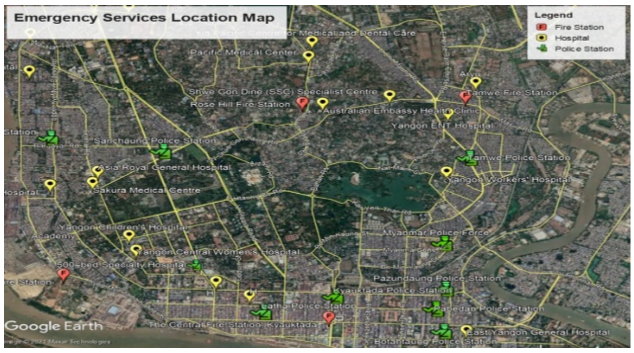

Emergency Services Location Map
Legend
Fire Station
Hospital
Police Station
Pacific Medical Center
Shwe Gon Dine (SSC) Specialist Centre
Rose Hill Fire Station
Australian Embassy Health Clinic
Tamwe Fire Station
Yangon ENT Hospital
Sanchaung Police Station
Tamwe Police Station
Asia Royal General Hospital
Sakura Medical Centre
Yangon Workers' Hospital
Yangon Children's Hospital
Myanmar Police Force
Yangon Central Women's Hospital
500-bed Specialty Hospital
Pazundaung Police Station
Kyauktada Police Station
Latha Police Station
Pabedan Police Station
The Central Fire Station Kyauktada
East Yangon General Hospital
Botahtaung Police Station
Google Earth

Fig. 1. Emergency Services Location Map.

Figure 1 is illustrated the location map of the emergency services in Yangon. All location of roads, narrow streets and road conditions are also added in created database.

B. *Quantum Geographical Information System(QGIS)*

The creating the vector map of Yangon Road network, data analyzing and editing the special data information are performed by QGIS. It is incorporated with other open-source packages as GRASS, PostGIS, Map and GIS server. Yangon's road network is made up of 100,000 edges and 35,000 nodes.

III. Proposed Optimal Route Finding System

The aim is to reduce the level of damage by searching the near emergency services from the incident place and minimizing delays along the route to an emergency location.

A. *Overview of the Proposed System*

The overview of the proposed web application structure is shown in Fig. 2. The system collects the address or street name as emergency location information to determine the exact location of the incident. The system can then check the location with special coordinates on Google-Maps. The close emergency services are estimated by the Haversine measurement.

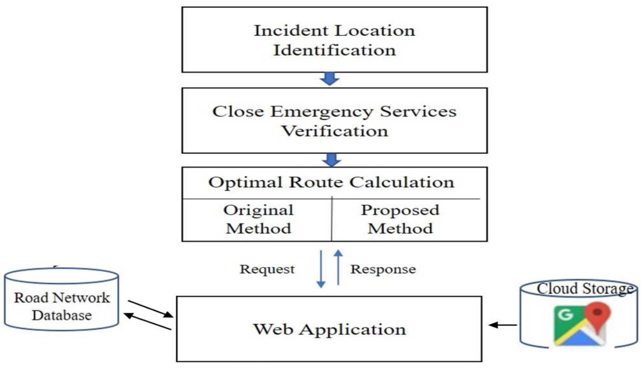

Incident Location Identification
Close Emergency Services Verification
Optimal Route Calculation
Original Method
Proposed Method
Request
Response
Road Network Database
Web Application
Cloud Storage

Fig. 2. Overview of the proposed System.

B. *Computing the Optimal Route*

After estimating the nearest emergency services, the optimal route from services center to the incident location is calculated. Source place is defined for emergency service and destination place is for incident place. For comparison result, optimal route is computed for both of ordinary method and proposed method. The proposed modified Dijkstra algorithm (pseodo code) is described as follow:

```
function MDijkstra(V, s)
    d[s] ←0
    target ← t
    for  all v ∈ V
        do  d[v] ←infinity
    end for
    S←Ø
    H_Queue←V

    while H_Queue ≠Ø
       u ← ExtractMin (H_Queue,d)
       S←S∪{u}
        if u==target
            break;
        end if
       for all v ∈ neighbors[u] and status(u,v) != 1
           if   d[v] > d[u] + w(u, v)
                then     d[v] ←d[u] + w(u, v)

         end if
        end for
        end while
    return d[destination]
```

IV. Experiments and Results

In experiment, the optimal evacuation route for fire vehicle is computed in original and modified Dijkstra method. The fire event is in KMS kyar street, Tamwe township. The result of incident location verification and estimated nearest emergency services are shown in Fig 3(a) and Fig. 3(b), respectively. The optimal route computing results for original and proposed approach are illustrated in Fig 3(b) and Fig 3(b), respectively. The computation time for number of nodes in each operation are shown in Table1.

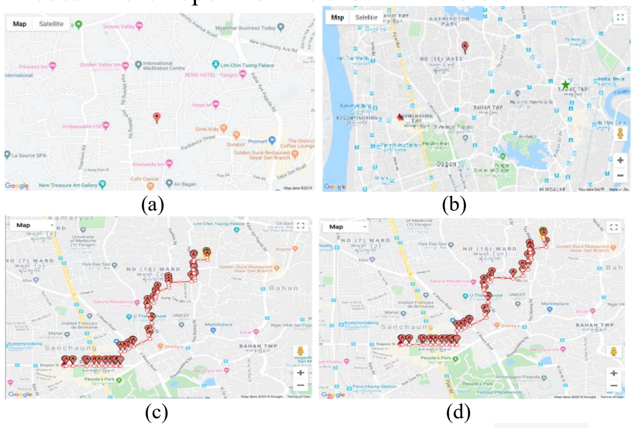


Fig. 3. The Optimal Route between the Fire Dept. and KMS kyar street.

Table 1 Evaluation of Runtime Complexity Based on No. of Nodes.

Run Time	10 nodes	100 nodes	1000 nodes	10000 nodes	100000 nodes
O (1)	1	1	1	1	1
O (n)	10	100	1000	10000	100000
O (log n)	3	7	10	13	17

V. Conclusion

The estimating of effective emergency route strategy is proposed for complexed road network of Yangon. The proposed work will help emergency rescue teams to reach the incident location in a short time save the lives and properties. It will deploy a nearby victim area and will carry the best rescue routes to evacuate people from dangerous areas. On integrating with real time road traffic condition obtained by IOT sensor will be considered to improve this proposed approach.

Acknowledgment

This publication 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). We thank our project members NICT-Japan, UCSY-Myanmar, PIT-Thailand, UTAR and MIMO Malaysia.

References

[1] K-zin Phyo and Myint Myint Sein ,"Investigation of Optimum Rescue Itinerary by Using Advanced Routing Method", IEEE GCCE Conference, Nara, Japan, October 4-13,2018,pp. 521-522.

[2] K-zin Phyo and Myint Myint Sein, "Optimal Path Finding for Emergency Cases on Android," Annual International Conference on Mobile Systems, Singapore, 23rd - 25th June, 2016, pp.71.

[3] Yutaka Ohsawa, Htoo Htoo, Naw Jacklin Nyunt; and Myint Myint Sein, "Generalized Bichromatic Homogeneous Vicinity Query Algorithm in Road Network Distance, Japan International Conference, IPSJ, Kyoto 2015, Mach 17~19.

[4] N. Kumar, M. Kumar and S. Kumarsrivastva," Geospatial Path optimization for Hospital: a case study of Allahabad city, Uttar Pradesh", Internation Journal of Modern Engineering Research, Vol. 4 ,Issue.10, 2014, pp.9-14.

[5] S. Sivakumar and C.Chandrasekar ,"Modified Dijkstra's Shortest Path Algorithm", International Journal of Innovative Research in Computer and Communication Engineering, 11- 2014, Vol. 2, pp.6450-6456.

Joint Disaster Classification and Victim Detection using Multi-Task Learning

2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) | 978-1-6654-0690-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/UEMCON53757.2021.9666576

Mau-Luen Tham
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
thamml@utar.edu.my

Yi Jie Wong
Department of Mechatronics and BioMedical Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
yjwong1999@1utar.my

Ban Hoe Kwan
Department of Mechatronics and BioMedical Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
kwanbh@utar.edu.my

Yasunori Owada
Resilient ICT Research Center
National Institute of Information and Communications Technology (NICT)
Tokyo, Japan
yowada@nict.go.jp

Myint Myint Sein
Geographical Information System
University of Computer Studies, Yangon (UCSY)
Myanmmar
myint@ucsy.edu.mm

Yoong Choon Chang
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
ycchang@utar.edu.my

***Abstract*— Recent advances in deep learning and computer vision have transformed surveillance into an important application for smart disaster monitoring systems. Based on the detected number of victims and activity of disasters, emergency response unit can dispatch manpower more efficiently, which could save more lives. However, most of existing disaster detection methods fall into the class of single-task learning, which can either detect victim or classify disaster. In contrast, this paper proposes a YOLO-based multi-task model which performs the aforementioned tasks simultaneously. This is accomplished by attaching a disaster classification head model to the backbone of a victim detection model. The head model is inherited from the MobileNetv2 architecture, and we precisely select the backbone feature map layer to which the head model is attached. For the victim detection, results reveal that the solution achieves up to 0.6938 and 20.31 in terms of average precision and frame per second, respectively. Whereas for the disaster classification, the algorithm is comparable with most deep learning models that are specifically trained for single task. This shows that our solution is flexible and robust enough to handle both victim detection and disaster classification.**

Keywords—deep learning, disaster image classification, YOLO, victim detection, multi-task learning

I. Introduction

Every year, natural disasters such as hurricanes and wildfires generate substantial amounts of damages, monetary costs, as well as injuries and deaths. For example, 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 response system plays a vital role in facilitating search and rescue efforts. Based on the reported number of victims and activity of disasters, emergency response unit can dispatch manpower more efficiently, which could save more lives. Clearly, the underlying premise behind these steps is an accurate disaster detection.

Video analytics is regarded as one of the most promising candidates for detecting disaster [3]. It prevails over dedicated sensors in the context of smoke and wildfire detections [4]. Traditional works rely on machine learning techniques, which require manual feature extraction. In [5], a hierarchical disaster image classification framework based multiple correspondence analysis was proposed to aid emergency managers in disaster response situations. The handcrafted features consist of twelve low-level color features and nine mid-level object location features. To classify disaster damage, the authors in [6] leveraged a Bag-of-Visual-Words (BoVW) model that utilizes Histogram of Oriented Gradients (HOG) handcrafted features.

Modern detection methods automatically learn high-level features through a convolutional neural network (CNN), which is known as deep learning. The output is regarded as the combination of object classification and localization. You Only Look Once (YOLO) [7] and Single-Shot Detector (SSD) [8] are two prevalent object detection methods. The former formulates the object detection as a regression problem in such a way that it can pass the input image only once to CNN for end-to-end training. Unlike YOLO, SSD does not segment the image into multiple gids and predict several bounding boxes per grid. Instead, SSD utilizes anchor boxes to make predictions on multi-scale feature map.

In this paper, we select YOLO over SSD due to its superiority in achieving the tradeoff between accuracy and speed [9]. Different from existing disaster-related works, which focus on single-task learning, we aim to propose a unified multi-task model that performs disaster classification and victim detection simultaneously. The contribution lies in eliminating the straightforward approach of running multiple individual CNN models, especially on low-powered embedded systems. The unified 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".

The rest of the paper is organized as follows. Section II describes related works. In Section III, we present the proposed solution. Section IV reports results and discussions. Section V concludes the paper.

II. Literature Review

A. *Disaster Detection*

The emergence of machine learning has paved the way for smart disaster response systems. Embedding versatile machine intelligence into various tasks of disaster detection has received considerable attention from both academia and industry communities.

Recognizing the power of CNN, the authors in [10] proposed a damage assessment method which outperforms the

BoVW model. The collected Damage Assessment Dataset (DAD) consists of four major natural disasters. Another CNN framework was adopted in [11], where multiple pretrained unimodal CNNs that extract features from raw text and images independently are combined and fed into a final classifier for disaster damage identification. Their dataset, known as Damage Multimodal Dataset (DMD), is composed of five different damage categories collected from various sources of text and images.

Inspired by the fact that Twitter has rapidly grown to a popular social network platform with a plethora of content-rich messages, the work in [12] released a large multimodal dataset collected from Twitter during different natural disasters, known as CrisisMMD. The authors in [13] took one step forward by automatically classifying tweet messages that people post during disasters into one dataset of user-defined situational awareness categories, known as Artificial Intelligence for Disaster Response (AIDR).

The performance of disaster detection model is tightly connected with the quality and quantity of dataset. On one hand, the availability of multiple datasets facilitates the learning process of CNN. On the other hand, the heterogeneity stemming from the datasets presents a hurdle for benchmarking purpose. In response, the authors in [14] first consolidated the aforementioned four datasets into a dataset called Crisis Image Benchmarks Dataset, and subsequently validated the performance on several CNN models such as VGG16 [15] and MobileNet [16]. Unlike the previous works [10-11,13] which focus on single-task classification, the same authors in [14] 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 classification tasks, without considering the additional requirement to locate the instances in an image. In contrast, our work aims to develop a multi-task learning (MTL) model which jointly executes disaster classification and victim detection.

B. *Multi-task Learning (MTL)*

MTL is to perform more tasks using one model, without the need of using separate model for each task. Generally, it can be categorized into two classes, namely hard and soft parameter sharing. Hard parameter sharing is the most frequently used approach to MTL in deep learning [18]. As illustrated in Fig. 1 (a), the general idea of hard parameter sharing is to share multiple hidden layers for all tasks, which are then branched out into several task-specific output layers. In the computer vision domain, the shared hidden layers are usually the modern CNN architectures. Although hard parameter sharing is useful in many scenarios, it could break down easily if the tasks are not closely related or require reasoning on different levels. As for soft parameter sharing, each task has its own backbone, where the parameters of each backbone are regularized to encourage them to be similar. These layers are often referred to as the constrained layers. After that, each backbone is connected to the task-specific output layers. Fig. 1 (b) shows an example of MTL using the soft parameter sharing approach.

In the context of object detection, MTL can be categorized into three types. Firstly, there are multi-task object detection models that add an additional head model(s) for other tasks (s). In such a model, a head model is branched out from the backbone or the neck of the original detector for each new task. Examples of using hard parameter sharing can be found in [19-21]. In self-driving car application, the work in [20] added another head model for lane lines detection to the joint segmentation and detection model. The algorithm in [21] adopted four head models for (i) citrus detection and (ii) segmentation, as well as (iii) maturity and (iv) quality classification on the detection citrus. On the other hand, the authors in [22] resorted to the soft parameter sharing approach, where a Task-related Attention Module (TAM) was used to share information between the two head models.

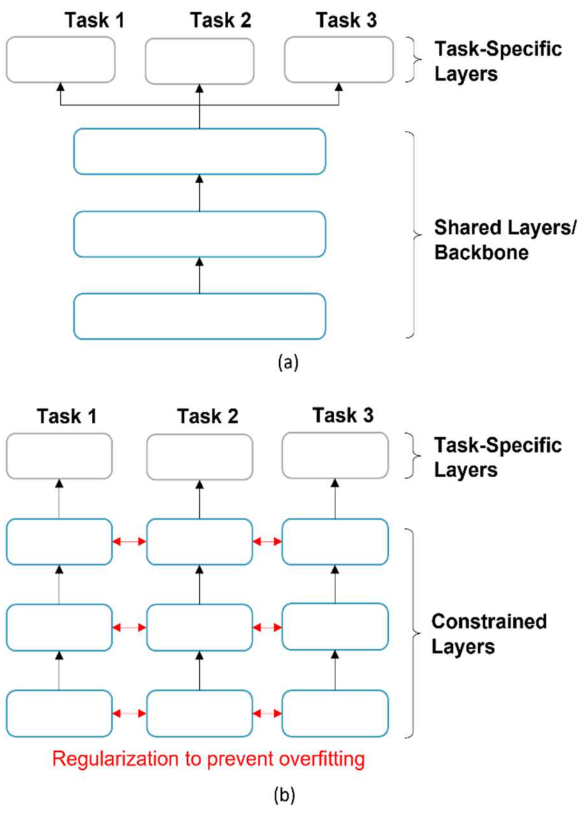


Fig. 1. Multi-task learning can be categorized into two main approaches. (a) Illustration of hard parameter sharing approach. (b) Illustration of soft parameter sharing.

Secondly, some multi-task object detection models are achieved with minimal modification on the original detector model. In [23], a multi-task object detection model was used to predict the class and the relative distance of the detected vehicle. The authors discretized the distance of the detected vehicle, making the distance prediction a classification task instead of a regression task. Then, the vehicle class and the distance labels were combined into one unified label. For instance, given M vehicles classes and N types of distance to be predicted, there will be a total of M x N new labels to be formed. As a result, the multi-task model is no different from a regular object detection model. However, it has to learn to predict the class and the distance of the detected vehicles simultaneously, making it a multi-task model.

Lastly, some multi-task object detection models used other tasks to improve the performance of object detection. In such applications, the additional tasks are auxiliary tasks, which serve as a refinement to the main task (object detection). The auxiliary tasks are added to learn features that could help the object detection head model to predict more accurately. For example, [24] used three auxiliary tasks, which include (i) closeness labelling, (ii) multi-object labelling and (iii) foreground labelling to learn additional features for the object detection head model.

III. Proposed Solution

A multi-task model is designed for victim detection and disaster classification, using the hard parameter sharing approach. A head model for disaster classification is added on top of the backbone of the selected object detection model.

A. *Object Detection Model*

As mentioned, we select YOLO as the object detection model because of its high speed and accuracy. Although YOLOv4 [25] is the latest version, YOLOv3 is still regarded as one of the most widely used object detector [26]. Without loss of generality, we focus on developing the YOLOv3 based multi-task model. The results are also expectably applicable to YOLOv4, which follows the same structural design as YOLOv3.

YOLOv3's backbone is DarkNet-53, which is a deep residual network inspired by the Residual Neural Network (ResNet). YOLOv3 also adopts the Feature Pyramid Network (FPN) as its neck model to extract features at three different scales. FPN takes three feature maps from the 82nd, 94th and the last layer from DarkNet-53 as its inputs. Lastly, three head models (or decoders) are used to detect objects based on the three different features from the FPN. The head model is very simple. It consists of two convolutional layers, where a 3 x 3 convolutional layer is followed by a 1 x 1 convolutional layer. The output channel of each head model is 3 (K + 5), where K is the total number of classes to be predicted. For the remaining five channels, four are used to predict the spatial coordinate of the bounding boxes (x, y, w, h) and one for objectness score.

Ideally, an object detection model should only predict one bounding box for each detected object. However, an object detector will likely predict more than one bounding box for each object. Thus, Non-Maximum Suppression (NMS) is applied to remove the redundant bounding boxes. After applying NMS, we set the number of bounding boxes to be the victim count. To this end, a layer will be added to compute and return the victim count as a tensor. Fig. 2 shows the model architecture of the YOLOv3 for victim detection and victim counting.

B. *Disaster Classification Head Model*

The disaster classification head model will be added to one of the output feature maps from the DarkNet-53. It is important to decide where should the head model be attached to the DarkNet-53. This is because the disaster classification task requires different high-level feature maps compared to the victim detection task. Since the disaster classification task's activation map does not focus on any victim-shape objects [17], it is important to select a feature map from DarkNet-53 that has not highly specialized for victim detection.

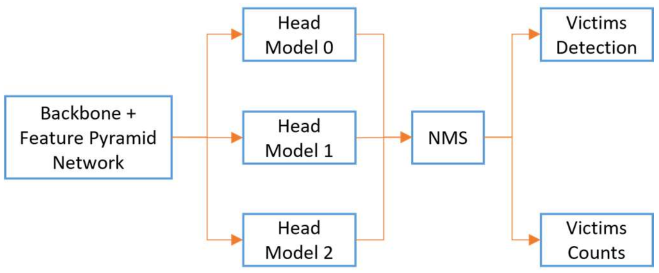


Fig. 2. The architecture of the YOLOv3 for victim detection and counting.

After some empirical testing, we decided to add the head model to the 94th layer of DarkNet-53 instead of the 82nd or the last layer. We do not attach the head model to the last layer of DarkNet-53, because the feature maps produced by this layer are highly specialized for victim detection. On the other hand, the feature maps extracted in the 82nd layer are considerably too low-level. The 94th layer of DarkNet-53 seems to provide feature maps that are not too specialized for victim detection, but still consist of some high-level features that could be shared for both tasks.

The head model for disaster classification adopts the MobileNetv2 architecture [27], because it is designed for lightweight and fast applications. Specifically, our disaster classification head model borrows the architecture of the last few blocks in the MobileNetv2, which consists of two inverted residual blocks, one pointwise convolution layer followed with a global average pooling, and one more pointwise convolution layer for the classification.

Fig. 3 shows the detailed structure of the disaster classification head model. DW represents a 3 x 3 depthwise-separable convolution layer, while PW represents a 1 x 1 conventional convolution layer (which is a pointwise convolution). BN represents a batch normalization layer, and ReLU6 is ReLU clipped at a maximum value of 6, as used in [28] for depthwise-separable convolution.

C. *The Unified Model*

A victim detection model based on YOLOv3 is used as the based model. Then, the disaster classification head model will be attached to the second feature map output by the DarkNet-53 backbone in YOLOv3. Together, a unified multi-task model for victim detection and disaster classification is formed. Fig. 4 illustrates the architecture of the model.

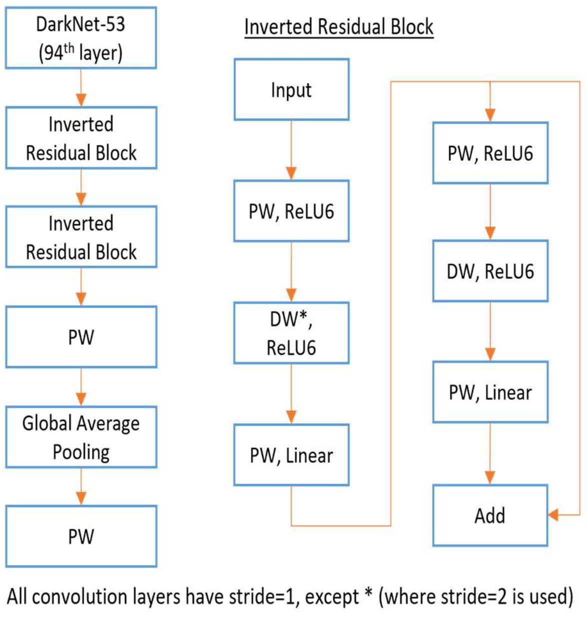


Fig. 3. The architecture of the disaster classification head model.

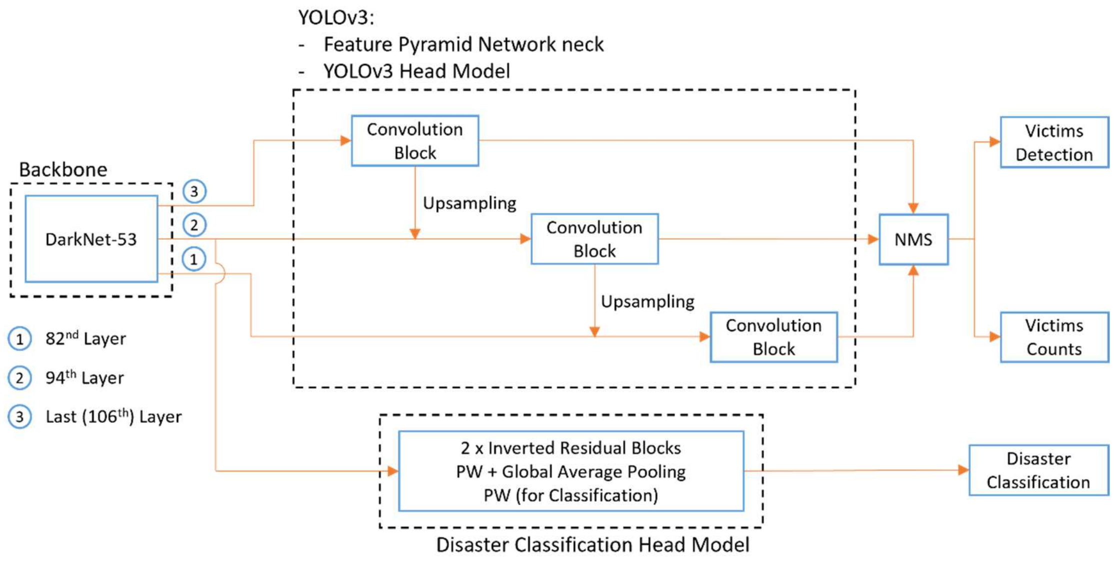


Fig. 4. The architecture of the unified model for victim detection and disaster classification. YOLOv3 is used as the base model, where a custom layer for victim counting after the NMS layer. A disaster classification head model is added to the second output feature maps from the DarkNet-53 backbone.

IV. Experiments

A. Datasets

The images used in the dataset are extracted from the Crisis Image Benchmarks Dataset [14]. The sub-dataset for disaster types is used to train both the disaster classification task and victim detection task. The class label for disaster type in the sub-dataset include (i) fire, (ii) hurricane, (iii) flood, (iv) earthquake, (v) landslide, (vi) other disasters (to cover the remaining disaster types, such as bus or car accident, plane crash, explosion, and war) and (vii) not disaster. The data split for the sub-dataset is as shown in Table I.

There is a lack of publicly available victim detection datasets. Thus, the disaster type dataset will be annotated for victim detection. The dataset is labelled using Auto-Annotate, which is built on top of a Mask R-CNN model. The pre-trained Mask R-CNN model has been forked 1400 times in GitHub, making it a reliable model. After auto-annotating the dataset, manual inspection is done to validate the generated bounding boxes. In total, 5994, 634 and 1448 images from the train, validation and test dataset, respectively, are labelled. The remaining images are not used because there is no victim in the images. Table II shows the data split for the custom victim detection dataset.

TABLE I. Data Split for Disaster Types Task.

Class Labels	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 II. Data Split for Victim Detection Dataset.

Class Labels	Count
Train	5994
Validation	634
Test	1448
Total	8076

B. Training Details

The head models for victim detection and disaster classification are trained separately. A YOLOv3 for victim detection will be trained as a base model. Then, the trained DarkNet-53 backbone will be frozen, and used as the backbone for the disaster classification head model. After the training, the head models will be attached to the trained YOLOv3, resulting a unified model. All models in this paper are trained on the NVIDIA GeForce RTX 2070 SUPER Graphic Cards. Table III shows the complete configuration of the experimental platform.

A YOLOv3 model pretrained on the COCO dataset is used for transfer learning. The weights for DarkNet-53 and FPN layers will be transferred to our model. These weights are frozen and will not be trained. On the other hand, the three head models will be initialized randomly. We trained the model for 100 epochs with 0.99 momentum and 0.0005

TABLE III. Configuration of Experimental Platform.

Names	Configuration
Operating System	Ubuntu 18.04
CPU	Intel Core i7-10875H CPU, 2.30 gigahertz
RAM (GB)	64
GPU	NVIDIA GeForce RTX 2070 SUPER
GPU Acceleration Libray	CUDA9.1, CUDNN7.6.5

weight decay. A batch size of 64 is used on one graphic processing unit only. Adam optimizer is used with the initial learning rate set to 0.001. To avoid over-fitting, adaptive learning rate method is used. The learning rate is reduced by a factor of 0.1 when the validation loss stops decreasing after three epochs. Similar training procedures are repeated for the disaster classification, except that the three head models are replaced by the customized MobileNetv2 like head model.

C. *Performance Evaluation*

The performance of the two tasks will be evaluated separately. For victim detection task, average prevision (AP) is used as the evaluation metric to evaluate the performance of the victim detection model. AP is derived from precision and recall. The definition of precision (P) and recall (R) are as shown in (1) and (2):

$$P = \frac{TP}{TP+FP} \quad (1)$$

$$R = \frac{TP}{TP+FN} \quad (2)$$

where TP, FP and FN are true positive count, false positive count, and false negative count, respectively. Based on the precision and recall, the AP can be expressed as the integral of function P of R as shown in (3).

$$AP = \int_0^1 P\,(R)dR \quad (3)$$

The train, validation and test PR curves of the victim detection model are plotted in Fig. 5. The train, validation and test AP of the model are 0.7814, 0.6907 and 0.6938, respectively. Also, the average frame per second is 20.31, making it suitable for real-time disaster monitoring.

Fig. 6 shows some examples of victim detection using our model. On the other hand, the performance of the disaster classification head model will be evaluated using accuracy, precision, recall and F1 score. Table IV compares the results with those extracted from [17].

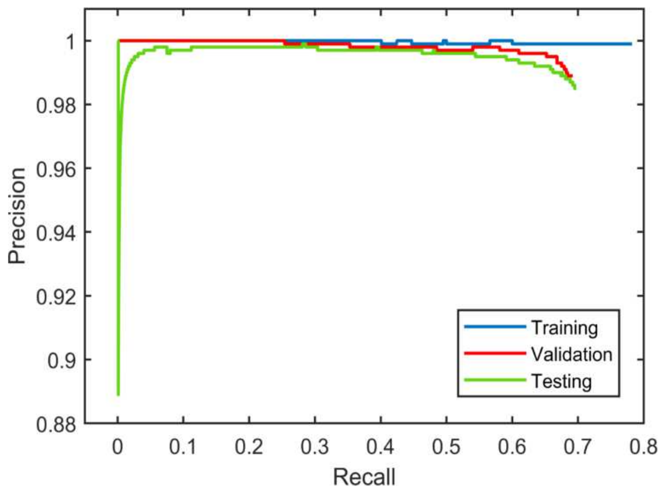
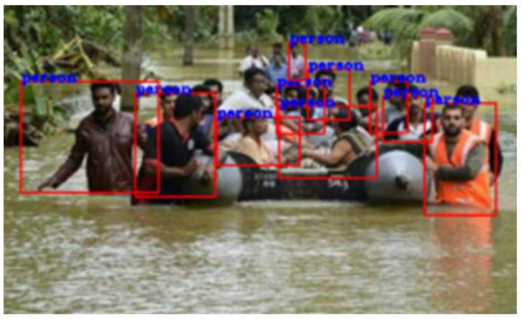
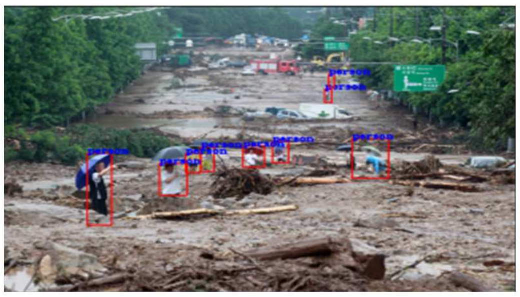


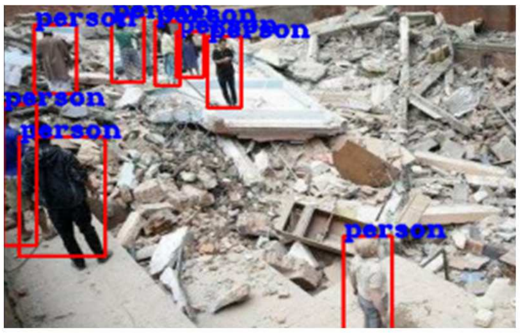
Fig. 5. Precision-recall curves of the YOLOv3 for victim detection.



(a)



(b)



(c)

Fig. 6. Victim detection at different areas. (a) Flood. (b) Landslide. (c) Earthquake.

TABLE IV. DISASTER CLASSIFICATION USING DIFFERENT MODELS.

Backbone	Accuracy	Precision	Recall	F1 Score
ResNet18	0.812	0.807	0.809	0.809
ResNet50	0.817	0.810	0.812	0.812
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
DenseNet (121)	0.817	0.811	0.813	0.813
SqueezeNet	0.726	0.719	0.717	0.717
InceptionNet (v2)	0.808	0.801	0.802	0.802
MobileNet (v2)	0.793	0.788	0.793	0.789
EfficientNet (b1)	**0.838**	**0.834**	**0.838**	**0.835**
Proposed Solution	0.792	0.827	0.769	0.766

Interestingly, the proposed solution is comparable with most CNN models that are specifically trained for single task. The ability to detect victim on top of disaster classification comes at the cost of only 2 % accuracy performance loss. As for precision, our model has the second highest precision and approximate the best model within 0.7 % gap. With regards to recall and F1 score, the proposed algorithm performs slightly worse than other models. Again, such performance drop is contributed by the hard parameter sharing setup. Overall, our solution is flexible and robust enough to handle both victim detection and disaster classification.

V. Conclusion

In this paper, we propose a multi-task model for disaster classification and victim detection. The model has a high AP of 0.6938 for victim detection, with precision and recall as high as 0.98 and 0.7. As for the disaster classification task, the performance does not surpass the best benchmark performance. However, the trade-off is acceptable since our backbone is shared for multi-tasking. In future works, the proposed models can be optimized using OpenVINO. OpenVINO performs static model analysis and redesign the model for optimal execution on an edge device. Then, the model can be deployed in an IoT framework for disaster response.

Acknowledgement

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] S. A. Shah, D. Z. Seker, S. Hameed and D. Draheim, "The Rising Role of Big Data Analytics and IoT in Disaster Management: Recent Advances, Taxonomy and Prospects," in IEEE Access, vol. 7, pp. 54595-54614, 2019, doi: 10.1109/ACCESS.2019.2913340.

[4] S. Khan, K. Muhammad, S. Mumtaz, S. W. Baik, and V. H. C. de Albuquerque, "Energy-Efficient Deep CNN for Smoke Detection in Foggy IoT Environment," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9237–9245, 2019, doi: 10.1109/JIOT.2019.2896120.

[5] Y. Yang, H. Ha, F. Fleites, S. Chen, and S. Luis, "Hierarchical disaster image classification for situation report enhancement," in 2011 IEEE International Conference on Information Reuse Integration, 2011, pp. 181–186, doi: 10.1109/IRI.2011.6009543.

[6] A. Vetrivel, M. Gerke, N. Kerle, and G. Vosselman, "Identification of Structurally Damaged Areas in Airborne Oblique Images Using a Visual-Bag-of-Words Approach," Remote Sens., vol. 8, no. 3, 2016, doi: 10.3390/rs8030231.

[7] 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.

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

[9] J. -a. Kim, J. -Y. Sung and S. -h. Park, "Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition," 2020

FPS versus processing unit. (a) YOLOv4. (b) YOLOv4 tiny. IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Nov. 2020, pp. 1-4.

[10] D. T. Nguyen, F. Ofli, M. Imran, and P. Mitra, "Damage Assessment from Social Media Imagery Data During Disasters," in *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2017, pp. 569–576.

[11] 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.

[12] F. Alam, F. Ofli, and M. Imran, "CrisisMMD: Multimodal twitter datasets from natural disasters," in *12th International AAAI Conference 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.

[13] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg, "AIDR: Artificial Intelligence for Disaster Response," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 159–162, doi: 10.1145/2567948.2577034.

[14] 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 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020, pp. 151–158, doi: 10.1109/ASONAM49781.2020.9381294.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[16] 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," arXiv:1704.04861, 2017.

[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] 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.

[19] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid, "BlitzNet: A Real-Time Deep Network for Scene Understanding," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4174–4182, doi: 10.1109/ICCV.2017.447.

[20] 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.

[21] C. Wen *et al.*, "Multi-scene citrus detection based on multi-task deep learning network," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 912–919, doi: 10.1109/SMC42975.2020.9282909.

[22] 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.

[23] 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.

[24] W. Lee, J. Na, and G. Kim, "Multi-Task Self-Supervised Object Detection via Recycling of Bounding Box Annotations," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4979–4988, doi: 10.1109/CVPR.2019.00512.

[25] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, Accessed: Aug. 17, 2021. [Online]. Available: https://arxiv.org/abs/2004.10934.

[26] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018, Accessed: Aug. 16, 2021. [Online]. Available: https://arxiv.org/abs/1804.02767.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[28] 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.

Efficient Device-Edge Inference for Disaster Classification

2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN) | 978-1-6654-8550-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICUFN55119.2022.9829668

Nathaniel Tan Sze Yang
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
nat980718@1utar.my

Mau-Luen Tham
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
thamml@utar.edu.my

Sing Yee Chua
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
sychua@utar.edu.my

Ying Loong Lee
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
leeyingl@utar.edu.my

Yasunori Owada
Resilient ICT Research Center
National Institute of Information and Communications Technology (NICT)
Tokyo, Japan
yowada@nict.go.jp

Suvit Poomrittigul
Software Engineering and Information System
Pathumwan Institute of Technology
Bangkok, Thailand
suvit@pit.ac.th

***Abstract*—Image classification can learn useful insights from crisis incidents and is gaining popularity in the field of disaster management. This is fueled by the recent advances in computer vision and deep learning techniques, where accurate neural network models for disaster type classification can be accrued. However, these studies quite commonly neglect the prohibitive inference workload which may hamper its wide-spread deployment, especially for model execution on low-powered edge devices. In this paper, we propose a lightweight disaster classification model that recognizes four types of natural disaster plus one non-disaster class. To support real-time applications, the proposed model is optimized with OpenVINO, which is a neural network acceleration platform. Different from existing works which focus on benchmarking at training stage, our experimental results reveal the actual performance at inference stage. Specifically, the optimized version achieves up to 23.93 frames per second (FPS), which is more than doubled the speed achieved by the original model, while sacrificing only 0.935 % of classification accuracy.**

Keywords—natural disaster, deep learning, disaster image classification, OpenVINO, benchmarking

I. Introduction

Artificial intelligence (AI) algorithms are developed with the intention of making decisions in real life. Moving forward, convolutional neural network (CNN), which is an advanced version of AI, is able to learn more meaningful insights from images. The training process of these neural network models can be facilitated by open-source deep learning (DL) frameworks such as TensorFlow [1] and Keras [2]. The growing popularity of CNN have paved the way for new computer vision applications. One specific area would be disaster management [3], where video surveillance cameras and sensors can be leveraged to gain situational awareness.

A natural disaster is an incident caused by nature's threat. It can be defined as a natural phenomenon that causes the health impacts of mankind, loss of livelihoods and services, social and economic disruption, or properties and environmental damage [4]. Some examples are tornadoes, earthquakes, floods, and wildfires. Monitoring these disasters at large-scale coverage would require a plethora of Internet of things (IoT) devices [5], which often have long-range transmission range but low computational power.

Existing works for disaster classification quite commonly neglect the prohibitive inference workload which may hamper its wide-spread deployment, especially for model execution on low-powered edge devices. In this paper, we propose a lightweight disaster classification model that identifies four types of natural disasters and one non-disaster class. 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".

The contributions in this study are threefold. First, we consolidate a dataset which consists of natural disaster and non-disaster images (natural sceneries). Second, we employ the transfer learning approach to output a disaster classification model before optimizing the model with OpenVINO. Third, we provide benchmark results for both training and inference stages, which sheds more insights into the actual implementation performance.

The rest of the paper is organized as follows. Section II discusses the related works. Section III describes the proposed solution. Section IV presents the experimental results and discussions. Section V concludes the article.

II. Related Work

DL, especially CNN has gained momentum in disaster monitoring. According to [6], majority works have focused on CNN instead of machine learning (ML) methods due to its superior performance. Such gain, however, are only possible under the availability of abundant labelled datasets.

Recognizing the importance of datasets, the authors in [7] consolidated a substantial amount of human detection and action detection dataset for disaster management application. The goal was to develop a DL-based drone surveillance framework. However, the framework did not consider disaster event classification. A similar work can be found in [8], where the authors utilized various CNN architectures including ResNet50, Inception V3 and AlexNet in identifying survivors in debris. This time, the annotated images were focused on earthquake-hit regions.

The work in [9] focused on classifying disaster events after collecting more than 7000 images consisting of cyclones, drought, earthquakes, floods, landslides, thunderstorms, snowstorms, and wildfires, from social media platforms. The burden of annotating data was relieved by adopting active learning, which automatically chooses and labels the data from which it learns without human interaction. The authors in [10] further divided disaster-related images into four different categories, namely disaster type detection, informativeness, humanitarian and damage severity. By setting binary and multiclass classification labels on a consolidated dataset, benchmark results using several CNN architectures were provided.

Apart from the disaster-related images, text messages contain critical information such as infrastructural damages, casualties, and help requests. The usefulness of such social media data has motivated the authors in [11] to develop a multimodal fusion model, which combines both visual and textual features to classify relevant disaster images. However, all the above studies focused on accuracy measurement, where high-end graphics processing unit (GPU) such as NVIDIA Tesla P100 GPU was utilized. Deploying these trained models directly on resource-constrained edge devices remains a challenging task [12]. This is especially true for real-time disaster monitoring applications.

Different from the aforementioned works, the authors in [13] assessed the CNN performance in terms of accuracy and speed. Results showed that their proposed model was able to achieve 9 frames per second (FPS) on a low-powered embedded device, while maintaining reasonable accuracy. However, they did not explore the potential of neural network optimization on target devices at the inference stage. Such performance acceleration is made possible with an open-source CNN model inference engine called OpenVINO Toolkit [14]. The study in [15] benchmarked several pretrained CNN models under the OpenVINO settings. However, it remains unaddressed as in how much improvement can be brought to implementation by OpenVINO, as compared to the unoptimized version.

III. DL Model Deployment

To achieve a robust DL model, training and inference phases must be analyzed correctly. To this end, we propose the methodology shown in Fig. 1, where the three stages are necessary to evaluate the actual performance.

A. Model Training

A new natural disaster classification model is trained using the transfer learning approach. Without loss of generality, we select VGG16 to be the neural network architecture due to its high accuracy [6,10,11,13,14, 15]. The results are also expectably applicable to other architectures such as DarkNet-53 [16]. The collected dataset contains natural disaster and non-disaster images, which were downloaded from public sources: [17] and [18], respectively. The natural disaster data consists of cyclones, earthquakes, floods, and wildfires. On the other hand, the non-disaster images comprise of nature scenes such as coast, mountain, forest, open country, as well as man-made scenes like street, inside city, buildings, and highways. Table I summarizes the data distribution among training, testing, and validation.

The dataset split is 67.5 % for training, 25 % for testing, and 7.5 % for the validation split. The parameters to fine-tune the VGG16 model are shown in Table II. The batch size means the number of images from the dataset that are selected from the beginning and used to train the natural disaster classification model in each iteration throughout the training dataset. The number of steps is the number of iterations. After each step or iteration, the gradient of the natural disaster classification model will be updated. Once all images of the training dataset are gone through, one epoch is completed. The values of the minimum and maximum learning rates are used by the cyclical learning rate (CLR) technique to improve the accuracy of the model [19].

CLR requires the minimum and maximum boundary values before it can be used. A test on the learning rate range is executed whereby training starts at a lowest learning rate of $10\,e^{-10}$. After each batch update, the learning rate will increase exponentially until it reaches a rate of $10e^{1}$, and the current learning rate and loss will be logged simultaneously. The loss gives the idea of how the model performs in the training and validation datasets as shown in Fig. 2. The CLR technique makes the learning rate moves cyclically between the set boundaries as shown in Fig 3.

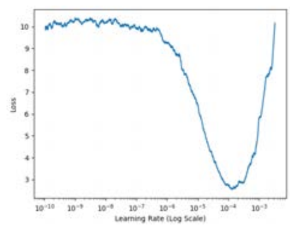


Fig. 2. Learning rate range test

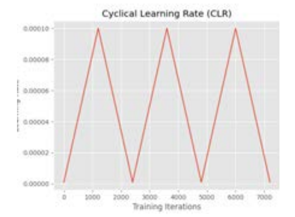


Fig. 3. CLR plot

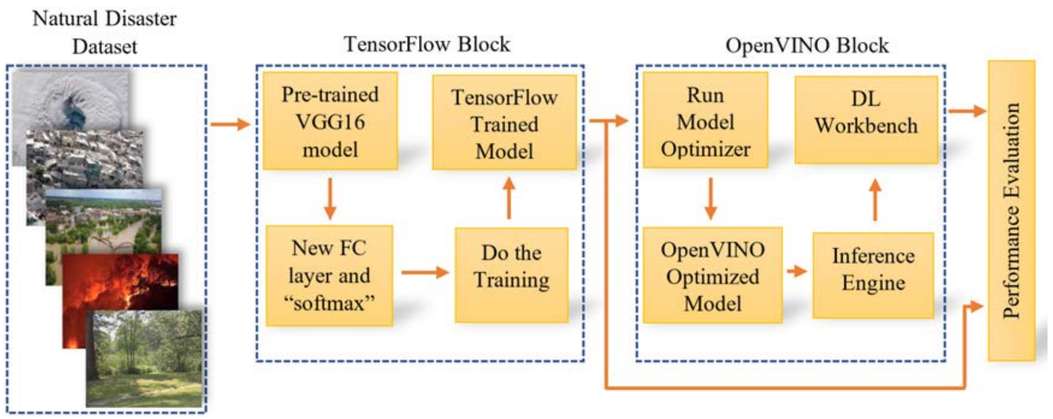


Fig. 1. Methodology for performance evaluation of the proposed model using OpenVINO Deep Learning Workbench.

TABLE I. DATA SPLIT FOR DISASTER CLASSIFICATION

Disaster Label	Train	Validation	Test	Total
Cyclone	599	78	251	928
Earthquake	923	86	341	1350
Flood	741	80	252	1073
Wildfire	724	68	285	1077
Non-Disaster	1821	223	652	2696

TABLE II. PARAMETERS AND ITS VALUES FOR FINE-TUNING THE VGG16 MODEL

Parameter	Value
Batch Size	32
Number of Steps	8
Epoch	48
Min Learning Rate	1e-6
Max Learning Rate	1e-4

Along with the immediate responses, the correctness of the result is also a very crucial parameter for such applications.

B. Model Optimization

To reach the goal of this study, we proposed the method shown in Fig. 1. There are five processes to get the output predictions for each image in the inference stage. Each stage will be discussed in the following sections.

1) *Obtaining the trained model.* A transfer learning approach is applied to the pre-trained VGG16 model to output a new natural disaster classification model. Fine-tunings and parameters are modified with the intention of minimizing execution time and increasing accuracy.

2) *Freezing the model.* The model is saved as a .pb file with the weights frozen during the training stage. TensorFlow version 2.0 is used to execute the training and freeze the model.

3) *Model conversion to a compatible format*. Conversion of the trained model into Intermediate Representation (IR) format needs to be done in order for it to be used in the OpenVINO environment [20]. OpenVINO's model optimizer tool is used to perform the conversion, with the following code and parameter:

```
python3 mo_tf.py --saved_model_dir <model-path>
--output_dir <output-dir> --input_shape
[1,224,224,3]
```

The parameter –input_shape [1,224,224,3] defines the input data properties of the model in the training as follows: Number of images [N] × Height [H] × Weight [W] × Channels [C]. Note that [N,H,W,C] is for a TensorFlow model. After a successful conversion of the model to IR format, a .xml (describes the network topology) and a .bin (contains the weights and biases binary data) file will be generated [21][22].

4) *Executing inferences*. In this stage, OpenVINO's Inference Engine tool is used to perform the inference. A custom Python script is executed to initiate plugins, load IR model, read the label, input data, infer and process the output. The alternative of using a python script is the Deep Learning Workbench (DL Workbench).

5) *Performance evaluation*. The original (TensorFlow) model and optimized (OpenVINO) model are evaluated based on the 25 % test dataset, for precise and accurate measurements. For the original model, it is evaluated using the *classification_report* function in TensorFlow while the optimized model uses DL Workbench. The precision of the TensorFlow's natural disaster classification model is floating-point (FP) 32. In the optimized model, the precision can be FP 32, FP 16, and integer (INT) 8. In theory, a higher precision gives a higher accuracy but requires higher computational power, and vice versa.

C. Model Inference

There are two inference modes: synchronous and asynchronous. The data were fed into the inference engine in a synchronous manner, allowing only one image to be processed per inference. Asynchronous inference, on the other hand, speeds up the process by inferencing one image while pre-processing the next image.

The script to run the inference of the TensorFlow model is in asynchronous mode. By default, the inference model in the Inference Engine of OpenVINO also uses asynchronous mode. However, there are certain disadvantages to this method, as acquiring the predictions comes after all of the flow is completed.

IV. EXPERIMENTAL RESULTS

The proposed model is evaluated using the dataset provided in [16] and [17]. Sample images from the dataset can be seen in Fig 1. The dimension of the images varies throughout the dataset. Image pre-processing based on the required input size of 224 × 224 pixels is done before the training or inferencing of the model.

Regarding the hardware used, there are two environments that are used to carry out the experiments. Their main specifications are described in the following items.

1) *Hardware on training phase*: The hardware used in this phase is an Intel NUC equipped with a 10th–generation i7-10710U 6-core Intel processor with 64 GB of memory and 1 TB of a solid-state drive as the storage. The operating system in the Intel NUC is the Ubuntu 18.04 LTS version. The software used for training is TensorFlow version 2.0.

2) *Hardware on the inference phase*: The hardware is similar to that of the training phase. The difference is that the inference for the optimized model is able to take advantage of the Intel integrated graphics, which is Intel UHD Graphics. The optimized model is executed in the OpenVINO environment, and the version is OpenVINO 2021.4.

A. Training Performance

Once the training of the model is completed, a performance test of precision, recall, f1-score, and accuracy is executed by the function *classification_report*. The 25 % test dataset is used for the performance evaluation and has a total number of 1781 images.

Table III shows the performance results of the TensorFlow model. It is noteworthy that *precision* gives us an idea of how well the model classifies a natural disaster when it output/classify a natural disaster; *recall* ratio gives us an idea of how well the model classifies a natural disaster, given an input of natural disaster scenario; *F1-score* is the harmonic mean between precision and recall; *support* is the number of occurrences (or images used) in each class to produce the results; and *accuracy* is the ratio of the correct predictions to all of the predictions.

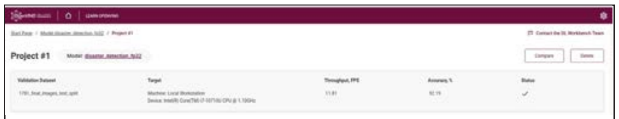
The performance of TensorFlow's model has achieved an accuracy of 93 %. A few videos have been used as inputs to the model and it achieved an average of 7.70 FPS. Note that the same test dataset is used to evaluate the performance of the optimized model in the subsequent sub-section.

B. Inferences Performance

Once the trained model is converted successfully into OpenVINO IR format, the optimized model is used by the Inference Engine to do inferencing. The DL Workbench tool is used to obtain the inference performance of the optimized model. Figs. 4 - 7 shows the inference performance overview obtained from the DL Workbench.

TABLE III. RESULTS OF TENSORFLOW NATURAL DISASTER CLASSIFICATION MODEL

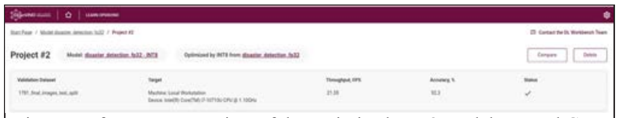
Disaster Class	Precision	Recall	F1-Score	Support
Cyclone	96%	98%	97%	251
Earthquake	94%	92%	93%	341
Flood	84%	90%	87%	252
Wildfire	93%	93%	93%	285
Non-Disaster	96%	93%	95%	652


Project #1 Model: disaster_detection_fp32

Validation Dataset	Target	Throughput, FPS	Accuracy, %	Status
1781_final_images_test_split	Machine: Local Workstation Device: Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz	11.81	92.19	✓

Start Page / Model disaster_detection_fp32 / Project #1
Contact the DL Workbench Team
Compare
Delete

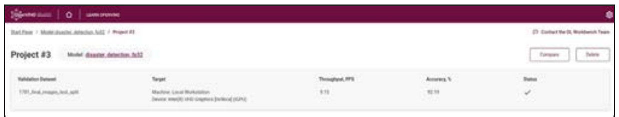
Fig. 4. Performance overview of the optimized FP32 model on Intel CPU


Project #2 Model: disaster_detection_fp32 - INT8 Optimized by INT8 from disaster_detection_fp32

Validation Dataset	Target	Throughput, FPS	Accuracy, %	Status
1781_final_images_test_split	Machine: Local Workstation Device: Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz	21.35	92.3	✓

Start Page / Model disaster_detection_fp32 / Project #2
Contact the DL Workbench Team
Compare
Delete

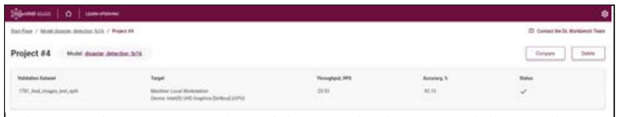
Fig. 5. Performance overview of the optimized INT8 model on Intel CPU


Project #3 Model: disaster_detection_fp32

Validation Dataset	Target	Throughput, FPS	Accuracy, %	Status
1781_final_images_test_split	Machine: Local Workstation Device: Intel(R) UHD Graphics [default] (GPU)	9.15	92.19	✓

Start Page / Model disaster_detection_fp32 / Project #3
Contact the DL Workbench Team
Compare
Delete

Fig. 6. Performance overview of the optimized FP32 model on Intel GPU


Project #4 Model: disaster_detection_fp16

Validation Dataset	Target	Throughput, FPS	Accuracy, %	Status
1781_final_images_test_split	Machine: Local Workstation Device: Intel(R) UHD Graphics [default] (GPU)	23.93	92.13	✓

Start Page / Model disaster_detection_fp16 / Project #4
Contact the DL Workbench Team
Compare
Delete

Fig. 7. Performance overview of the optimized FP16 model on Intel GPU

TABLE IV. RESULTS OF OPENVINO OPTIMIZED MODEL RUNNING ON CPU

Precision	FP 32	FP 16	INT 8
Throughput (FPS)	11.81	-	21.35
Accuracy (%)	92.19	-	92.30

TABLE V. RESULTS OF OPENVINO OPTIMIZED MODEL RUNNING ON INTEGRATED GPU

Precision	FP 32	FP 16	INT 8
Throughput (FPS)	9.15	23.93	-
Accuracy (%)	92.19	92.13	-

Table IV presents the performance results of the optimized model that runs on an Intel CPU. The INT 8 precision model has achieved an increase of 80.8 % FPS and 0.119 % accuracy, showing higher performance as compared to the FP 32 precision model. Hence, the best performance with Intel CPU is the INT 8 precision model. The FP 16 precision model is not available as it will upscale the model to the FP 32 to perform inference due to the limitation of DL workbench and the particular Intel CPU used in this study.

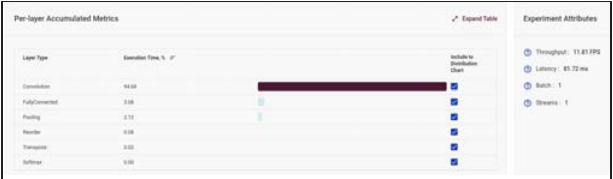
Table V shows the performance results of the optimized model that runs on an Intel integrated GPU. The FP 16 precision model has obtained 162 % higher FPS while sacrificing 0.0650 % accuracy, compared to the FP 32 precision model. The INT 8 precision model is not supported on the integrated GPU model [25]. Since the accuracy drop in the FP 16 precision model is very low, along with the considerable increase of throughput, the FP 16 precision model provides the best performance on Intel integrated GPU hardware.

Since TensorFlow's model runs on the CPU, to ensure reliable and accurate results, the comparison is done on the optimized FP 32 and INT 8 precision models of the optimized model that ran on the same CPU. The optimized FP 32 precision model achieves an increase of 53.4 % in throughput with a loss of 0.871 % in accuracy. Meanwhile, the optimized INT 8 precision model achieves an increase of 177 % in throughput at the cost of 0.753 % in accuracy.

On the other hand, if the program is implemented on an edge device, which is the Intel NUC in our study, and the optimized model is able to run on the Intel integrated GPU hardware. Only the optimized FP 32 and FP 16 precision model is able to take advantage of the GPU hardware. Since the optimized FP 16 precision model provides the best performance on GPU hardware, it achieves a substantial improvement of 211 % in throughput while only sacrificing 0.935 % accuracy as compared to the TensorFlow's model.

The comparisons show that the OpenVINO optimized models have a better performance enhancement over the TensorFlow's model in terms of frame rate inference while losing a negligible amount of accuracy.

DL Workbench is able to display the results of the performance summary of the model as shown in Figs. 8 - 11. This allows us to identify the throughput, latency, batch, and streams values of the model.



Per-layer Accumulated Metrics

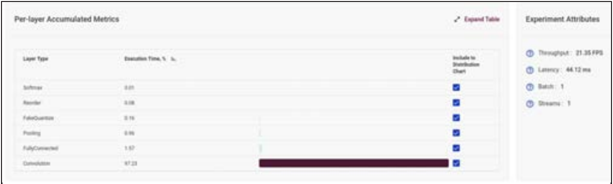
Expand Table

Layer Type	Execution Time, %	Include to Distribution Chart
Convolution	94.68	✓
FullyConnected	3.08	✓
Pooling	2.12	✓
Reorder	0.08	✓
Transpose	0.02	✓
Softmax	0.00	✓

Experiment Attributes

Throughput : 11.81 FPS
Latency : 81.72 ms
Batch : 1
Streams : 1

Fig. 8. Performance results of the optimized FP32 model on Intel CPU



Per-layer Accumulated Metrics

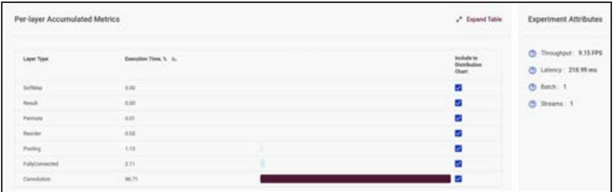
Expand Table

Layer Type	Execution Time, %	Include to Distribution Chart
Softmax	0.01	✓
Reorder	0.08	✓
FakeQuantize	0.16	✓
Pooling	0.96	✓
FullyConnected	1.57	✓
Convolution	97.23	✓

Experiment Attributes

Throughput : 21.35 FPS
Latency : 44.12 ms
Batch : 1
Streams : 1

Fig. 9. Performance results of the optimized INT8 model on Intel CPU



Per-layer Accumulated Metrics

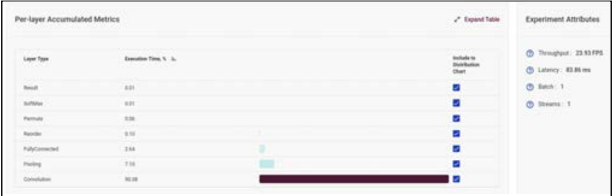
Expand Table

Layer Type	Execution Time, %	Include to Distribution Chart
SoftMax	0.00	✓
Result	0.00	✓
Permute	0.01	✓
Reorder	0.03	✓
Pooling	1.13	✓
FullyConnected	2.11	✓
Convolution	96.71	✓

Experiment Attributes

Throughput : 9.15 FPS
Latency : 218.99 ms
Batch : 1
Streams : 1

Fig. 10. Performance results of the optimized FP32 model on Intel GPU



Per-layer Accumulated Metrics

Expand Table

Layer Type	Execution Time, %	Include to Distribution Chart
Result	0.01	✓
SoftMax	0.01	✓
Permute	0.06	✓
Reorder	0.10	✓
FullyConnected	2.64	✓
Pooling	7.10	✓
Convolution	90.08	✓

Experiment Attributes

Throughput : 23.93 FPS
Latency : 83.86 ms
Batch : 1
Streams : 1

Fig. 11. Performance results of the optimized FP16 model on Intel GPU

In this part, we evaluate the performance in terms of throughput and latency. It is noteworthy that throughput is the number of images processed in a certain amount of time, which is one second, and latency is the amount of time used to perform an inference for a single image [23]. The INT8 model on the CPU and the FP16 model on the Intel GPU achieve a high throughput while only the INT8 model achieves the lowest latency of 44.12 milliseconds.

V. Conclusion and Future Lines

Natural disasters happen all around the world. Early detection of natural disasters for the people staying around the danger zone can enable safe evacuation of the people to a nearby shelter. The main issue that the current study aims to address is the unbalanced dataset and the need of powerful hardware for performing inference. To overcome the dataset limitation, we have consolidated a natural disaster dataset, and trained a new natural disaster classification model to classify natural disaster and non-disaster scenarios. We have addressed the need for powerful hardware by deploying the trained model into the OpenVINO platform. Lastly, we have evaluated the performance of the trained model and concluded that the model performs significantly better in the OpenVINO environment as compared to the TensorFlow environment. DL Workbench is a great tool for conversion of model, analysis of the converted model, as well as the performance measurement that can be done on it. As for future research works, the power consumption of the model running in different environments can be measured and it will be used as one of the performance metrics.

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] "TensorFlow", [online] Available: https://www.tensorflow.org/overview.

[2] "Keras", [online] Available: https://keras.io/

[3] Lopez-Fuentes, L., van de Weijer, J., González-Hidalgo, M. *et al.* Review on computer vision techniques in emergency situations. *Multimed Tools Appl* **77,** 17069–17107 (2018). https://doi.org/10.1007/s11042-017-5276-7

[4] Chen, Y., Li, C., Chang, C. and Zheng, M., 2021. Identifying the influence of natural disasters on technological innovation. *Economic Analysis and Policy*, 70, pp.22-36.

[5] M. A. Al-Mashhadani, M. M. Hamdi and A. S. Mustafa, "Role and challenges of the use of UAV-aided WSN monitoring system in large-scale sectors," *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2021, pp. 1-5, doi: 10.1109/HORA52670.2021.9461292.

[6] R. R. Arinta and E. Andi W.R., "Natural Disaster Application on Big Data and Machine Learning: A Review," *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2019, pp. 249-254, doi: 10.1109/ICITISEE48480.2019.9003984.

[7] B. Mishra, D. Garg, P. Narang and V. Mishra, "Drone-surveillance for search and rescue in natural disaster", *Computer Communications*, 2020.

[8] Chaudhuri N, Bose I (2020) Exploring the role of deep neural networks for post-disaster decision support. Decis Support Syst 130:113234. https://doi.org/10.1016/j.dss.2019.113234

[9] 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," in *IEEE Access*, vol. 8, pp. 208518-208531, 2020, doi: 10.1109/ACCESS.2020.3038676.

[10] F. Alam, F. Ofli, M. Imran, T. Alam and U. Qazi, "Deep Learning Benchmarks and Datasets for Social Media Image Classification for Disaster Response," *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020, pp. 151-158, doi: 10.1109/ASONAM49781.2020.9381294.

[11] Zou, Z.; Gan, H.; Huang, Q.; Cai, T.; Cao, K. Disaster Image Classification by Fusing Multimodal Social Media Data. *ISPRS Int. J. Geo-Inf.* 2021, *10*, 636

[12] Z. Jiang, T. Chen, and M. Li, "Efficient Deep Learning Inference on Edge Devices", in *Proceedings of ACM Conference on Systems and Machine Learning (SysML'18)*, 2018.

[13] C. Kyrkou and T. Theocharides, "Deep-learning-based aerial image classification for emergency response applications using unmanned aerial vehicles", *Proc. IEEE Conf. Comput. Vision Pattern Recognit. Workshops*, pp. 517-525, Jun. 2019.

[14] A. Rosebrock, Detecting Natural Disasters with Keras and Deep Learning:PyImagesearch, 2019, [online] Available: https://pyimagesearch.com/2019/11/11/detecting-natural-disasters-with-keras-and-deep-learning/.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv:1409.1556*, 2014.

[16] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018

[17] Gautam Kumar, Google Images, 2019 (accessed March 23, 2022). [Online]. Available: https://drive.google.com/file/d/1NvTyhUsrFbL91E10EPm38IjoCg6E2c6q/view

[18] Gautam Kumar, Google Images, 2019 (accessed March 23, 2022). [Online]. Available: https://drive.google.com/file/d/11KBgD_W2yOxhJnUMiyBkBzXDPXhVmvCt/view

[19] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 464-472, doi: 10.1109/WACV.2017.58.

[20] Docs.openvinotoolkit.ai. 2022. *Converting a Model to Intermediate Representation (IR)*. [online] Available at: <https://docs.openvino.ai/latest/openvino_docs_MO_DG_prepare_model_convert_model_Converting_Model.html> [Accessed 21 March 2022].

[21] Docs.openvinotoolkit.ai. 2022. *Model Optimizer Developer Guide*. [online] Available at: <https://docs.openvino.ai/latest/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html> [Accessed 21 March 2022].

[22] Docs.openvinotoolkit.ai. 2022. *Converting a Model Using General Conversion Parameters*. [online] Available at: <https://docs.openvino.ai/2021.1/openvino_docs_MO_DG_prepare_model_convert_model_Converting_Model_General.html> [Accessed 24 March 2022].

[23] Docs.openvinotoolkit.ai. 2022. *DL Workbench Key Concepts*. [online] Available at: <https://docs.openvino.ai/2021.3/workbench_docs_Workbench_DG_Key_Concepts.html> [Accessed 28 March 2022].

Artificial Intelligence of Things (AIoT) for Disaster Monitoring using Wireless Mesh Network

Mau-Luen Tham
Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Selangor 43000, Malaysia
thamml@utar.edu.my

Yi Jie Wong
Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Selangor 43000, Malaysia
yjwong1999@1utar.my

Ban-Hoe Kwan
Department of Mechatronics and Biomedical Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Selangor 43000, Malaysia
kwanbh@utar.edu.my

Xin Hao Ng
Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman, Selangor 43000, Malaysia
penguinng0928@gmail.com

Yasunori Owada
Resilient ICT Research Center, Network Research Institute, National Institute of Information and Communications Technology (NICT), Tokyo 184-8795, Japan
yowada@nict.go.jp

ABSTRACT

The inherent characteristics of Internet of things (IoT) such as low computation power of IoT nodes and transmission reliability of IoT links demand a new paradigm for efficient data processing and dissemination. This is especially true for disaster situations with high possibility of communication breakdowns. On one hand, the concept of artificial intelligence of things (AIoT) has been introduced as a technology to push data storage and computing closer to the network edge. On the other hand, wireless mesh network offers a strong self-healing capability and network robustness against disaster damages. To enable smart disaster monitoring applications, we first implement a lightweight multi-task model that performs joint disaster classification and victim detection. These AI outputs are then wirelessly synchronized via a mesh network solution called NerveNet. All the experiments are conducted in a real urban environment, including static and mobile nodes. Experimental results validate the effectiveness of the proposed solution, where text and images can be synchronized within two minutes across a multi-hop Wi-Fi network. Furthermore, the optimized AI model has ultra-low power consumption around 1.23 W with frames per second (FPS) of 2.01.

CCS CONCEPTS

• **Artificial intelligence**; • **Network experimentation**; • **Wireless integrated network sensors**;

KEYWORDS

AIoT, OpenVINO, Multi-Task Learning, Disaster Management, Wireless Mesh Network

ACM Reference Format:
Mau-Luen Tham, Yi Jie Wong, Ban-Hoe Kwan, Xin Hao Ng, and Yasunori Owada. 2023. Artificial Intelligence of Things (AIoT) for Disaster Monitoring using Wireless Mesh Network. In *2023 The 6th International Conference on Software Engineering and Information Management (ICSIM 2023), January 31–February 02, 2023, Palmerston North, New Zealand.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3584871.3584905

1 INTRODUCTION

Internet of things (IoT) is a network of a cluster of connected embedded devices with identifiers. IoT provides many functions such as intelligent information processing, reliable information transmission and overall information perception. These characteristics of IoT can provide an effective guarantee for disaster forecasting, detection, and precaution ahead of time through the IoT-based early warning system so that the impact of a disaster can be reduced.

In the traditional IoT framework, these data are transmitted to a remote central cloud platform through the Internet to be processed. However, there is an issue where the big data transmission process consumes enormous energy, time, cost, and bandwidth. Therefore, edge computing is introduced to process and analyze the valuable information from the raw sensor data at the network edge in real-time [1]. Thus, it can improve the quality of service (QoS) of applications and reduce the task latency [2].

The evolution of edge computing technology has driven the smart applications towards the use of artificial intelligence (AI) / machine learning (ML) / deep learning (DL) algorithms such as convolutional neural networks (CNN) in image analysis, and recurrent neural network (RNN) in semantic analysis. The fusion technology of AI and IoT is referred to as artificial intelligence of things (AIoT). The feasibility of this new paradigm has been demonstrated in various personal and business applications [3]. However, the limited processing capacity constraints of IoT devices present a challenge to integrate AI into AIoT applications [4]. Therefore, various efforts have been put to improve the AI performance in terms of speed and power consumption. For example, the work in [5] shows how ML can be used for flood detection based on weather data recorded by IoT sensors such as (i) relative humidity, (ii) temperature, (iii) pressure, (iv) month, and (v) estimated rainfall range.

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. Such approach is ill-suited for AIoT applications due to high memory footprint and computing power. A better candidate would be using a multi-task learning model, as proposed in [6], which offers faster frames per second (FPS) and more accurate prediction.

Transmission reliability is another challenge that deserves further study in the context of AIoT. This is especially relevant for disaster situations, where communications infrastructure, such as cellular base stations (BSs) may be destroyed. Mesh network can combat against node failures by using several redundant links and paths to the destination node. If one of these node fails, then other nodes can be used to reroute the data until it reaches the destination sink node. In this paper, we implement a mesh networking solution called NerveNet, which enables fast route switching on layer 2 [7]. The main contribution of this paper is that we assess the experimental performance of AIoT based disaster monitoring application with real implementation of edge AI and wireless mesh network. Note that the proposed solution facilitates mesh network database synchronization, which is one of goals of the ASEAN IVO project titled "Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network".

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 discusses the hardware and configuration used to build the testbed. Section 4 presents the experimental performance evaluation. Section 5 concludes the paper.

2 RELATED WORK

2.1 Artificial Intelligence of Things (AIoT)

Several existing works [8]–[10] explored the potential of AIoT for situational awareness and disaster recovery operations. An ideal disaster resilient smart cities model is one that could utilize IoT devices such as smart cameras, drones, radio frequency identification (RFID), and sensors in the cities for rapid data collection, coupled with mobile edge computing for real-time computing. AIoT has shown its potential in disaster response for both real time data collection and inferencing. The authors in [11] demonstrated how sequence model could predict the flow rates in downstream gauging station based on the flow rate in upstream station. The study in [12] utilized signals from fire detection system to predict the potential of house fire and alert the appropriate authorities using IoT networks.

Edge AI is a special type of AIoT, which brings computing closer to the data sources, which could be the connected IoT devices or local edge servers [13]. In edge AI applications, IoT devices will deploy the AI model locally, bypassing the need of sending the input data to a cloud server for model inferencing before obtaining the inference output. This could significantly reduce the latency, especially if the input data is bandwidth-hungry such as high-resolution images. However, IoT devices generally are low-powered devices with limited computational capacity, which might inhibit the large-scale deployment of edge AI. Thus, recent efforts have focused on optimizing ML models via methods such as model compression and knowledge distillation [14-15]. Among these efforts, Intel OpenVINO toolkit emerges as an extremely useful tool for edge AI facilitation. It is an open-source and production-ready that optimizes DL models across any target Intel hardware while minimizing the inference time [16]. It also comes with a complementary tool called OpenVINO DL Workbench, which provides a user-friendly dashboard for model's performance measurement, optimization and deployment using OpenVINO. A plethora of works exploited OpenVINO for inference optimization, such as license plate detection [17].

Arduino and Raspberry Pi are both suitable candidates for AIoT, However, Raspberry Pi is preferred for complicated projects [18], especially when dealing with edge AI deployment. In fact, OpenVINO provides a detailed documentations on OpenVINO installation for Raspberry Pi [19].

2.2 Disaster Classification and Victim Detection

Literature on disaster classification often surrounds the dataset since the robustness of disaster monitoring is tightly correlated with the quality and quantity of training data. There are five major datasets for disaster classification, which are Artificial Intelligence for Disaster Response (AIDR) [20], Damage Multimodal Dataset (DMD) [21], Damage Assessment Dataset (DAD) [22], CrisisMMD [23] dataset, and Crisis Image Benchmark Datasets (CrisisIBD) [24]. The most notable dataset among all is the CrisisIBD dataset, which comprises of all the aforementioned dataset. It is labelled for four different tasks: (i) disaster type classification, (ii) informativeness, (iii) humanitarian categories, and (iv) damage severity. The consolidated dataset is meant for benchmarking in deep learning tasks related to disaster response, to address the lack of benchmark datasets for disaster response domain.

For victim detection task, there is a lack of a proper benchmark dataset. However, victim detection is essentially an object detection task. Thus, pretrained object detection models such as You Only Look Once (YOLO) and Single Shot Multi-Box Detector (SSD) can be directly adopted without further fine-tuning. In [6], a pretrained YOLOv3 was used for transfer learning, and fine-tune the model for a custom victim detection dataset. Meanwhile, [25] explored similar problem by considering a thermal camera-based victim detection. Similarly, a MobileNet-SSD pretrained on VOC dataset was fine-tuned on a custom thermal dataset. Generally, all of the above works [6], [25] could achieve a robust victim detection model without needing a large size victim detection dataset, since they are adopted from pretrained object detection model.

2.3 Multi-Task Learning

Another pool of literature focuses on solving multiple tasks using one unified model, which is termed as multi-task learning (MTL). MTL can be categorized into two types, which are hard parameter sharing and soft parameter sharing. In the first category, different

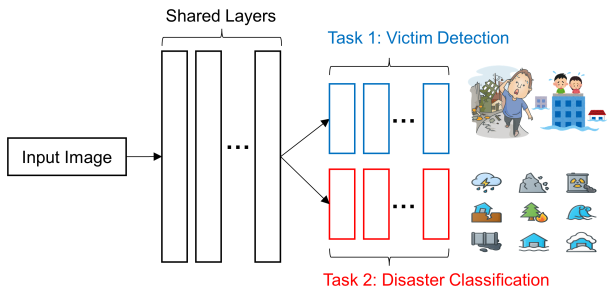


Figure 1: Proposed multi-task learning model.

tasks share the same backbone for feature extraction, while having separate head models (added on top of the backbone) for individual prediction. Whereas for soft parameter sharing, each task is allowed to have its own backbone. Clearly, hard parameter sharing approach is more suitable for edge AI, as it requires lesser computation workload since each task shares the same backbone. Examples of hard parameter sharing can be found in [26] and [27]. In [26], a MTL model adopted four head models for joint citrus detection and segmentation with citrus maturity and quality classification. Meanwhile, [27] proposed a MTL model for traffic object detection, with road segmentation and lane line detection.

There are limited works in disaster response domain that address multiple tasks together. Research work in [28] was the first to address the need of MTL model for (i) disaster classification, (ii) informativeness, (iii) humanitarian categories, and (iv) damage severity assessment on a given input image. On the other hand, our previous work [6] is the first to propose a MTL model for joint disaster classification and victim detection, as shown in Figure 1. This eliminates the straightforward approach of running multiple separate DL models for each task, reducing the total latency with while preserving the accuracy of all tasks. This facilitates real-time disaster detection on the edge using any camera sensors in an IoT network.

2.4 NerveNet

NerveNet is a resilient network developed by Japan's National Institute of Information and Communications Technology (NICT). NerveNet is a specially developed network for the regional area to provide reliable network access and a stable, resilient information-sharing platform in emergencies, even if the base station is destroyed in a disaster. The base stations of NerveNet are interconnected by the Ethernet-based wired or wireless transmission systems such as satellite, Wi-Fi, LoRa and so on. They will form a mesh-topological network.

Nowadays, the current trend of the common network infrastructures uses the tree topology. As compared to it, NerveNet has the characteristic that is more tolerant to the faults such as node failures, disconnections, and destruction of the base station. Since the base station in the NerveNet supports basic services such as SIP proxy, DNS, and DHCP, the NerveNet can continuously provide connectivity services to the devices.

NerveNet has the feature of database synchronisation. It uses a hearsay daemon to synchronize the database of every node within the NerveNet network. Specifically, hearsay daemon synchronizes MySQL databases by updating the queries only and will not delete any actions when there is a lack of queries in another node's database. When the NerveNet node is connected to the NerveNet network, it will seek the difference in the table with other nodes.

After that, the database will be updated with the latest data. However, suppose all the NerveNet nodes are shut down. In that case, the data in the database will be deleted, and they cannot relieve the data back by using the hearsay daemon synchronisation since all the existing databases are empty.

3 AIOT IMPLEMENTATION

The aim of this section is to describe the basic hardware and software building blocks needed to establish an AIoT platform for disaster monitoring. The testbed consists of one Raspberry Pi 4 (RP4) serving as NerveNet monitoring node and five Intel next unit of computing (NUC) serving as NerveNet base station nodes. The testbed composition is depicted in Figure 2.

In default, NerveNet is configured to operate in the 172.16.0.0/16 network. The IP address of each node is 172.16.n.1 where n is the node id of the node defined during the network installation. For instance, the node in Figure 2a with a label of 208 has a NerveNet IP of 172.16.208.1.

The wireless links of the NerveNet are established using the Ethernet remote bridge (ERB) feature of NerveNet. To establish an ERB link between two nodes, one node must have a wireless interface configured as a wireless access point (AP) while the other node must have a wireless interface configured as a station (STA). Each ERB link is static and defined with a collection of configuration files included in the NerveNet distribution. To avoid the Wi-Fi interference, a different channel is assigned for each NerveNet link. Alfa wireless adapter is used to establish the NerveNet wireless links thanks to their superiority in long-range transmission. The module transmission power of RP4 and Intel NUC are set to 12 dBm and 15 dBm, respectively. All NerveNet nodes except the link between 210 and 204 are equipped with an omnidirectional antenna of 9 dBi. The positions of 210 and 204 are located at ground floor and 8th floor, respectively. Their connectivity is established based on a pair of 10 dB directional antenna.

Once the wireless mesh network is established, NerveNet Hearsay daemon can synchronize the data within the MySQL database based on the checksum of tables defined in the '/writable/etc/tables.d/' directory. In other words, any telemetric data from any node placed into this database will be automatically synchronized among all nodes. In this paper, we assume all the data stems from the NerveNet monitoring node, where RP4 is selected due to its high portability. Intel Neural Compute Stick 2 (NCS 2) is a dedicated hardware accelerator, and it is plugged into RP4 so that it can run deep neural network models optimized by Intel OpenVINO toolkit [18]. Here, we will run the multitask model [6] in the Intermediate Representation (IR) format. It can classify a total of seven categories: fire, hurricane, flood, earthquake, landslide, other disaster and not disaster.

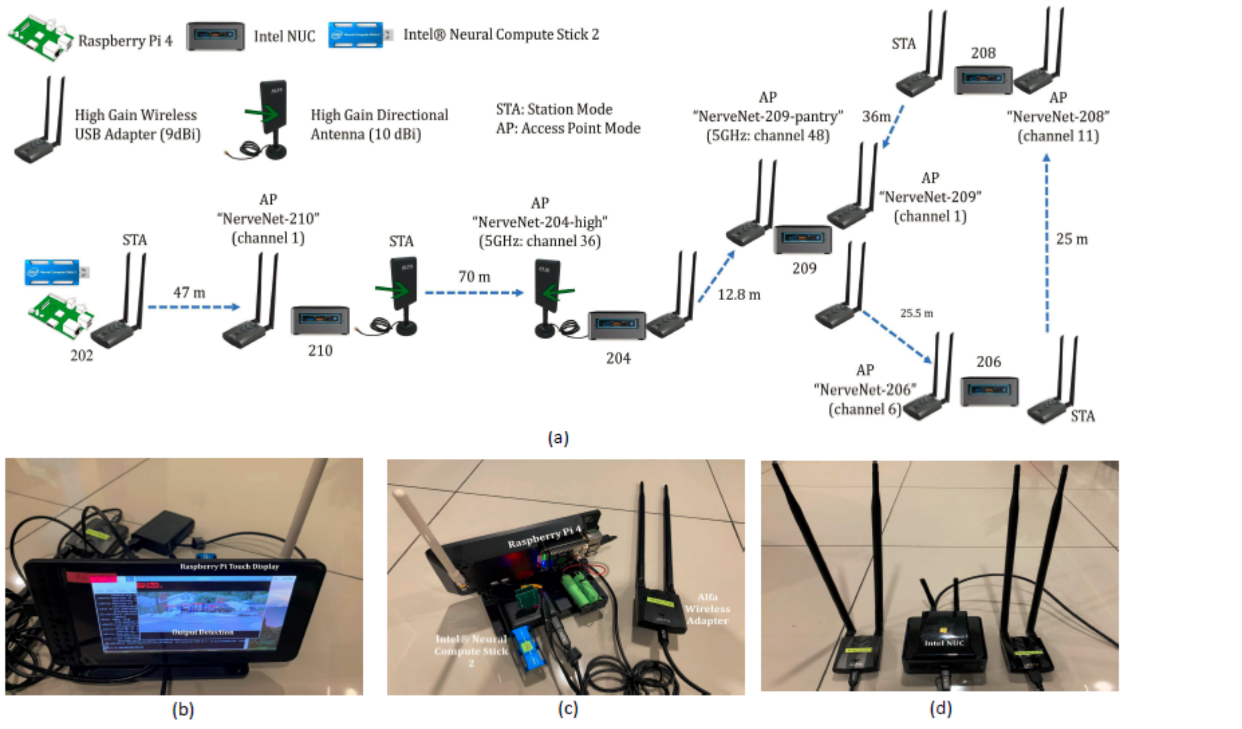

Raspberry Pi 4
Intel NUC
Intel® Neural Compute Stick 2
High Gain Wireless USB Adapter (9dBi)
High Gain Directional Antenna (10 dBi)
STA: Station Mode
AP: Access Point Mode
STA
208
AP "NerveNet-209-pantry" (5GHz: channel 48)
36m
AP "NerveNet-208" (channel 11)
AP "NerveNet-209" (channel 1)
25 m
AP "NerveNet-210" (channel 1)
AP "NerveNet-204-high" (5GHz: channel 36)
STA
47 m
STA
70 m
12.8 m
209
25.5 m
202
210
204
206
AP "NerveNet-206" (channel 6)
STA
(a)
Raspberry Pi Touch Display
Output Detection
Raspberry Pi 4
Alfa Wireless Adapter
Intel® Neural Compute Stick 2
Intel NUC
(b)
(c)
(d)

Figure 2: AIoT testbed implementation. (a) Testbed. (b) NerveNet monitoring node (front view). (c) NerveNet monitoring node (rear view). (d) NerveNet base

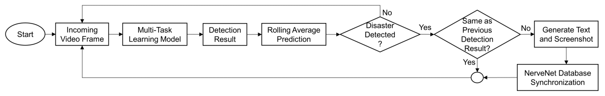

Start
Incoming Video Frame
Multi-Task Learning Model
Detection Result
Rolling Average Prediction
Disaster Detected ?
No
Yes
Same as Previous Detection Result?
No
Yes
Generate Text and Screenshot
NerveNet Database Synchronization

Figure 3: Working flow of AIoT monitoring.

4 PERFORMANCE EVALUATION

4.1 Disaster Monitoring

A video clip containing various types of disaster is used for the inference. Its duration and resolution are 38 s and 720 x 1072, respectively. There are a total of 14 disaster events from this video clip. Figure 4 displays one of the screenshots, where the FPS, disaster type and total victims are reported as 2.01, flood and 5, respectively. To demonstrate the ultra-low power consumption of our AIoT solution, we adopted an USB power meter for the measurement purpose. Figures 5a and 5b show the power consumption recorded during idle time and multitask execution time, respectively. From the figure, it can be observed that the AIoT model consumes around 1.23 W.

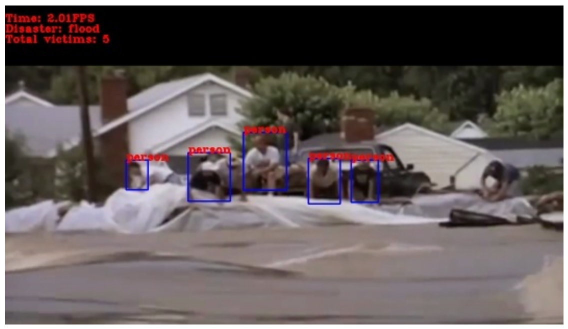


Figure 4: Detection screenshot.

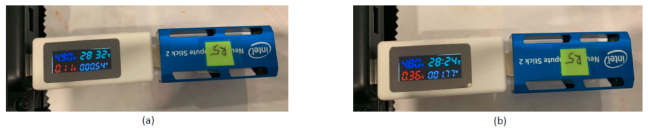
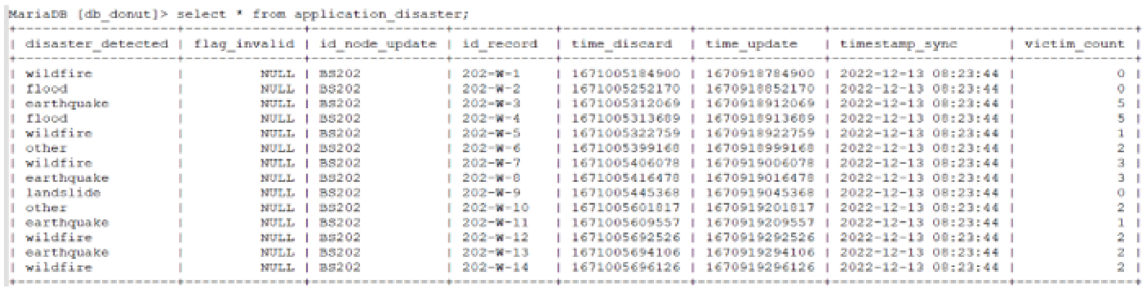


Figure 5: Power Measurement. (a) Idle time. (b) Execution time.

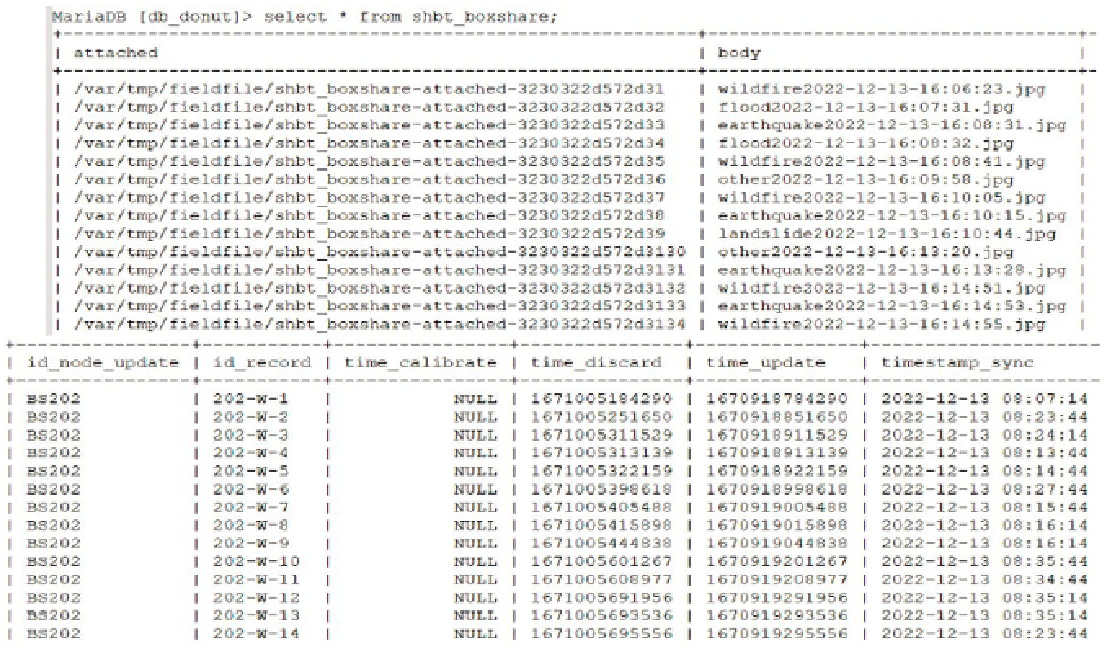
4.2 NerveNet Database Synchronization

Figure 6 displays the NerveNet database related to text and image synchronization. The columns “time update” and “timestamp_sync” indicate the time generated by the source and the time received by this specific node, respectively. By comparing the “timestamp_sync” of each NerveNet node, the synchronization latency can be calculated and plotted in Figure 7. From the figure, it can be observed that the closer the NerveNet node to source node 202 and first NerveNet base station node 210, the shorter the network latency.



```
MariaDB [db_donut]> select * from application_disaster;
+-------------------+--------------+----------------+------------+---------------+---------------+---------------------+--------------+
| disaster_detected | flag_invalid | id_node_update | id_record  | time_discard  | time_update   | timestamp_sync      | victim_count |
+-------------------+--------------+----------------+------------+---------------+---------------+---------------------+--------------+
| wildfire          |         NULL | BS202          | 202-W-1    | 1671005184900 | 1670918784900 | 2022-12-13 08:23:44 |            0 |
| flood             |         NULL | BS202          | 202-W-2    | 1671005252170 | 1670918852170 | 2022-12-13 08:23:44 |            0 |
| earthquake        |         NULL | BS202          | 202-W-3    | 1671005312069 | 1670918912069 | 2022-12-13 08:23:44 |            5 |
| flood             |         NULL | BS202          | 202-W-4    | 1671005313689 | 1670918913689 | 2022-12-13 08:23:44 |            5 |
| wildfire          |         NULL | BS202          | 202-W-5    | 1671005322759 | 1670918922759 | 2022-12-13 08:23:44 |            1 |
| other             |         NULL | BS202          | 202-W-6    | 1671005399168 | 1670918999168 | 2022-12-13 08:23:44 |            2 |
| wildfire          |         NULL | BS202          | 202-W-7    | 1671005406078 | 1670919006078 | 2022-12-13 08:23:44 |            3 |
| earthquake        |         NULL | BS202          | 202-W-8    | 1671005416478 | 1670919016478 | 2022-12-13 08:23:44 |            3 |
| landslide         |         NULL | BS202          | 202-W-9    | 1671005445368 | 1670919045368 | 2022-12-13 08:23:44 |            0 |
| other             |         NULL | BS202          | 202-W-10   | 1671005601817 | 1670919201817 | 2022-12-13 08:23:44 |            2 |
| earthquake        |         NULL | BS202          | 202-W-11   | 1671005609557 | 1670919209557 | 2022-12-13 08:23:44 |            1 |
| wildfire          |         NULL | BS202          | 202-W-12   | 1671005692526 | 1670919292526 | 2022-12-13 08:23:44 |            2 |
| earthquake        |         NULL | BS202          | 202-W-13   | 1671005694106 | 1670919294106 | 2022-12-13 08:23:44 |            2 |
| wildfire          |         NULL | BS202          | 202-W-14   | 1671005696126 | 1670919296126 | 2022-12-13 08:23:44 |            2 |
+-------------------+--------------+----------------+------------+---------------+---------------+---------------------+--------------+
```

(a)



```
MariaDB [db_donut]> select * from shbt_boxshare;
+-------------------------------------------------------------+-------------------------------------------+-
| attached                                                    | body                                      |
+-------------------------------------------------------------+-------------------------------------------+-
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d31    | wildfire2022-12-13-16:06:23.jpg           |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d32    | flood2022-12-13-16:07:31.jpg              |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d33    | earthquake2022-12-13-16:08:31.jpg         |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d34    | flood2022-12-13-16:08:32.jpg              |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d35    | wildfire2022-12-13-16:08:41.jpg           |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d36    | other2022-12-13-16:09:58.jpg              |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d37    | wildfire2022-12-13-16:10:05.jpg           |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d38    | earthquake2022-12-13-16:10:15.jpg         |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d39    | landslide2022-12-13-16:10:44.jpg          |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d3130  | other2022-12-13-16:13:20.jpg              |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d3131  | earthquake2022-12-13-16:13:28.jpg         |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d3132  | wildfire2022-12-13-16:14:51.jpg           |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d3133  | earthquake2022-12-13-16:14:53.jpg         |
| /var/tmp/fieldfile/shbt_boxshare-attached-3230322d572d3134  | wildfire2022-12-13-16:14:55.jpg           |
+----------------+-----------+----------------+---------------+---------------+---------------------
| id_node_update | id_record | time_calibrate | time_discard  | time_update   | timestamp_sync
+----------------+-----------+----------------+---------------+---------------+---------------------
| BS202          | 202-W-1   |           NULL | 1671005184290 | 1670918784290 | 2022-12-13 08:07:14
| BS202          | 202-W-2   |           NULL | 1671005251650 | 1670918851650 | 2022-12-13 08:23:44
| BS202          | 202-W-3   |           NULL | 1671005311529 | 1670918911529 | 2022-12-13 08:24:14
| BS202          | 202-W-4   |           NULL | 1671005313139 | 1670918913139 | 2022-12-13 08:13:44
| BS202          | 202-W-5   |           NULL | 1671005322159 | 1670918922159 | 2022-12-13 08:14:44
| BS202          | 202-W-6   |           NULL | 1671005398618 | 1670918998618 | 2022-12-13 08:27:44
| BS202          | 202-W-7   |           NULL | 1671005405488 | 1670919005488 | 2022-12-13 08:15:44
| BS202          | 202-W-8   |           NULL | 1671005415898 | 1670919015898 | 2022-12-13 08:16:14
| BS202          | 202-W-9   |           NULL | 1671005444838 | 1670919044838 | 2022-12-13 08:16:14
| BS202          | 202-W-10  |           NULL | 1671005601267 | 1670919201267 | 2022-12-13 08:35:44
| BS202          | 202-W-11  |           NULL | 1671005608977 | 1670919208977 | 2022-12-13 08:34:44
| BS202          | 202-W-12  |           NULL | 1671005691956 | 1670919291956 | 2022-12-13 08:35:14
| BS202          | 202-W-13  |           NULL | 1671005693536 | 1670919293536 | 2022-12-13 08:35:14
| BS202          | 202-W-14  |           NULL | 1671005695556 | 1670919295556 | 2022-12-13 08:23:44
```

(b)

Figure 6: Database synchronization. (a) Text. (b) Image.

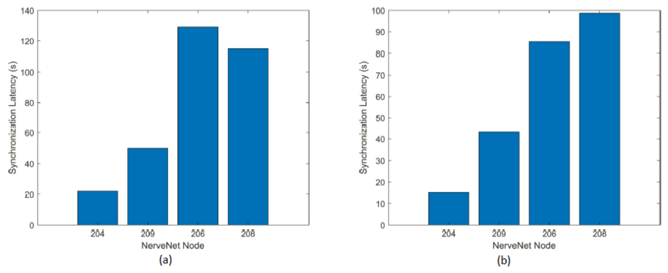


Figure 7: Synchronization Latency with respect to node 210. (a) Text. (b) Image.

Surprisingly, the latency for text synchronization is larger than that of the image synchronization. This is because both text and screenshots are pushed to the NerveNet database at the same time. In this case, NerveNet may attempt to synchronize the images first before the text. The size of total synchronized images is 647168 bytes. Nevertheless, both text and images can be synchronized within two minutes across a multi-hop Wi-Fi network.

5 CONCLUSIONS

In this paper, we have proposed a AIoT-based disaster monitoring using NerveNet wireless mesh network. To reduce the heavy workload of AI inference, we utilized OpenVINO to accelerate the process so that it can be executed on low-powered Raspberry Pi device. As for the data robustness, we invoked the feature of data synchronization to disseminate the data among NerveNet nodes. The effectiveness of the solution has been demonstrated via a testbed implementation. In future, we plan to test the framework in a LoRa based mesh network.

ACKNOWLEDGMENTS

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).

REFERENCES

[1] Rausch, T., Nastic, S. and Dustdar, S., 2018. EMMA: Distributed QoS-aware MQTT middleware for edge computing applications. Proceedings - 2018 IEEE International Conference on Cloud Engineering, IC2E 2018, pp.191– 197. https://doi.org/10.1109/IC2E.2018.00043.

[2] Chen, M., Li, W., Hao, Y., Qian, Y. and Humar, I., 2018. Edge cognitive computing based smart healthcare system. Future Generation Computer Systems, 86, pp.403–411. https://doi.org/10.1016/J.FUTURE.2018.03.054.

[3] Shi Q, Zhang Z, Yang Y, Shan X, Salam B, Lee C. Artificial Intelligence of Things (AIoT) Enabled Floor Monitoring System for Smart Home Applications. ACS Nano. 2021 Nov 23;15(11):18312-18326. doi: 10.1021/acsnano.1c07579. Epub 2021 Nov 1. PMID: 34723468.

[4] S. Basu, M. Karuppiah, K. Selvakumar, K. Li, S.H. Islam, M.M. Hassan, and M.Z. Bhuiyan, An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment. Future Gener. Comput. Syst., 88, 254-261. 2018.

[5] S. P. M. K. W. Ilukkumbure, V. Y. Samarasiri, M. F. Mohamed, V. Selvaratnam, and U. U. Samantha Rajapaksha, "Early Warning for Pre and Post Flood Risk Management by Using IoT and Machine Learning," *ICAC 2021 - 3rd International Conference on Advancements in Computing, Proceedings*, pp. 252–257, 2021, doi: 10.1109/ICAC54203.2021.9671141.

[6] Y. J. Wong, M. -L. Tham, B. -H. Kwan, E. M. A. Gnanamuthu and Y. Owada, "An Optimized Multi-Task Learning Model for Disaster Classification and Victim Detection in Federated Learning Environments," in IEEE Access, vol. 10, pp. 115930-115944, 2022, doi: 10.1109/ACCESS.2022.3218655.

[7] M. Inoue and Y. Owada, "NerveNet Architecture and Its Pilot Test in Shirahama for Resilient Social Infrastructure," IEICE Transactions on Communications, vol. E100–B, no. 9, pp. 1526–1537, 2017.

[8] S. A. Shah, D. Z. Seker, M. M. Rathore, S. Hameed, S. ben Yahia, and D. Draheim, "Towards Disaster Resilient Smart Cities: Can Internet of Things and Big Data Analytics Be the Game Changers?," IEEE Access, vol. 7, pp. 91885–91903, 2019, doi: 10.1109/ACCESS.2019.2928233.

[9] S. H. Alsamhi, O. Ma, M. S. Ansari, and F. A. Almalki, "Survey on collaborative smart drones and internet of things for improving smartness of smart cities," IEEE Access, vol. 7, pp. 128125–128152, 2019, doi: 10.1109/ACCESS.2019.2934998.

[10] N. Suri *et al.*, "Exploiting smart city IoT for disaster recovery operations," IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings, vol. 2018-January, pp. 458–463, May 2018, doi: 10.1109/WF-IOT.2018.8355117.

[11] F. S. Mousavi, S. Yousefi, H. Abghari, and A. Ghasemzadeh, "Design of an IoT-based Flood Early Detection System using Machine Learning," *26th International Computer Conference, Computer Society of Iran, CSICC 2021*, Mar. 2021, doi: 10.1109/CSICC52343.2021.9420594.

[12] M. Sultan Mahmud, M. S. Islam, and M. A. Rahman, "Smart Fire Detection System with Early Notifications Using Machine Learning," https://doi.org/10.1142/S1469026817500092, vol. 16, no. 2, May 2017, doi: 10.1142/S1469026817500092.

[13] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," *High-Confidence Computing*, vol. 1, no. 1, p. 100008, 2021, doi: https://doi.org/10.1016/j.hcc.2021.100008.

[14] C. Bucilua, R. Caruana, and A. Niculescu-Mizil, "Model Compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 535–541. doi: 10.1145/1150402.1150464.

[15] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *CoRR*, vol. abs/2006.0, 2020, [Online]. Available: https://arxiv.org/abs/2006.05525

[16] 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).

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

[18] R. S. Abhishek, A. Dhanus Kanth, S. Hariharan, S. Hariharasudhan, and S. Saravanan, "Design of Solar PV Emulator using Raspberry Pi Controller," *5th International Conference on Inventive Computation Technologies, ICICT 2022 - Proceedings*, pp. 581–584, 2022, doi: 10.1109/ICICT54344.2022.9850727.

[19] OpenVINO, "Install OpenVINO^{TM} Runtime for Raspbian OS — OpenVINO^{TM} documentation — Version(latest)," 2022. https://docs.openvino.ai/latest/openvino_docs_install_guides_installing_openvino_raspbian.html (accessed Dec. 10, 2022).

[20] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg, "AIDR: Artificial Intelligence for Disaster Response," in *Proceedings of the 23rd International Conference on World Wide Web*, 2014, pp. 159–162. doi: 10.1145/2567948.2577034.

[21] H. Mouzannar, Y. Rizk, and M. Awad, "Damage Identification in Social Media Posts Using Multimodal Deep Learning," *Proceedings of the International ISCRAM Conference*, vol. 2018-May, no. May, pp. 529–543, 2018.

[22] D. T. Nguyen, F. Ofli, M. Imran, and P. Mitra, "Damage Assessment from Social Media Imagery Data During Disasters," in *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2017, pp. 569–576.

[23] F. Alam, F. Ofli, and M. Imran, "CrisisMMD: Multimodal twitter datasets from natural disasters," in *12th International AAAI Conference 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

[24] 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 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020, pp. 151–158. doi: 10.1109/ASONAM49781.2020.9381294.

[25] M. I. Perdana, A. Risnumawan, and I. A. Sulistijono, "Automatic Aerial Victim Detection on Low-Cost Thermal Camera Using Convolutional Neural Network," *2020 International Symposium on Community-Centric Systems, CcS 2020*, Sep. 2020, doi: 10.1109/CCS49175.2020.9231433.

[26] C. Wen, H. Zhang, H. Li, H. Li, J. Chen, H. Guo, and S. Cheng, "Multiscene 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.

[27] 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.

[28] 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.

Flood Forecasting using Edge AI and LoRa Mesh Network

Mau-Luen Tham
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
thamml@utar.edu.my

Xin Hao Ng
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
penguinng0928@gmail.com

Rong-Chuan Leong
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
lrongchuan@1utar.my

Yasunori Owada
Resilient ICT Research Center
National Institute of Information and Communications Technology (NICT)
Tokyo, Japan
yowada@nict.go.jp

Abstract—**Remote flood forecasting has exponentially grown over the past decade together with the unprecedented expansion of Internet of Things (IoT) network. This is feasible with the use of long range wireless communication technology such as LoRa. Ideally, each LoRa device shall process the sensor data locally and trigger warnings to the remote server based on prediction results. However, conventional prediction methods rely on highly computational artificial intelligence (AI) algorithms, which are not suitable for low-powered LoRa network. In this paper, the LoRa device is integrated with an edge AI model, which is based on long short-term memory (LSTM) neural network. OpenVINO is adopted to optimize the LSTM model, before executing the solution on a Raspberry Pi 4 in combination with Intel Movidius Neural Computing Stick 2 (NCS2). Experimental results demonstrate the feasibility of deployment of the customized model on low-cost and power-efficient embedded hardware.**

Keywords—Edge AI, LSTM, Flood Forecasting, LoRa Mesh Network, IoT

I. INTRODUCTION

Flood forecasting models have been researched in the hydrological engineering area for many years. Recently, there has been increased research interest in river flood prediction and modelling, defined as data-driven approaches. The artificial neural network (ANN) model is the most famous usual data-driven approach. Most conventional statistical methods require a lot of data for their models, and they can generate no assumptions for both linear and non-linear systems. Hence, the data-driven approach, ANN, is an alternative to hydrological flood forecasting instead of the existing methods [1].

Artificial intelligence (AI) has made essential development in modelling hydrological forecasting and dynamic hydrological issues. With the advancement of information technology, the application of ANN models in many aspects of science and engineering is increasingly becoming common due to its simplicity of structure. Diverse neural network modelling approaches have been applied, like implementing the model approaches individually or combining process-based approaches to minimize mistakes and increase the models' forecasting accuracy. The study in [2] applied AI model to forecast river flow for 15 years starting from 2000.

However, there are some limitations of the ANN model. One of them is lacking understanding of watershed processes. Furthermore, the limitation of memory in calculating sequential data exposes the disadvantages of the ANN model. The breakthrough in computational science has recently increased the interest in deep neural network (DNN) approaches. In addition, the most recent DNN applications, such as the long short-term memory (LSTM) [3] and gated recurrent unit (GRU) [4] neural networks, have been efficiently implemented in diverse areas and fields, such as time sequence problems. Those models can apply to machine translation, speech recognition, tourism field, language modelling, rainfall-runoff simulation, stock prediction and river flow forecasting.

On 11th March 2011, around 29000 cellular towers were damaged in the East Japan Great Earthquake. These damages have restricted the broadcast of evacuation notices and the collection of historical information for disaster forecasting. Hence, it can be known that the resilience of a network remains an open issue in the deployment of the fault-tolerant network during an emergency disaster. Fortunately, a disaster-resilient mesh-topological network called NerveNet was developed by Japan NICT. Each NerveNet node is independent and tolerant to system failure and link disconnection due to its mesh structure.

In this paper, a flood forecasting model is proposed. In the study area, rainfall and river water levels collected at hydrological stations serve as dataset for the training and testing process of the AI models. Then, the forecasted flood water level will be processed to generate the flood warning message. It will be sent through the NerveNet LoRa mesh network. Note that the proposed solution 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".

The rest of the paper is organized as follows. Section II discusses the related works. Section III describes the system architecture. Section IV presents the experimental results and discussions. Section V concludes the article.

II. Related Work

A. Edge AI

Several existing works [5]–[6] explored the potential of edge AI for various applications. The authors in [5] focused on real-time apple detection with the implementation of YOLOv3-tiny algorithm on various embedded platforms. However, they did not consider the communication aspects. Recognizing the importance of LoRa, the authors in [6] proposed an edge AI in LoRa-based fall detection system with fog computing and LSTM. The processing burden is placed on an LoRa-based edge gateway, where the collected sensor information is transmitted from an edge node via Bluetooth Low Energy (BLE). Differently, our solution integrates both edge AI and LoRa functionalities into one single device, which simplifies the deployment effort.

B. NerveNet

NerveNet is a resilient network developed by Japan National Institute of Information and Communications Technology (NICT) [7]. NerveNet is a specially developed network for the regional area to provide reliable network access and a stable, resilient information-sharing platform in emergencies, even if the base station is destroyed in a disaster. The base stations of NerveNet are interconnected by the Ethernet-based wired or wireless transmission systems such as satellite, Wi-Fi, LoRa and so on. They will form a mesh-topological network.

Nowadays, the current trend of the common network infrastructures uses the tree topology. As compared to it, NerveNet has the characteristic that it is more tolerant to the faults such as node failures, disconnections, destruction of the base station and so on. Since the base station in the NerveNet supports basic services such as SIP proxy, DNS, DHCP, the NerveNet can also continuously provide connectivity services to the devices.

III. SYSTEM ARCHITECTURE

A. Dataset

The dataset we employ is the Abashiri River watershed [8], located northeast of Hokkaido, Japan. The area of the watershed is around 1380 km^2. It has a 115 km long main river to the North Pacific and a range of elevation from 0 m to 978 m [9]. All AI models are trained and tested using the datasets observed at the downstream stations called 'Hongou'. The used datasets are hourly datasets with the water level and rainfall variables from 1st January 2019 to 31st December 2020.

During data pre-processing, the rainfall and water level data undergo a train-test split, separated into 70 % of the data as training dataset and 30 % as a testing dataset, as listed in Table I. The training data calculates the training process error and estimates the AI models' parameters. The testing data provides an independent performance evaluation of the AI models after training.

Next, the hydrological dataset has also undergone data standardisation where the values' distribution is rescaled to a mean value of 0 and a standard deviation value of 1. Data scaling is essential to fasten the training process of the AI model because the AI models can converge more rapidly if the dataset features are closer to the normal distribution. Prior to the AI model training, the time series dataset is converted into sequential data with 24-time steps as the sequence length. The model performs equally well when the sequence length is between five to 15 or more. Therefore, in this paper, the sequence length value of 24 is used in the model to represent 24 hours in one day.

TABLE I. TRAINING AND TESTING PERIOD FOR THE DATASET

Dataset	Training	Test
Hongou (Jan 2019 to Dec 2020)	Jan 2019 to May 2020	Jun 2020 to Dec 2020

B. AI Model Training in Google Colab

In this paper, four types of AI models, namely Random Forest, SVM, LSTM and GRU, are trained and tested on the dataset to benchmark the performance of the system in terms of flood water level forecasting. Trained in in Google Colab platform, the best AI model will be selected as the edge AI.

For Random Forest, the parameter 'max_depth' represents each tree's depth in the forest. Here, we set the max_depth value to 2. There are several hyperparameters in the LSTM model-building process. Firstly, the optimisation algorithm is the stochastic gradient descent procedure's extension to update the weights iterative of the network according to the training dataset. Secondly, an epoch is defined as the whole dataset transferring forward and backwards across the model's neural network once. Thirdly, the batch size is the number of samples propagating throughout the entire neural network. Table II demonstrates the hyperparameter settings of the LSTM model. For fair comparison, the same hyperparameters are adopted to train the GRU models.

C. AI Model Optimization using OpenVINO

The immediate output format of the LSTM model is .h5, which will be converted to pb format. The intention is to utilize the OpenVINO toolkit [10], which enables the faster running of the AI model in edge device. There are two main components in the OpenVINO toolkit, which are the model optimiser and inference engine. Firstly, when the trained model in pb format is fed into the model optimiser, it converts them to the IR format. At the same time, it optimises the performance, space, and hardware-agnostic with conservative topology transformations. The outputs of the model optimiser are .xml and .bin.

Secondly, the AI inferencing process is performed at the edge device by setting the inference engine to Intel Neural Compute Stick 2 (NCS2), which is a hardware accelerator. Before feeding to the inference engine, the data is scaled using the scaler.gz exported from the training process. The scaled data is then reframed. The historical time series data representing the last 24 hours is extracted from the scaled dataset by retrieving the top 24 values of the rainfall and water level data. After that, the sequence data and the trained model in IR format are fed into the inference engine to generate the water levels ahead of 1 hour in text form and the result graph in image form.

TABLE II. HYPERPARAMETER SETTINGS FOR LSTM MODEL.

Hyperparameter	Value
Sequence Length	24
Optimisation Algorithm	Root Mean Squared Propagation
Epoch	50
Batch Size	64

D. Evaluation Metrics

The mean absolute error (MAE) is the mean of the differences between the original value with the forecasted value. On an excellent flood forecast, the MAE should be smaller. Mathematically, it can be expressed as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{1}$$

The mean absolute percentage error (MAPE) is the percentage of the mean of the total error. On an excellent flood forecast, the MAPE should be smaller. It is written as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{e_i}{y_i}\right| \times 100 \tag{2}$$

The root mean squared error (RMSE) is the square root of the mean of the squared deviation of the forecasted flood water level value. On an excellent flood forecast, the RMSE should be smaller. It is written as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} {e_i}^2} \tag{3}$$

R^2 is the coefficient of determination and goodness of fit. With an excellent flood forecast, the R^2 should be larger.

$$R^2 = 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}} \tag{4}$$

The NerveNet LoRa data transmission performance is evaluated by calculating the packet delivery ratio (PDR) of LoRa packets.

$$\text{PDR} = \frac{\text{number of packets received}}{\text{number of packets sent}} \tag{5}$$

IV. Results and Discussions

Table III compares the water level forecasting performance of the aforementioned five AI model types on the testing dataset. Theoretically, the deep learning methods outperform the conventional machine learning methods when the big data comes into its input. This is consistent with the result, where the LSTM and GRU models have a lower value of MAE, MAPE and RMSE than the Random Forest and SVM models. This indicates that the deep learning models have a lower deviation of the forecasted results from the ground truth and a lower error percentage. A higher R^2 value indicates a more excellent time series forecasting performance from the deep learning models.

From the table, it can be observed that the LSTM model has more excellent performance than the GRU model since it has lower MAE, MAPE, RMSE and higher R^2. This finding is consistent with the findings in [11], where the LSTM model performs better than the GRU model in the case of short text processing and large-size datasets. In this paper, there is a huge amount of rainfall and water level dataset where both types of variables are short integers. They act as the inputs to the LSTM and GRU models. Therefore, it can be seen that the LSTM is more appropriate than the GRU models in these scenarios.

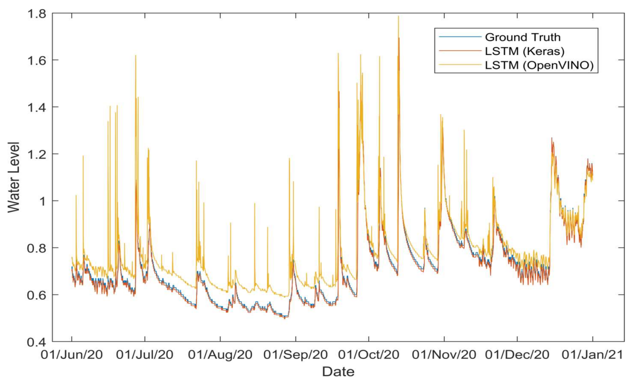
All in all, the LSTM has the best performance in the AI water level forecasting since it has the lowest MAE, MAPE and RMSE while the highest R^2 among all the proposed AI models. Therefore, LSTM is chosen as the AI water level forecasting model. Specifically, OpenVINO is used to convert the .h5 model to .xml and .bin format. It can be seen that there is a performance degradation of the converted model in all aspects.

Fig. 1(a) display the prediction versus ground truth for test dataset by using LSTM variations. As expected, the prediction using Keras model is close to the actual values. To reveal more insights, Fig. 1(b) compares the inference time between these two LSTM models. It can be seen that the LSTM (OpenVINO) is 28x slower than the Keras version. The reason is that the Keras model was using the Intel® Xeon® CPU @ 2.20Ghz provided by the Google Colab. This hardware has more computational power than the NCS2, which consumes only around 1.5W.

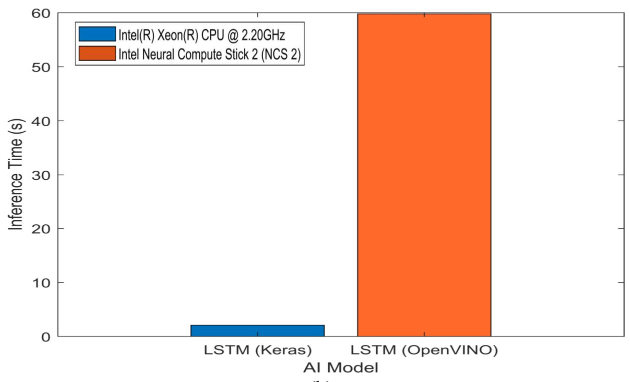
Fig. 2 shows the actual deployment of LoRa nodes. For the LoRa parameters, we adopted spreading factor of 12, transmission power of 20 mW, and bandwidth of 500 kHz. Three NerveNet LoRa nodes serve as MQTT subscriber whereas one NerveNet LoRa node acts as MQTT publisher.

TABLE III. Benchmarking Performance for Prediction

AI Model	MAE	RMSE	MAPE	R^2
Random Forest	0.0656	0.078	0.0972	0.7807
SVM	0.0541	0.0632	0.0763	0.8562
GRU	0.0138	0.0154	0.0217	0.9915
LSTM (Keras)	**0.0088**	**0.0092**	**0.0126**	**0.997**
LSTM (OpenVINO)	0.0593	0.0907	0.0899	0.704



(a)



(b)

Fig. 1 LSTM performance benchmarking. (a) Prediction vs ground truth. (b) Inference time.

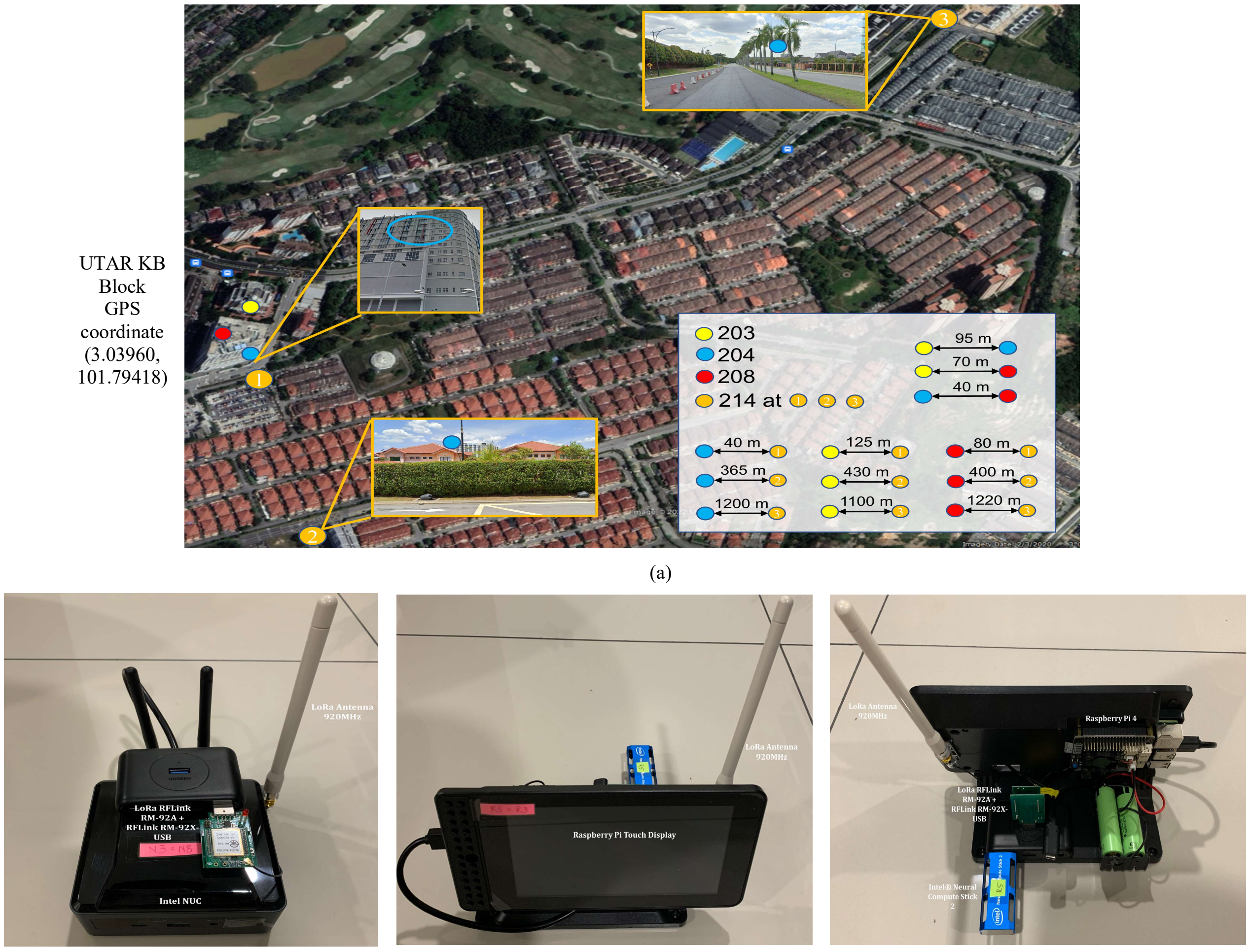

UTAR KB Block GPS coordinate (3.03960, 101.79418)
203
204
208
214 at 1 2 3
95 m
70 m
40 m
40 m
125 m
80 m
365 m
430 m
400 m
1200 m
1100 m
1220 m
(a)
LoRa Antenna 920MHz
LoRa RFLink RM-92A + RFLink RM-92X-USB
Intel NUC
Raspberry Pi Touch Display
Raspberry Pi 4
Intel® Neural Compute Stick 2
(b)
(c)
(d)

Fig. 2 System deployment. (a) The location of three subscriber nodes (203,204 and 208) and one publisher node (214). (b) Subscriber node (Intel NUC). (c) Publisher node (front view). (d) Publisher node (rear view).

The publisher publishes the MQTT message at three different locations. At each location, a total of 11 LoRaMesh packets are transmitted. The quality of service (QoS) level is set to zero, which guarantees best-effort message delivery. In other words, the publisher only transmits each packet once and LoRa message packets may be lost during the transmission process. Node 208 is located inside the building in such a way that nodes 203 and 204 can act as relay node. We implement subscriber and publisher nodes using Intel next unit computing (NUC) and Raspberry Pi 4, respectively. The latter is chosen due to its high portability and low cost, which is suitable for massive deployment of flood monitoring.

Fig. 3 depicts the overall performance of NerveNet LoRaMesh. It can be observed from Fig. 3a, only extra hops are needed at location 3. This is reasonable since the distance between 204/208 and location 3 is at least 1200 m. In this case, node 203 which is closer to location 3 acts as relay node. For LoRaMesh packet to arrive at node 208, the packet initially sent by node 214 at location 3 is passed to 203, through 204 to 208. For other two locations, only one hop transmission is needed. This is because there are less obstacles, such as trees and buildings. The multi-hop transmission is affected by the received signal strength indicator (RSSI), as reported in Fig. 3b. All RSSI values are measured with respect to the publisher node 214, except the last two columns. Specially, 204 and 208 measurements are based on their relay nodes 203 and 204, respectively.

All LoRaMesh packets are received when the publisher transmits messages at locations 1 and 2. For location 3, two out of 11 packets are lost during the transmission for nodes 204 and 208. Specifically, when node 204 does not receive the packets from 203, it could not forward them to 208. Fig.3 d compares the time on air. In LoRaMesh, time on air defines the elapsed time on air for a LoRaMesh packet between publisher and subscriber. As expected, the further the distance, the longer time needed to transmit the LoRaMesh packets.

V. Conclusion

In this paper, we have proposed an edge AI solution that forecasts flood water level and transmits the packet via LoRa mesh network. the AI model training and the testing dataset are obtained from Japan's organisation. Hence, the AI results may not apply to the local area since the weather, season, humidity, and geographical condition of Malaysia are different from Japan. The local dataset can be requested from the local government to build an AI model that can fit the situation in Malaysia's local area so that a better understanding of the feasibility of the AI model in disaster detection in Malaysia.

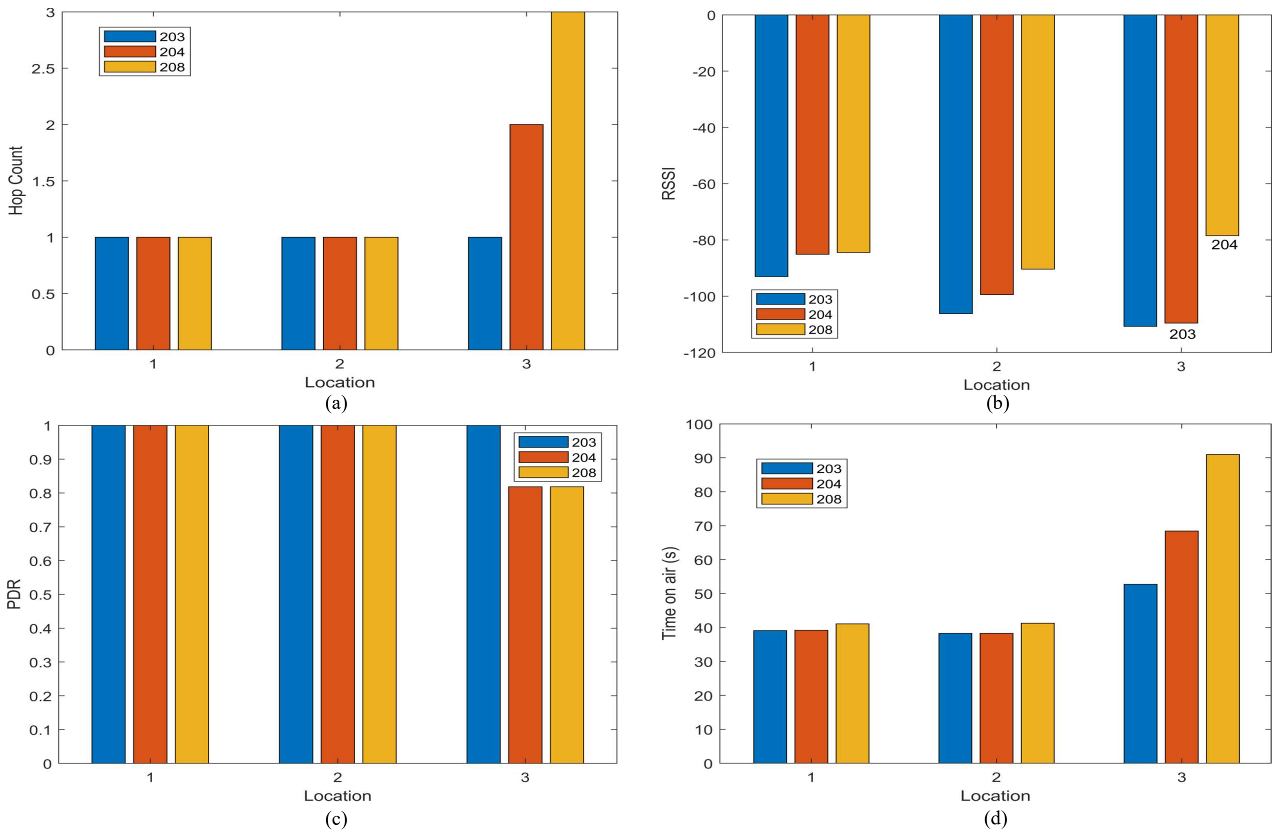


Fig. 3 Performance of NerveNet LoRaMesh. (a) Hop count. (b) RSSI. (c) PDR. (d) Time on air.

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] O. A. Kişi, "A combined generalized regression neural network wavelet model for monthly streamflow prediction," KSCE J Civ Eng, vol. 15, pp. 1469–1479, 2011.

[2] Z. M. Yaseen et al., "Artificial intelligence based models for stream-flow forecasting: 2000–2015," *Journal of Hydrology*, vol. 530, pp. 829–844, 2015.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735‐1780, 1997.

[4] K. Cho, B. Van Merri¨enboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," arXiv preprint arXiv:1409.1259, 2014.

[5] V. Mazzia, A. Khaliq, F. Salvetti and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: An edge ai application", IEEE Access, vol. 8, pp. 9102-9114, 2020.

[6] J. P. Queralta, T. N. Gia, H. Tenhunen and T. Westerlund, "Edge-AI in LoRa-based health monitoring: Fall detection system with fog computing and LSTM recurrent neural networks", Proc. Int. Conf. Telecommun. Signal Process. (TSP), pp. 601-604, Jul. 2019.

[7] M. Inoue and Y. Owada, "NerveNet Architecture and Its Pilot Test in Shirahama for Resilient Social Infrastructure," IEICE Transactions on Communications, vol. E100–B, no. 9, pp. 1526–1537, 2017

[8] Ministry of Land, Infrastructure, Transport, and Tourism in Japan (MLIT Japan). Hydrology and Water Quality Database. Available online: http://www1.river.go.jp/ (accessed on 30 July 2022).

[9] N. Kimura et al., "Convolutional Neural Network Coupled with a Transfer-Learning Approach for Time-Series Flood Predictions," Water, vol. 12, no. 96, 2020.

[10] OpenVINO Toolkit n.d., accessed 1 July 2022, <https://software.intel.com/enus/openvinotoolkit>.

[11] S. Yang, X. Yu and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example", 2020 International workshop on electronic communication and artificial intelligence (IWECAI), pp. 98-101, 2020.

Performance Study of Disaster-Resilient Mesh Networking using NerveNet Wi-Fi and LoRa

Mau-Luen Tham
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
thamml@utar.edu.my

Chee Hong Lean
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
lean2051999@gmail.com

Wei-Sean Lim
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
seanlimweisean@gmail.com

Rong-Chuan Leong
Department of Electrical and Electronic Engineering
Universiti Tunku Abdul Rahman
Kajang, Malaysia
lrongchuan@1utar.my

Yasunori Owada
Resilient ICT Research Center
National Institute of Information and Communications Technology (NICT)
Tokyo, Japan
yowada@nict.go.jp

Myint Myint Sein
Geographical Information System
University of Computer Studies, Yangon (UCSY),
Myanmar
myint@ucsy.edu.mm

2023 6th Conference on Cloud and Internet of Things (CIoT) | 979-8-3503-9669-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/CIoT57267.2023.10084890

Abstract— **When a natural disaster event happens, it could trigger regional cellular network outages and hence disable network communication within the affected area. If a resilient network is implemented, alert messages with sufficient information can be sent over the Internet to provide a nationwide response. Japan National Institute of Information and Communication Technology (NICT) has invented a resilient network framework called NerveNet, which supports robust communications via mesh networking. Using their technology as the communication platform, disaster nodes could be installed at disaster hotspots to send out disaster information or even provide lightweight Internet services. NerveNet can support data synchronization using Wi-Fi and LoRa. The former is used to provide wide bandwidth but low-range data transmission, whereas the latter enables narrow-bandwidth data transmission in coverage of kilometers, which is suitable for crucial or emergency disaster data updates.**

Keywords—wireless mesh network, disaster resilient network, LoRa, database synchronization

I. Introduction

According to [1], the total occurrence of global natural disasters in 2021 is 13 % higher than the average over the last 30 years. Malaysia, despite its geographically stable region, is facing similar disasters such as tsunami, floods, drought and earthquake. In fact, Malaysia had experienced 51 natural disaster events from the year 1998 to 2018, causing 281 people to die and more than 3 million people were affected, which caused around RM8 billion in damages [2]. Flood is the most common natural disaster in Malaysia, resulting in total residential and commercial damage of RM455 million and RM142 million, respectively [3].

To raise the public awareness of the emergency situations, an efficient communication network should be robust against infrastructure damage during disasters. Using the existing cellular communication service, the alert packets are generally transmitted to the remote cloud via base stations (BSs), which are vulnerable to disaster damage. Japan, a country with a high natural disaster rate, has been using a resilient mesh network named NerveNet to overcome this challenge. NerveNet has been developed by the National Institute of Information and Communications Technology (NICT) in Japan since the year 2006. Its resiliency was demonstrated by conducting a large scale of testbed with 30 BSs constructed within Tohoku University in Sendai in 2011 [4]. Later in 2014, NICT started the real deployment of NerveNet for disaster prevention purposes. The advantage is that the end devices do not rely on the availability of each other. When one NerveNet node goes inactive, it does not affect the overall service provided as other nodes will self-configure a new pathway to transfer data. Logically, any node can peer with any other nodes if they are under NerveNet network, which gives it fault-tolerance property during disaster events.

In this paper, we design and implement a disaster-resilient mesh networking using NerveNet Wi-Fi and LoRa. The former is featured with short-range connectivity and high data rates whereas the latter offers long-range connectivity for low data rate applications. These two wireless technologies can complement each other to expand the Internet of things (IoT) connectivity and provide nation-wide coverage. By invoking the NerveNet Hearsay daemon, collected sensor data such as alert text and images can be wirelessly synchronized in multiple NerveNet nodes' database. This is one of goals of the ASEAN IVO project titled “Context-Aware Disaster Mitigation using Mobile Edge Computing and Wireless Mesh Network”.

The rest of the paper is organized as follows. Section II discusses the related works. Section III describes the proposed solution. Section IV presents the experimental results and discussions. Section V concludes the article.

II. Related Work

In [5], the authors proposed an architecture for a drone-based communication infrastructure for disaster response. The formed wireless mesh network, however, relies on the Wi-Fi technology, which poses transmission range limitations. Another scheme was presented in [6], where the authors developed a synchronous content distribution system via Wi-Fi mesh networking. In an effort to extend the IoT coverage, the authors in [7] implemented a Device-to-Device (D2D) based LoRaMAC solution to disseminate the data. However, the packet transmission scenario considers generic data, not the multimedia data such as text and image.

In [8], the work analyzed the performance of unmanned aerial vehicle (UAV)-enable LoRa networks for disaster management applications, from the perspective of ns-3 simulation. Recognizing the complementary benefits of Wi-Fi and LoRa, the work in [9] designed a hybrid Wi-Fi LoRa ad-hoc network which leverages smartphones and IoT devices as nodes in a mesh. However, their data distribution focuses on plain text string. In contrast, our work focuses on both text and image synchronization among NerveNet nodes. This facilitates crowdsourcing during disaster events, where response team can extract more useful insights from image for critical decision-making.

III. Proposed Solution

The overall system architecture consisting of hardware and software is depicted in Fig. 1. There is a total of six NerveNet nodes. Nodes BS203, BS204 and BS205 are running x86 NerveNet OS in Ubuntu 18.04 of Intel NUC whereas nodes BS206, BS207 and BS208 are executing armhf NerveNet OS in Raspbian Buster of Raspberry Pi. Node BS203 concurrently serves as the NerveNet web application.

A. NerveNet Wireless Mesh Network

Usually, a mesh network is simply adding a redundant connection for each device within the network topology, then the device will look up for an alternative pathway to reach its destination if its primary peer is down. NerveNet Wi-Fi mesh network framework not only provides the function to look up every single mesh node in the network, but also adds database synchronization to share common data within the mesh network. The lookup feature is built by using a service daemon called Path Tree Management Generation (PTMGR), which is installed in the essential node within the mesh network. PTMGR continuously seeks for peers' network status to identify if any node is down or new node has joined the network. If the connection between nodes is steadily maintained by PTMGR, the nodes could directly connect or access to each other and perform NerveNet SQL database synchronization. The nodes will compare the data rows within each other to update with the latest data.

NerveNet also supports LoRa mesh network with the use of specific LoRa equipment. NerveNet LoRa uses a frequency band of 920 MHz for all LoRa nodes. To overcome the potential LoRa signal interference, NerveNet LoRa uses time division multiple access (TDMA). Each node synchronizes the time from the global positioning system (GPS) receiver to other nodes in such as way that the node will transmit LoRa data within the period of the pre-configured time slot. For example, 10 seconds are divided into five slots to form a cycle, the time slots are allocated to five nodes, thus each node will transmit LoRa signal during its time slot only. With properly configured time slots and channels to reduce disturbance, NerveNet LoRa nodes are possible to communicate at the speed of 100 Bytes per second over several kilometers distance with a power consumption of only tens of milliwatt.

B. Wi-Fi Mesh Network

To install Wi-Fi mesh network, each device is initially configured with NerveNet IP address, access point (AP) interface and Wi-Fi Protected Access (WPA) client interface. These interfaces form Ethernet remote bridge (ERB) tunnel link, which is used by the PTMGR to enable each node to communicate. Both x86 and armhf versions of NerveNet OS are running Docker containers.

C. LoRa Mesh Network

To install LoRa mesh network, we use RFlink-RM92A as the LoRa module, parameters of which are set as follows: RF-channel of 41, RF-bandwidth of 500 kHz, and spreading factor of SF12. The TDMA is set to four slots per minute, granting each LoRa node (BS203, BS206, BS207, and BS208) around 15 seconds duration. To transmit plain text data, NerveNet LoRa node uses MQTT service to buffer published data, then sends out the LoRa data within the allocated time slot with its best effort. The LoRa data will remain in the MQTT buffer and waits for the next cycle if the attempt to transmit fails. However, NerveNet LoRa MQTT communication uses QoS level zero, LoRa packet loss is still possible. Similar as Wi-Fi, both x86 and armhf versions of NerveNet OS are running Docker containers.

D. NerveNet Monitoring Dashboard (NerveDASH)

NerveDASH is a web application for visualizing the results of disaster detection. The main components to support NerveDASH are Neo4j, MQTT service, HTTP server, REST API, WebSocket API, and Nginx server, as shown in Fig. 2.

When NerveNet gateway sends a message, it will first be handled by MQTT client for JSON encodable data. If the data is media (image, video), it will be sent to HTTP server instead of MQTT broker. When the media storage has exceeded its limit, the oldest stored media will be replaced by the latest media data. Both MQTT and HTTP service point towards Neo4j server, a graphical-based NoSQL database server. By default, MQTT broker will set a count-down timer to receive

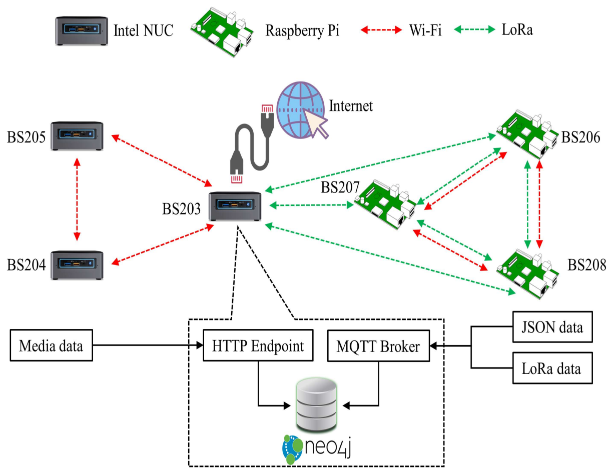


Fig. 1 NerveNet network architecture.

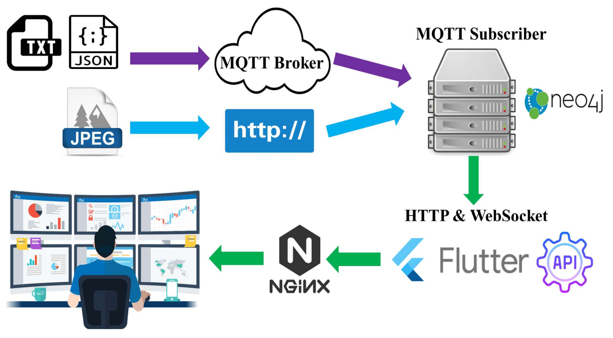


Fig. 2 NerveDASH.

a "heartbeat" message from NerveNet gateway (publisher). If the message is not received in 20 seconds interval, an inactive message will be sent to Neo4j (subscriber) to record that the respective node is down until a new "heartbeat" message is received.

The data in Neo4j can then be retrieved via a RESTful API or a WebSocket API. The RESTful API is used for simple text retrieval, while WebSocket API is used for file streaming. Finally, the Nginx static file hosting feature is used to serve the frontend static files of the web application. The Nginx server can also be configured to provide load balancing for all HTTP endpoints if needed.

E. Evaluation Metrics

The network latency within NerveNet Wi-Fi domain is benchmarked using the ping command. We consider two scenarios: (1) one broken mesh link (2) no broken mesh link. Both TCP and UDP throughput are measured using the popular tool Iperf3 [10]. The former uses settings of 100 packets and the latter configures the sender bandwidth and duration to be 50 Mbps and 10 s, respectively. Clearly, packet retransmission is only possible at the TCP scenario. When measuring UDP throughput, we also record the jitter. With respect to database synchronization, the average time taken to synchronize images of different resolutions is recorded. The metadata of images is tabulated in Table I.

IV. Results and Discussions

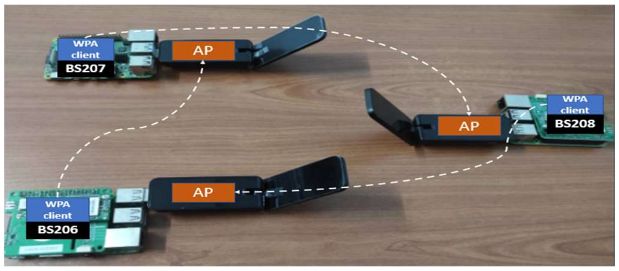
The NerveNet Wi-Fi mesh network performance within x86 (BS203, BS204, BS205) and armhf (BS206, BS207, BS208) are evaluated. Fig. 3 (a) and (b) show the Wi-Fi links for Intel NUC and Raspberry Pi, respectively. For the NerveNet LoRa mesh testbed as shown in Fig. 3(c), any device within the LoRa network could perform LoRa MQTT data exchange with each other. The NerveNet LoRa mesh MQTT messaging performance is carried out between BS203 and BS207.

Fig. 4 evaluates the TCP and UDP throughput for P2P and mesh links in NerveNet x86 Wi-Fi. To activate P2P link, we shutdown one out of three NerveNet nodes within the Wi-Fi domain. From the figure, it can be observed that P2P link has generally higher TCP and UDP throughput as compared with mesh-link. This is because the sender does not need to calculate a pathway to transfer the data. Also, when the route direction is from Wi-Fi client interface to Wi-Fi AP interface of peer device, the TCP throughput is also higher as compared with the reverse ordered route direction.

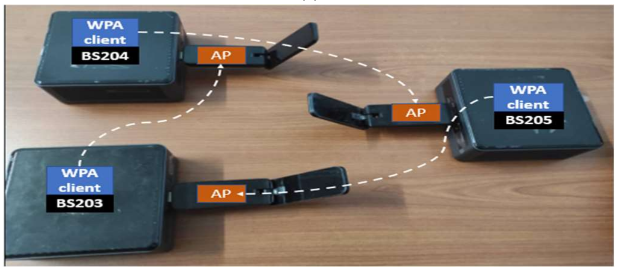
Fig. 4 (c) displays the jitter of NerveNet Wi-Fi within x86 domain in the cases of P2P and mesh links are more or less similar. The difference between the highest and lowest jitter is less than two milliseconds. According to [11], the QoS requirements of jitter for video conferencing is less than 30 ms. Therefore, NerveNet Wi-Fi within triangular x86 nodes

TABLE I. Image Resolution and Size for Database Synchronization Test

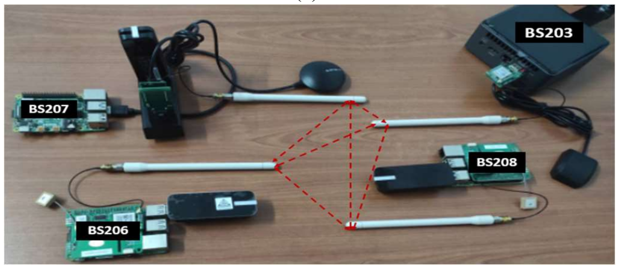
Image resolution (label)	Image Size
960 x 540 (low)	264.8 kB
1920 x 1080 (moderate)	909.7 kB
3554 x 1999 (high)	2.5 MB
9600 x 6800 (ultra high)	10.9 MB



(a)



(b)



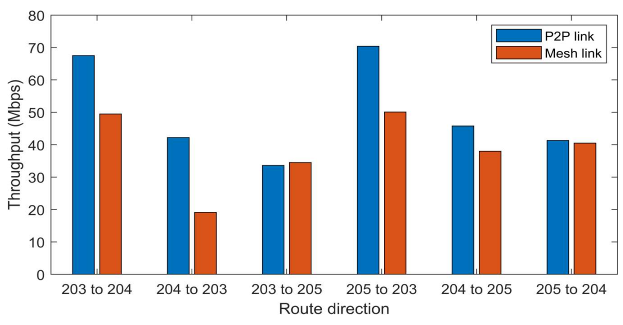
(c)

Fig. 3 NerveNet testbed. (a) x86 Intel NUC Wi-Fi. (b) armhf Raspberry Pi Wi-Fi. (c) x86 Intel NUC and armhf Raspberry Pi LoRa Mesh Network.

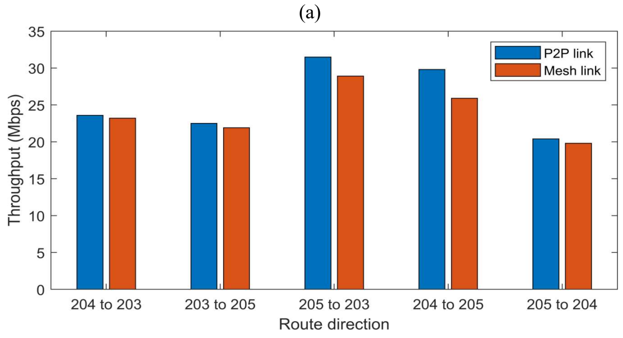
has good fundamentals to handle applications that require low jitter, such as providing VoIP services.

Fig. 5 measures the NerveNet x86 Wi-Fi latency. It can be seen that there is no big difference in terms of P2P and mesh links within NerveNet x86 Wi-Fi domain. However, the latency is less than 10 ms when the route direction is from Wi-Fi client interface to Wi-Fi AP interface, while the latency is five times greater if the route direction is reversed. This can be explained by NerveNet Wi-Fi in x86 machines having a lower route cost when the target's AP interface is the next hop of its own client interface. Therefore, even if the target is just located at the next hop of its AP interface, the sender would still seek for target from its client interface's next hop, causing the packet return time to increase.

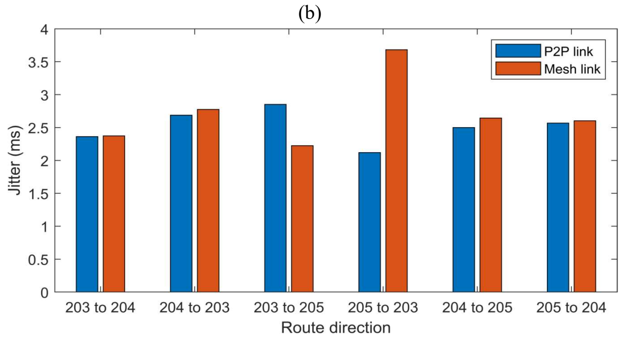
Similar as in x86 environment, Fig. 6 evaluates the TCP and UDP throughput for P2P and mesh links in NerveNet armhf Wi-Fi. From the figure, it is observed that the P2P link generally has a higher throughput as compared with mesh link. Unlike x86 machines, the relationship between throughput and route direction in NerveNet armhf Wi-Fi domain is not obvious. The highest throughput (BS207 to BS206) and lowest throughput (BS206 to BS208) via mesh link are both obtained when the target is located at the next hop of the sender's AP interface. Fig. 6 (c) shows the average jitter of P2P and mesh links within NerveNet armhf Wi-Fi domain.



(a)



(b)



(c)

Fig. 4 NerveNet x86 Wi-Fi. (a) TCP throughput (b) UDP throughput. (c) UDP jitter.

The variance of jitter at each link and route directions is as tiny as ignorable. However, even the highest jitter is just between 0.5 to 0.6 ms, which is at least three times lesser than the jitter in NerveNet x86 Wi-Fi domain.

To evaluate the time taken from an image file to be synchronized in all NerveNet node databases via Wi-Fi mesh, the cases of all nodes as image senders are tested. The corresponding time taken for the other two peers to receive the synchronized image file is recorded in Fig. 7.

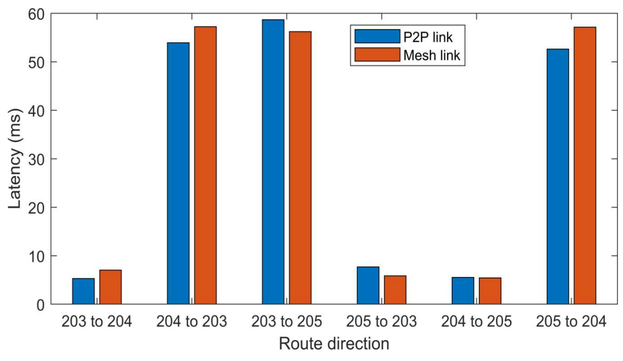
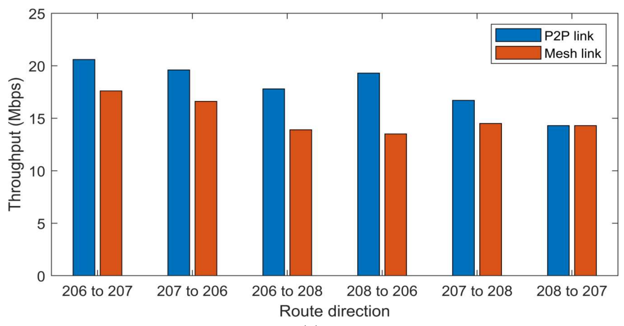
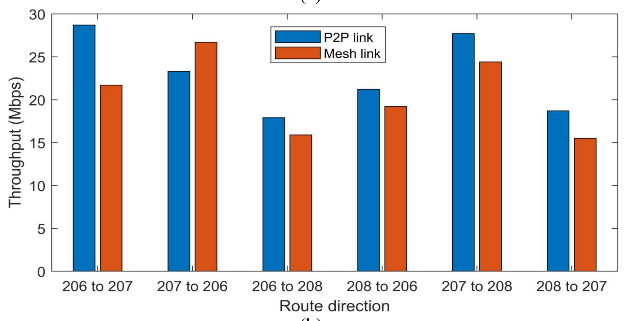


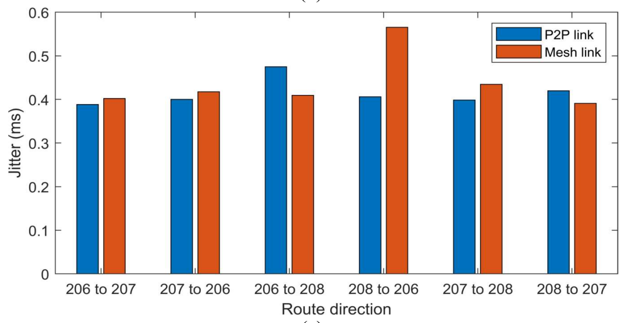
Fig. 5. NerveNet x86 Wi-Fi Latency.



(a)



(b)

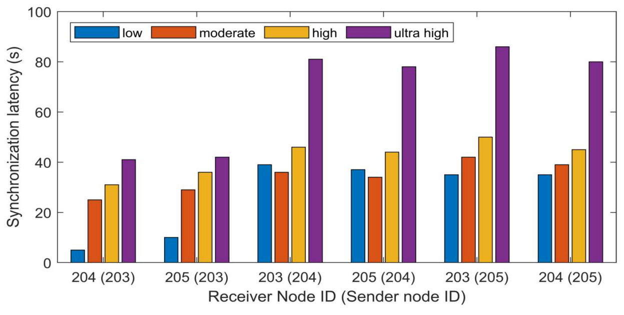


(c)

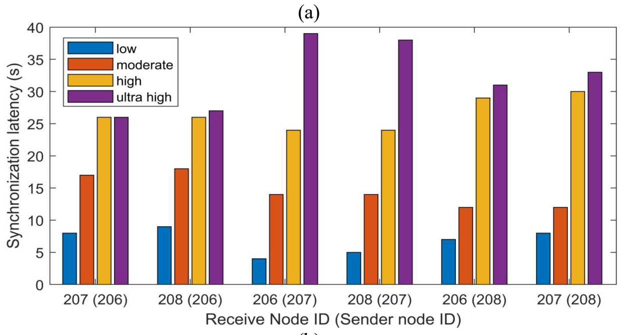
Fig. 6 NerveNet armhf Wi-Fi. (a) TCP throughput (b) UDP throughput. (c) UDP jitter.

From Fig. 7 (a), it is clearly stated that as the image file size increased, the time taken for the peers to receive the synchronized file also increased. However, the time taken is not linear. For example, Node 203 takes 50 seconds to receive a 2.5 MB image file from BS205. However, it only takes 86 seconds to receive a 10.9 MB image file, which is at least four times greater than the 2.5 MB image file. The figure also shows that the image sent by BS203 takes least time to be synchronized in the peers' database, this could be due to the PTMGR daemon running at BS203, therefore it takes the least time to calculate Wi-Fi pathways. Similar trend can be observed in Fig. 7 (b).

Since NerveNet LoRa mesh MQTT uses QoS level zero, the percentage of lost LoRa packets is interested. To test the NerveNet LoRa mesh MQTT messaging performance, the number of packets lost with MQTT payload size of 30 Bytes and 90 Bytes are recorded accordingly. Not only that, the number of LoRa packets sent at once could affect the ratio of lost packets, hence the number of LoRa messages published at once is varied at 10, 20, 40, and 60 messages. After the LoRa MQTT subscriber has not received any message for 20 minutes, the remaining LoRa packets are considered lost. The test is carried out using BS203 as LoRa MQTT subscriber while BS206 as the LoRa MQTT publisher.



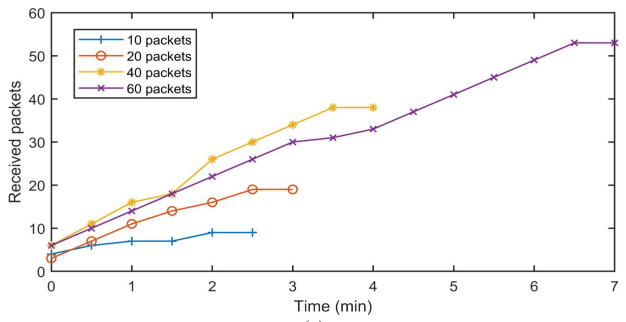
(a)



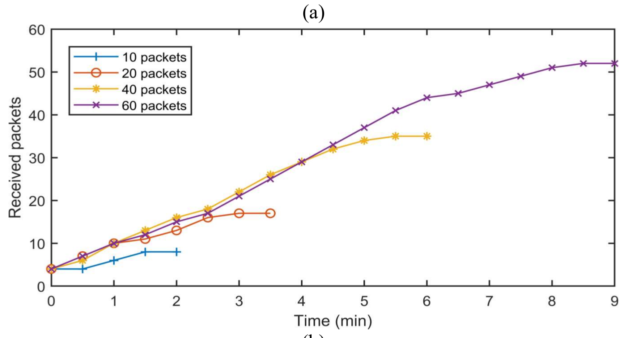
(b)

Fig. 7. Time taken for NerveNet image synchronization. (a) x86 Wi-Fi (b) armhf Wi-Fi.

Fig. 8 shows the time taken for LoRa packet payload transmission. It can be seen that the number of received LoRa MQTT messages is almost linear with time. This means the rate of LoRa MQTT message to be received is almost constant. However, the time taken to receive LoRa MQTT message is not a manipulated variable on the subscriber side. Instead, the published LoRa message is queued and buffered in MQTT broker to wait for transmission, once the published message is transmitted over NerveNet LoRa mesh, the over-the-air duration is short, thus it is almost immediately received by the subscriber side. The manipulating variable is said to be LoRa MQTT payload size because the bandwidth is fixed, higher payload size means a higher bit rate, therefore increasing the risk of LoRa signals being interfered or corrupted.



(a)



(b)

Fig. 8. Time taken for NerveNet LoRa packet payload transmission. (a) 30 bytes (b) 90 bytes.

To reveal more insight, Fig. 9 displays the number of NerveNet MQTT LoRa packet loss. By dividing the total message sent by the number of packets lost, the percentages of lost packets with 30 Bytes payload size in 10, 20, 40, and 60 messages are 10%, 10%, 5%, and 11.67% respectively. On the other hand, with the payload size of 90 Bytes, the percentage of lost packets in 10, 20, 40, and 60 messages are 20%, 15%, 12.5%, and 13.33% respectively. Hence, it is concluded that the larger the LoRa MQTT payload size, the slower the LoRa packet transmission, and the higher the risk of LoRa packet being lost.

Fig.10 displays the visual design of NerveDASH. The design phase of the frontend development is separated into three phases, namely sitemap designing, wireframing and visual designing.

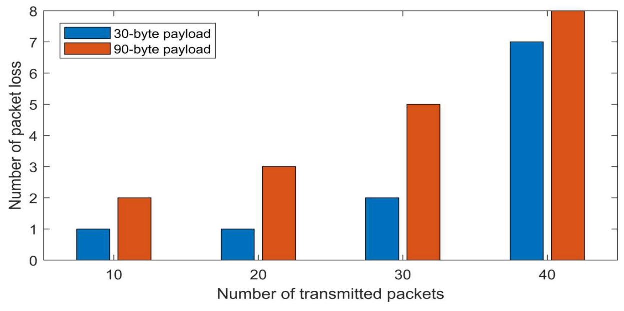
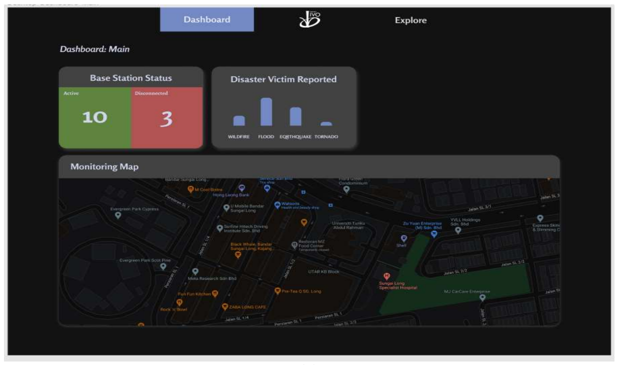
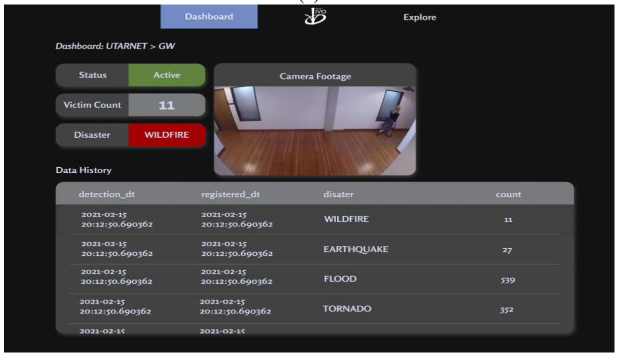


Fig. 9. Number of LoRa MQTT Packet Loss.



(a)



Dashboard: UTARNET > GW

Status	Active
Victim Count	11
Disaster	WILDFIRE

Data History

detection_dt	registered_dt	disater	count
2021-02-15 20:12:50.690362	2021-02-15 20:12:50.690362	WILDFIRE	11
2021-02-15 20:12:50.690362	2021-02-15 20:12:50.690362	EARTHQUAKE	27
2021-02-15 20:12:50.690362	2021-02-15 20:12:50.690362	FLOOD	539
2021-02-15 20:12:50.690362	2021-02-15 20:12:50.690362	TORNADO	352
2021-02-15	2021-02-15		

(b)

Fig. 10. NerveDASH. (a) Main page. (b) Node page.

V. Conclusions

A testbed for disaster response and monitoring platform using NerveNet has been designed and deployed. The tools such as the Hearsay daemon provided in NerveNet has been proven beneficial for application services within the regional local private network. The platform design also meets the requirements set by the International Telecommunication Union. The performance of the platform is also within an acceptable range of a regional disaster response and monitoring network. The outcome of the real implementation can serve as a guideline on designing and deploying a disaster response and monitoring platform using NerveNet. Hopefully, it will promote disaster-resilient telecommunications and distributed application development for disaster response.

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] "2021 Global Natural Disaster Assessment Report", [online] Available: https://reliefweb.int/report/world/2021-global-natural-disaster-assessment-report.

[2] "Malaysia: Disaster Management Reference Handbook (June 2019)", [online] Available: https://reliefweb.int/report/malaysia/malaysia-disaster-management-reference-handbook-june-2019

[3] N. S. Romali and Z. Yusop, "Flood damage and risk assessment for urban area in Malaysia," Hydrology Research, vol. 52, no. 1, pp. 142–159, 2021, doi: 10.2166/nh.2020.121

[4] M. Inoue and Y. Owada, "NerveNet Architecture and Its Pilot Test in Shirahama for Resilient Social Infrastructure," IEICE Transactions on Communications, vol. E100–B, no. 9, pp. 1526–1537, 2017.

[5] S. Ganesh, V. Gopalasamy and N. B. Shibu, "Architecture for Drone Assisted Emergency Ad-hoc Network for Disaster Rescue Operations," in COMSNETS, 2021.

[6] J. C. P. M. Villanueva et al, "Design and Deployment of Content Stacks and Portable Asynchronous Learning Platforms for Socially Distanced Learning in a Pandemic or Post Disaster Situation," in IEEE GHTC 2022.

[7] Y. Dalpathadu et al., "Disseminating Data using LoRa and Epidemic Forwarding in Disaster Rescue Operations," in GoodIT 2021.

[8] O. A. Saraereh, A. Alsaraira, I. Khan and P. Uthansakul, "Performance Evaluation of UAV-Enabled LoRa Networks for Disaster Management Applications," Sensors, vol. 20, no. 8, 2020, doi: https://doi.org/10.3390/s20082396

[9] J. Lohokare and R. Dani, "An Intelligent cloud ecosystem for disaster response and management leveraging opportunistic IoT mesh networks," in ICT-DM, 2021, pp. 125-133, doi: 10.1109/ICT-DM52643.2021.9664137.

[10] ESNET. Iperf. http://software.es.net/iperf/, 2022.

[11] A. Al-Shaikhli, A. Esmailpour and N. Nasser, "Quality of service interworking over heterogeneous networks in 5G", in 2016 IEEE ICC, pp. 1-6, 2016.