Accurate and Fast Classification of Natural Disasters using CNN-LSTM and Inference Acceleration

Nathaniel Sze Yang Tan, Mau-Luen Tham, Sing Yee Chua, and Ying Loong Lee

Abstract—**Catastrophic occurrences induced by disasters often lead to fatalities, extensive damage, and societal disruptions. In pursuit of realizing disaster-resilient smart cities, video surveillance systems incorporating artificial intelligence (AI) can automatically process and classify the disaster content in real-time. This advancement is fueled by the recent progress in computer vision and AI algorithms, specifically deep learning neural networks, which can be leveraged for disaster categorization tasks. However, minimizing the complexity of AI models while preserving accurate disaster classification remains a formidable challenge. In this paper, we propose a convolutional neural network-long short-term memory (CNN-LSTM) model capable of discerning four types of natural disasters and a non-disaster event. Contrary to prior research that treats input video as a sequence of independent frames, we demonstrate the significance of spatio-temporal characteristics in reaping high prediction accuracy. Furthermore, conventional methods rely on resource-intensive hardware to boost AI model performance, which may not suit real-time monitoring. To facilitate real-time disaster monitoring applications, the trained model is further optimized by utilizing a neural network acceleration platform known as OpenVINO. Our findings reveal that the optimized version of the proposed CNN-LSTM model sustains 100% accuracy while boosting throughput by 25% in terms of frames per second (FPS).**

Index terms—**CNN-LSTM, deep learning, disaster classification, inference optimization, OpenVINO.**

I. Introduction

Over the past decade, there has been a marked escalation in the frequency of natural disasters worldwide, including cyclones, earthquakes, floods, and wildfires [1]. These calamitous events pose considerable threats to both physical and environmental securities, potentially resulting in numerous casualties and extensive property damage. Large-scale disaster monitoring is possible by deploying a plethora of video surveillance cameras [2]. Equipped with well-trained artificial intelligence (AI) model, these smart systems can automatically process and classify the disaster content in real-time. For actual AI model deployments, both the correctness of the AI result and processing time play a crucial role in establishing effective disaster response.

Convolutional neural networks (CNNs) represent one of the most commonly employed AI techniques for disaster classification, owing to their exceptional performance in the domain of image classification [3]. Specifically, CNNs can process input videos on a frame-by-frame basis, extracting features that facilitate accurate classification. Popular CNN architectures, such as VGG-16 [4] and EfficientNet [5], have demonstrated the ability to achieve up to 81.6% accuracy [6] when utilizing datasets comprised of static disaster images. On the other hand, the data obtained from video feeds, such as video surveillance cameras, are presented in the form of moving images that are connected to one another rather than static disaster images. Consequently, these models are unable to adequately capture the diverse spatio-temporal characteristics inherent in video data. In practice, the direct application of pure CNN models to camera feeds may result in fluctuating predictions across successive video frames, an issue that can be mitigated through the implementation of rolling average prediction (RAP).

Long short-term memory (LSTM) networks are utilized in a vast array of problem domains. LSTMs can be employed either individually or combined with other deep learning architectures. LSTM networks possess the capacity to effectively discern temporal features within sequential data [7]. However, their performance is limited to low-dimensional data, such as texts and speeches [8-9]. Lacking convolutional and max-pooling layers typically found in CNNs, LSTMs are ill-equipped to manage the complex patterns present in video data. Given the prevalence of deep learning (DL) techniques, numerous researchers have combined CNN with LSTM, resulting in CNN-LSTM models for applications such as speech and handwriting recognition, as well as image and sound classifications [8-10]. In this study, we leverage the synergistic advantages of CNN-LSTM to develop an accurate disaster classification model.

Model complexity is undeniably a critical factor in the successful implementation of disaster monitoring systems. Traditional approaches rely on resource-intensive graphics processing units (GPUs) to accelerate AI workloads. While the GPU parallelism is highly appreciated during the training phase, optimizing and executing models during the inference phase without drawing massive power consumption from the GPU remains an open research question. The Intel OpenVINO

Manuscript received November 23, 2023; revised January 8, 2024. Date of publication January 30, 2024. Date of current version January 30, 2024. The associate editor prof. Maja Braović has been coordinating the review of this manuscript and approved it for publication.

This research was supported by Universiti Tunku Abdul Rahman (UTAR), Malaysia and ASEAN IVO project. The preliminary version of the paper was presented in at the 13th International Conference on Ubiquitous and Future Networks (*ICUFN*) 2022.

Authors are with the Department of Electrical and Electronic Engineering, UTAR, Malaysia (e-mails: nat980718@1utar.my, thamml@utar.edu.my - corresponding author, sychua@utar.edu.my, leeyingl@utar.edu.my).

Digital Object Identifier (DOI): 10.24138/jcomss-2023-0129

toolkit has emerged as a promising solution to these challenges, thanks to its ability to fine-tune and optimize DL inference performance across various low-powered target platforms [11].

In this work, we focused on classification models to identify the key elements that impact the performance of quantization. We discovered that the model parameters play a significant role in quantization performance. VGG16 [4] has a large number of parameters, whereas MobileNet_v1 [40] and ResNet-50 [41] reduce their parameters using depth-wise separable convolutional layers and 1×1 filters. As a result, the speedup and improvement factors achieved through quantization are not as significant compared to VGG16.

This study has the potential to be expanded to various other deep learning models, including natural language processing, graph neural networks, pose estimation, and segmentation models, in order to gain valuable insights into how quantization impacts the performance of these models. Additionally, by examining the optimization techniques employed in deep learning frameworks, researchers can adopt the most effective practices for achieving efficient results during both training and inference stages. This characterization of optimization techniques can further facilitate the development of more efficient deep learning optimization methods by understanding their performance across different models and datasets.

To achieve optimal deep neural network inference, different processors such as NVIDIA’s TensorRT, Google's TensorFlow Lite, and Intel's OpenVINO require specific deep learning frameworks. NVIDIA’s TensorRT is a software to optimize and accelerate deep learning inference that utilizes NVIDIA GPUs equipped with embedded processors and neural processing units (NPUs) [42]. TensorFlow Lite [43] is a toolset that facilitates on-device machine learning, allowing developers to deploy their models on edge devices. OpenVINO is an advanced toolkit developed by Intel to enhance the performance of neural networks on Intel hardware [44]. Additionally, OpenVINO is specifically designed to optimize the efficiency and speed of neural network computations for resource-constrained edge device. In this work, we select OpenVINO as it enables the swift deployment of applications and solutions in various domains [45]. The proposed solution should be portable in such a way that the OpenVINO optimized model can be directly converted to TensorFlow Lite, as supported in [46].

In our previous study [12], we applied a transfer learning approach to a VGG-16 model with varying model precisions for disaster classification. Upon optimization by OpenVINO, we observed a loss of approximately 1% in accuracy compared to our original disaster classification model. Specifically, we consider the synchronous mode of the inference process, wherein the AI-optimized model receives images from a video stream and generates instantaneous predictions.

In this paper, we build upon the work presented in [12] with the objective of developing a new disaster classification model utilizing CNN-LSTM. This model is designed to capture the temporal dependencies within the input video, enabling the classification of cyclone, flood, earthquake, and wildfire disasters, as well as non-disaster events, from video clips. The main contributions of this research are summarized as follows:

1. We have assembled a dataset comprising up to 343 video clips, each containing one type of event (cyclone, flood, earthquake, wildfire disaster, or non-disaster). This dataset serves as the input for the proposed CNN-LSTM model.
2. We have developed a CNN-LSTM-based disaster classification model to classify disaster and non-disaster events. The proposed CNN-LSTM model contains 30 LSTM cell units, capable of storing long-term dependencies from 30 input data frames. The model is further finetuned by optimizing parameters within the dense layer, establishing the best architecture for disaster classification tasks given the relatively small dataset.
3. Trained using the native TensorFlow platform, the proposed CNN-LSTM model demonstrates superior accuracy performance compared to pure-CNN and CNN-RAP models. Optimized in the OpenVINO platform, the performance in terms of inference speed and latency of the proposed CNN-LSTM method has been further enhanced. The inference speed of the optimized models increases by 25.0% - 63.1% in terms of FPS and exhibits 19.7 - 38.8% lower latency compared to the original (unoptimized) models when running in CPU-only mode.

The remainder of this paper is organized as follows: Section II reviews related works; Section III details the research methodology and model architectures; Section IV presents the results and discussions; and finally, Section V offers concluding remarks.

II. Related Work

A. *CNN*

Recently, many disaster classification studies have been proposed using AI, particularly CNNs [13]. Notably, majority of the related studies conducted were based on CNNs rather than traditional machine learning methods by the reason of the superior performance of CNNs [6].

In particular, the authors of [14] used a CNN and employ a transfer learning approach using VGG-16 to classify four types of natural disasters, including hurricanes, earthquakes, floods and wildfires. However, CNN has the drawbacks of classifying each input image independently of each other, which may hamper its performance, given that video is used as the input. As a result, their CNN model scores low accuracy in classifying natural disasters in videos. Furthermore, their model was not trained to classify non-disaster scenarios.

Recognizing the importance of the absence of non-disaster event classification, our previous work in [12] trained a model to classify the above-named natural disasters and an additional non-disaster event from video clips. The CNN model achieves an accuracy of 92.30 % with 21.35 FPS in the OpenVINO environment.

Nevertheless, inspired by the effectiveness of the VGG-16, we adopt it for the current study due to the ease of use and superior accuracy of the model.

B. RNN

A number of studies that work on long-term data dependencies is based on recurrent neural networks (RNNs) [15], as RNNs are widely used in modeling sequence data. RNNs can remember the important information i.e., the input data, thanks to its internal memory, that allows them to be very precise in predicting the output. LSTM [7] and gated recurrent unit (GRU) [16] are actually based on the architecture of RNNs that is also designed to work on long-term dependencies in data. LSTM has three gates: Forget gate, input gate and output gate; whereas GRU has fewer parameters than LSTM with only two gates: Update gate and reset gate [17].

Researchers have made efforts to utilize RNN for capturing long-term temporal patterns in videos. For instance, the study in [18] proposed using LSTM to convert videos into textual sentences, leveraging knowledge from image description tasks. Another study [19] employed LSTM for modeling temporal information in the context of video description. Their work did not consider using RNN video classification, rather they focused on using it to generate video descriptions. This solidifies the statement that RNNs are limited to low-dimensional data, such as texts and speeches as it lacks convolutional and max-pooling layers typically found in CNNs to extract the features.

It is noteworthy that there are several variants of LSTM. Vanilla LSTM has a single hidden layer containing LSTM units that is composed of a cell, an input gate, an output gate and a forget gate [20]. Stacked LSTM has multiple hidden LSTM layers that are stacked on top of one another, making the model deeper and higher level of abstraction [21]. Bidirectional LSTM has two hidden LSTM layers of opposite direction, so that information is utilized from both sides [21].

In this paper, we select the simpler vanilla LSTM structure which is more preferred for real-time disaster monitoring applications rather than other LSTM structure as vanilla LSTM is computationally less intensive, allowing better performance in inference speed.

C. CNN-LSTM

A video is comprised of a sequential arrangement of frames, where each frame holds spatial data, and the sequence of frames captures temporal information. To effectively capture both of these elements, a hybrid architecture, which combines CNN for spatial processing and LSTM for temporal processing, is employed. This particular type of hybrid architecture is commonly referred to as a CNN-LSTM.

The study in [14] combined the CNN and LSTM to classify natural disasters such as cyclones, earthquakes, floods, and wildfires from videos. Interestingly, their VGG16-LSTM model is able to achieve two times higher accuracy than that of using CNN alone. However, their LSTM only trains to classify texts in the video description. In contrast, our aim is to utilize the extracted features from the CNN and LSTM to classify natural disasters based on the extracted features.

Realizing that RNNs have demonstrated good performance on the tasks where temporal information is important, the study in [22] utilizes the power of CNN for extracting spatial information followed by a single-layer LSTM for temporal processing. Their CNN-LSTM model achieved 61-% accuracy in video classification for video facial expression recognition tasks. The study's moderate accuracy scores do not provide a definitive conclusion on the superiority or inferiority of using CNN-LSTM, as the authors noted the absence of prior publications demonstrating video classification performance on the specific dataset employed in their research.

It is worth noting that one of the challenges of using deep learning for video processing, particularly CNN-LSTM, is the high computing resources required for large-scale video data processing [23].

D. CNN-RAP

In general, a video contains a series of moving images that are co-related with each other. Therefore, averaging the prediction can improve the accuracy of the classification result. The motivation of integrating the rolling average prediction (RAP) to the pure-CNN model is that the CNN-RAP model may achieve a performance similar to that of the CNN-LSTM model with a lower computing power.

The research conducted in [24] proposed a technique for implementing a rolling prediction average (similar to RAP) algorithm to enable video classification and recognition of abnormal interference in surveillance camera systems. The outcomes indicate that CNN-RAP outperforms alternative machine learning approaches like support vector machine (SVM) in terms of accuracy scores.

III. RESEARCH METHODOLOGY AND MODEL ARCHITECTURES

A disaster-related image dataset is obtained online known as Crisis Image Benchmarks Dataset (CrisisIBD) [25]. The disaster images are combined to form multiple videos of exactly 30 frames per video. Natural disasters images and videos are used as input data that require preprocessing before the deep learning tasks. Since the CNN models do not accept different input image sizes, a suitable image dimension of 224 × 224 is selected.

The dataset collected from [25] is first split into 82% for the training and then validation while 18% for the testing. The 82% data is subsequently separated into 90% and 10% for the training and validation datasets, respectively. To further assess the generalization capability of the trained models, samples collected from another source of image dataset [47] and video [48-51] are adopted for the testing purpose. A total of three models are trained: A pure-CNN model and two proposed CNN-LSTM models are trained with either 80 or 160 of neurons in the dense layer. After the pure-CNN model has been trained, an additional model is developed by integrating the RAP to the pure-CNN model, as known as the CNN-RAP model. The performance of the four models is measured using the testing dataset.

In the interest of deploying the disaster monitoring models in real-time, it is important to know their processing speed in terms of FPS and latency. Thus, additional performance measurement is accomplished in the TensorFlow environment and the OpenVINO environment.

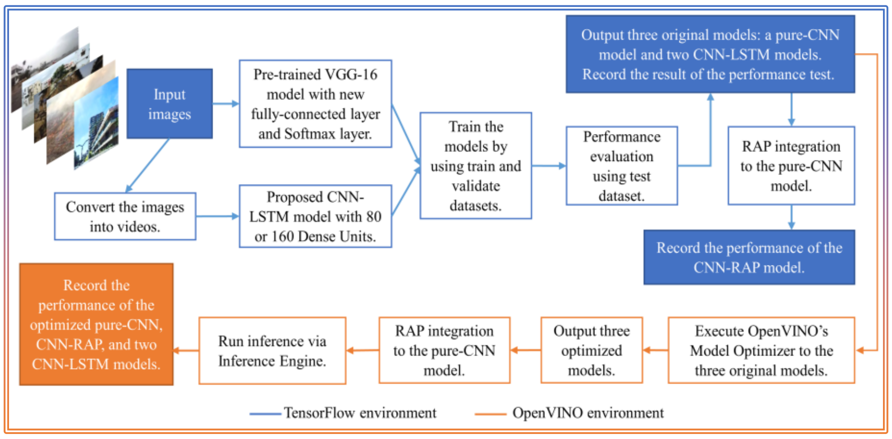
Input images
Pre-trained VGG-16 model with new fully-connected layer and Softmax layer.
Convert the images into videos.
Proposed CNN-LSTM model with 80 or 160 Dense Units.
Train the models by using train and validate datasets.
Performance evaluation using test dataset.
Output three original models: a pure-CNN model and two CNN-LSTM models. Record the result of the performance test.
RAP integration to the pure-CNN model.
Record the performance of the CNN-RAP model.
Execute OpenVINO's Model Optimizer to the three original models.
Output three optimized models.
RAP integration to the pure-CNN model.
Run inference via Inference Engine.
Record the performance of the optimized pure-CNN, CNN-RAP, and two CNN-LSTM models.
TensorFlow environment
OpenVINO environment

Fig. 1. Methodology of the proposed models

The overall implementation of this work is visualized in Fig. 1, which also shows the procedure of training and attaining the performance of various models in two different environments.

A. *Dataset Preparation*

To ensure a high classification accuracy of the new models, the models need to be trained with a good image dataset. The images in the dataset have to be grouped together with the correct labels. The reason for the need of a good dataset is that the pure-CNN model and the proposed CNN-LSTM models are deep learning models, and their classification performance is directly affected by the training dataset. Assuming that the image dataset contains non-related images for a specific label, the model's classification accuracy for that specific label will perform poorly. Preprocessing refers to the point of obtaining the dataset online, filtering the dataset, and attaining dataset quality enhancements for improving the performance of the deep learning models. Enhancements of the dataset include image dimension resizing for better learning efficiency of the deep learning models in the training phase.

The effectiveness of disaster classification of the models is closely linked to the amount and quality of data available. Artificial Intelligence for Disaster Response (AIDR) is a dataset created by the authors in [26], which consists of filtered tweet messages posted by individuals during disasters. A similar work can be found in [27] where the authors have published a sizable collection of multimedia data sourced from Twitter known as Multimodal Crisis Dataset (CrisisMMD) that relates to various natural disasters.

In order to make benchmarking fair-and-square, the authors in [25] combined the datasets mentioned earlier into one dataset named CrisisIBD, which will be used as the input dataset for this paper, as the dataset yields promising results by the work in [28]. Fig. 2 shows the typical images found in the dataset. The number of classes in the dataset used in this study will be less than that of the dataset obtained in [25]. Our objective is not to enhance accuracy performance by eliminating certain classes from the original dataset. Instead, we aim to improve accuracy performance by leveraging a more sophisticated deep learning model, specifically the CNN-LSTM model. The images within each category of natural disasters in the dataset were utilized in their original form to train the different models and any overlapping categories are not taken into account.

Table I shows the dataset in the proposed study for the pure-CNN model. Using the same dataset, the image datasets have been converted to video as shown in Table II for the proposed CNN-LSTM models. Each video has a fixed frame rate of 30 FPS, and each frame relates to one image only. As for the video source obtained from [48-51], each video was trimmed to contain precisely 30 frames. Hence, each video is precisely one second long and comprises exactly 30 images of a specific class of natural disaster or non-disaster class, thereby fulfilling the time consistency needed for video-based disaster classification for the CNN-LSTM models. The non-disaster class of the test dataset exhibits a larger data volume compared to the other classes, as this is intended to mimic real-world scenarios where the absence of a disaster is the norm.

TABLE I
IMAGE DATASET SPLIT FOR THE PURE-CNN MODEL

Disaster Label	*Train*	*Validate*	*Test*	*Total*
Cyclone	1,080	120	270 + 90*	1,470
Earthquake	1,080	120	270 + 90*	1,470
Flood	1,080	120	270 + 90*	1,470
Non-disaster	1,080	120	870 + 90*	2,070
Wildfire	1,080	120	270 + 90*	1,470

*Additional test images from source [47-51] for inference.

TABLE II
VIDEO DATASET SPLIT FOR THE PROPOSED CNN-LSTM MODELS

Disaster Label	*Train*	*Validate*	*Test*	*Total*
Cyclone	36	4	9 + 3*	49
Earthquake	36	4	9 + 3*	49
Flood	36	4	9 + 3*	49
Non-disaster	36	4	29 + 3*	69
Wildfire	36	4	9 + 3*	49

*Additional test videos from source [47-51] for inference.

Fig. 2. Typical images for different disaster types [25]

B. Proposed Deep Learning Models

Neural networks in deep learning shortens the efforts of training a new model in months into hours or even minutes [29], depending on the hardware. For example, by using high-speed general-purpose hardware, such as GPUs, are able to effectively reduce the training time [30]. The pixel values of an input image are extracted and assigned to each neuron in a numeric form. The connection between neurons contains weights that serves as a strength between different layers of neurons. This study considers two types deep learning models, which are the pure-CNN model and the proposed CNN-LSTM models. Both models share a similar base CNN architecture, which is referred to as the base CNN model shown in Fig. 3. It is noteworthy that the CNN-RAP model adopts exactly the pure-CNN model with RAP integrated in the inference phase.

The selection of pure-CNN's model parameters is based on the findings from our previous studies in [12]. The CNN-LSTM's model parameters are experimented with some reference to [31], in which the authors used CNN and LSTM to detect violence from videos.

B.1. Pure-CNN Model

The pure-CNN model consists of a network of layered architectures. The pure-CNN model is selected to be a pre-trained VGG-16 model, due to its superior performance over other models [32]. The development of VGG-16 is to compete in image classification tasks; hence it suits our task at hand. The CNN layered architecture extracts the image pixels in the form of features. The layers of the pure-CNN model include the input layer, the multiple 2D convolutional layers, the multiple max-pooling layers as shown in Fig. 3, and custom model head such as the flatten layer, the dense layers and the dropout layer, as shown in Fig. 4. Table III shows the number of parameters present in each layer of the pure-CNN model and a total number of 12,848,133 trainable parameters that are attributed by the custom model head. The layers of the original VGG16 model [4] are not trained as we are employing a transfer learning approach, therefore, the parameters of the original layers are treated as non-trainable parameters. Rectified Linear Unit (ReLU) activation function is used in the feature extraction layer and SoftMax activation function is selected in the classification layer [33].

B.2. CNN-LSTM Model

Our proposed CNN-LSTM model is an improved CNN followed by an LSTM model architecture in terms of parameter fine-tuning. The hybrid model has the ability of storing long term dependencies of 30 data, hence allowing better prediction as the model is able to receive an input of multiple consecutive images [31]. Since the CNN model only accepts a single input image, the layers of the base CNN model are wrapped in the *TimeDistributed* function to enable the model to process 30 images as an input. A *TimeDistributed* layer will add an additional dimension to the input shape [34], that is the number of images to process per input in addition to the image dimensions and the number of color channels (i.e., frame, width, height, channel). The input parameters for the proposed model are (30, 224, 224, 3).

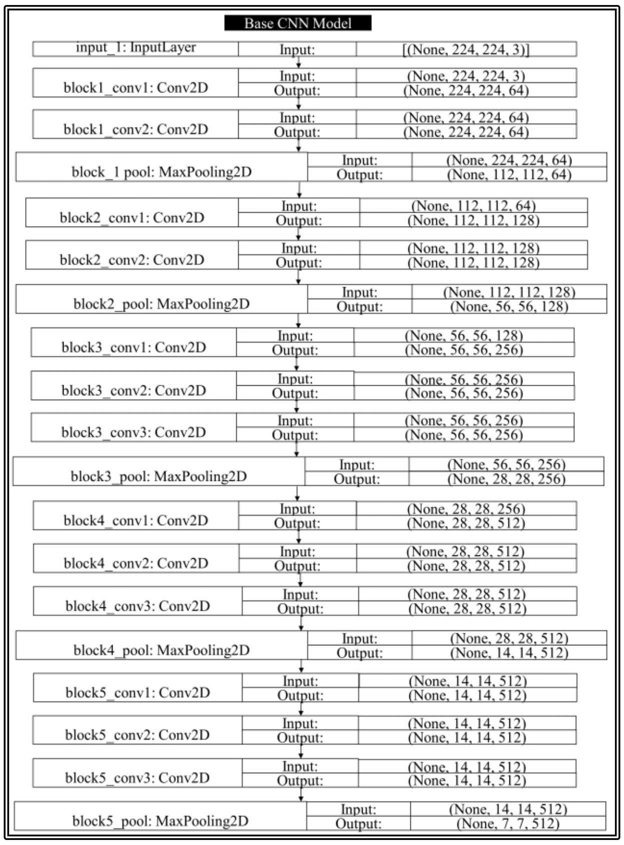

Fig. 3. Architecture of the base CNN model

In the training phase, the CNN-LSTM model utilizes each video (30 frames) in the dataset as input to learn the temporal relationship across different disasters. In the inference phase, the CNN-LSTM model analyzes frame-by-frame from the 1st to the 30th frame of the test video, before producing the classification result. Since the test videos can be up to 29 seconds long, during the second inference, the CNN-LSTM model analyzes frames sequentially from the 2nd to the 31st frame and then produces the classification result. This process is repeated until the entire video stream has been processed.

TABLE III
PURE-CNN MODEL LAYERS AND THE NUMBER OF PARAMETERS

Pure-CNN Layer (type)	*Number of Parameters*
input_1 (InputLayer)	0
block1_conv1 (Conv2D)	1,792
block1_conv2 (Conv2D)	36,928
block1_pool (MaxPooling2D)	0
block2_conv1 (Conv2D)	73,856
block2_conv2 (Conv2D)	147,584
block2_pool (MaxPooling2D)	0
block3_conv1 (Conv2D)	295,168
block3_conv2 (Conv2D)	590,080
block3_conv3 (Conv2D)	590,080
block3_pool (MaxPooling2D)	0
block4_conv1 (Conv2D)	1,180,160
block4_conv2 (Conv2D)	2,359,808
block4_conv3 (Conv2D)	2,359,808
block4_pool (MaxPooling2D)	0
block5_conv1 (Conv2D)	2,358,808
block5_conv2 (Conv2D)	2,359,808
block5_conv3 (Conv2D)	2,359,808
block5_pool (MaxPooling2D)	0
flatten (Flatten)	0
dense (Dense)	12,845,568
dropout (Dropout)	0
dense_1 (Dense)	2,565
Total parameters	27,562,821
Trainable parameters	12,848,133
Non-trainable parameters	14,714,688

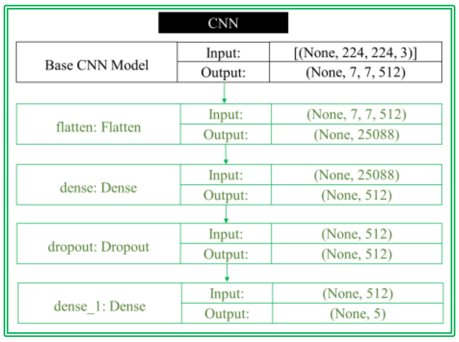

Fig. 4. Architecture of the pure-CNN model

The units in the LSTMs are known as memory cells, which are used to store the knowledge of previous states given a known time interval. Fig. 5 illustrates a single LSTM cell [35]. Each LSTM cell has three gates to regulate the information flow. In each time step, the current input x_t, current hidden state h_t and past hidden state h_{t-1} of the cell information c_t is regulated by the three gates, that are the input gate i_t, output gate o_t, and forget gate f_t. Sigmoid function $\sigma(.)$ ensures that the output is within the range of [0,1]. The value of the forget gate f_t can decide what information to be retained or discarded from the cell state, that is, the value of 1 is to keep the information and value 0 means release all the information [35].

The output of the *TimeDistributed* CNN model with the shape of [30×25088] is served as the input to the LSTM layer. The LSTM layer encompasses 30 unit of cells, with each cell representing a time step, and each time step is a frame of an image. Utilizing the full sequence prediction of the LSTM, a dense layer with 80 or 160 neurons is applied followed by a global average pooling layer, and a dense layer as the final layer for classification duty. In this paper, the number of neurons in the dense layers (80 or 160) is the hyperparameter settings to obtain the best performance. Similar strategy is advocated in [52-54], where at least five choices of different number of neurons are analyzed.

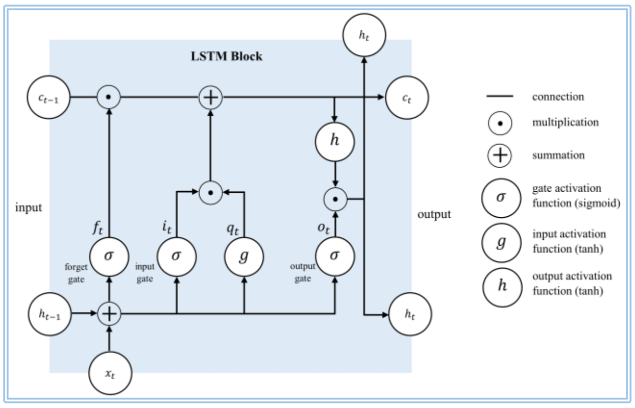

Fig. 5. Architecture of a vanilla LSTM cell

In short, the architecture of the proposed CNN-LSTM model is built on the CNN model as a feature extractor followed by LSTM cells, and can be seen by linking Fig. 3 and Fig. 6. The number of parameters present in each layer of the CNN-LSTM model is shown in Table IV. In the CNN-LSTM model, the LSTM component accounts for the majority of the trainable parameters, with 3,017,165 and 3,020,045 parameters for models containing 80 and 160 neurons in the dense layer, respectively.

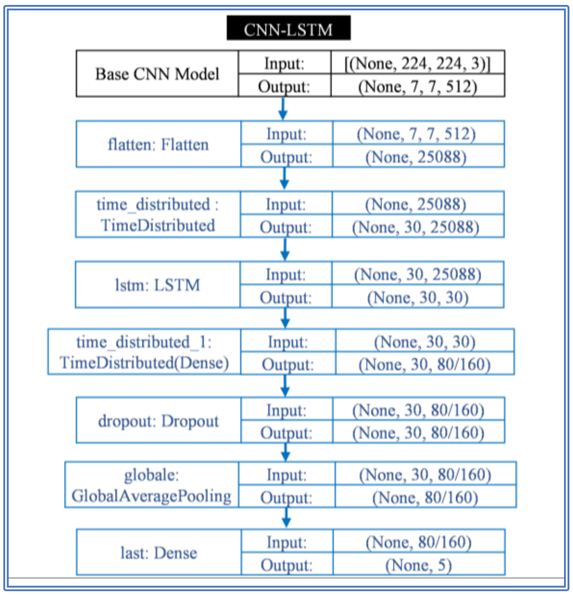
CNN-LSTM
Base CNN Model
Input: [(None, 224, 224, 3)]
Output: (None, 7, 7, 512)
flatten: Flatten
Input: (None, 7, 7, 512)
Output: (None, 25088)
time_distributed : TimeDistributed
Input: (None, 25088)
Output: (None, 30, 25088)
lstm: LSTM
Input: (None, 30, 25088)
Output: (None, 30, 30)
time_distributed_1: TimeDistributed(Dense)
Input: (None, 30, 30)
Output: (None, 30, 80/160)
dropout: Dropout
Input: (None, 30, 80/160)
Output: (None, 30, 80/160)
globale: GlobalAveragePooling
Input: (None, 30, 80/160)
Output: (None, 80/160)
last: Dense
Input: (None, 80/160)
Output: (None, 5)

Fig. 6. Architecture of the proposed CNN-LSTM models

C. *OPENVINO Optimization*

To benchmark the performance of the natural disaster classification models, all the trained models need to be converted and executed in the OpenVINO environment due to the superior performance in terms of throughput. The steps of this phase are described as follows.

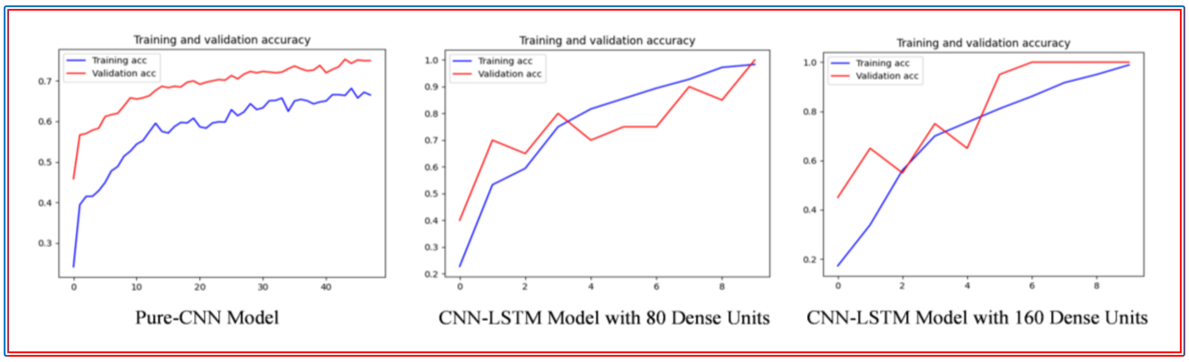

Fig. 7. Training and validation accuracy of various models

TABLE IV
CNN-LSTM MODEL LAYERS AND THE NUMBER OF PARAMETERS

CNN-LSTM Layer (type)	Number of Parameters
input_1 (InputLayer)	0[1,2]
block1_conv1 (Conv2D)	1,792[1,2]
block1_conv2 (Conv2D)	36,928[1,2]
block1_pool (MaxPooling2D)	0[1,2]
block2_conv1 (Conv2D)	73,856[1,2]
block2_conv2 (Conv2D)	147,584[1,2]
block2_pool (MaxPooling2D)	0[1,2]
block3_conv1 (Conv2D)	295,168[1,2]
block3_conv2 (Conv2D)	590,080[1,2]
block3_conv3 (Conv2D)	590,080[1,2]
block3_pool (MaxPooling2D)	0[1,2]
block4_conv1 (Conv2D)	1,180,160[1,2]
block4_conv2 (Conv2D)	2,359,808[1,2]
block4_conv3 (Conv2D)	2,359,808[1,2]
block4_pool (MaxPooling2D)	0[1,2]
block5_conv1 (Conv2D)	2,358,808[1,2]
block5_conv2 (Conv2D)	2,359,808[1,2]
block5_conv3 (Conv2D)	2,359,808[1,2]
block5_pool (MaxPooling2D)	0[1,2]
time_distributed (TimeDistributed)	14,714,688[1,2]
lstm (LSTM)	3,014,280[1,2]
time_distributed_1 (TimeDistributed)	2,480[1], 4,960[2]
dropout (Dropout)	0[1,2]
globale (GlobalAveragePooling)	0[1,2]
last (Dense)	405[1], 805[2]
Total parameters	17,731,853[1], 17,734,733[2]
Trainable parameters	3,017,165[1], 3,020,045[2]
Non-trainable parameters	14,714,688[1,2]

[1] CNN-LSTM model with 80 dense units, [2] CNN-LSTM model with 160 dense units.

1. *Obtaining the trained model:* All models are trained via a transfer learning approach and saved as the saved model ProtoBuf (PB) format for the pure-CNN model, whereas the models are saved with Keras Hierarchical Data Format version 5 (HDF5) for the two proposed CNN-LSTM models. It is noteworthy that fine-tuning of parameters is carried out for the purpose of decreasing the execution time and increasing the accuracy.
2. *Freezing the model:* The HDF5 file of the two proposed CNN-LSTM models are converted into the saved model (PB) format with its weight frozen. In contrast, the pure-CNN model in PB format can skip this step as the weights have already been frozen in the training stage.
3. *Converting the model to a compatible format:* All the trained models need to be converted into the Intermediate Representation (IR) format prior to their implementation in the OpenVINO environment [36]. The conversion is carried out via the Model Optimizer tool by the OpenVINO toolkit [37]. The input parameter for pure-CNN model is [1, 224, 224, 3] and proposed CNN-LSTM models is [1, 30, 224, 224, 3]. Explaining from the right-to-left of the input parameter, the value “3” represents the number of color channels, “224” and “224” represents the image dimensions, “1” or “30” represents the number of images or frames, and specially for our proposed CNN-LSTM, “1” represents one video per input. Instead of the command-line interface (CLI) method, the graphical user interface (GUI) method is utilized because of the intuitiveness of DL Workbench GUI and the essential scaling parameters. An XML file (which describes the network topology) and a BIN file (which contains the weights and biases binary data) [11] are generated once the conversion is successful.
4. *Executing inferences:* The execution of the model is performed via the Inference Engine (IE) tool from the OpenVINO toolkit. A custom script is applied to initiate essential plugins, load the IR model, read the label, fetch input video, perform inference using IE and process the output.
5. *Getting the prediction results:* Every output of each model is printed out in the terminal and recorded. The accuracy is manually calculated based on the recorded data. Latency in milli-seconds (ms) and throughput in FPS are generated after the inference is completed.

IV. Results and Discussions

This paper evaluates the performance of the natural disaster classification models to classify four natural disaster classes and a non-disaster class using three types of DL models, namely the pure-CNN model, CNN-RAP model, and proposed CNN-LSTM models.

1. *Hardware on the Training Phase:* The experiment in the TensorFlow environment is carried out on a computer equipped with an Intel i7-10710U 10th-generation central processing unit (CPU), an NVIDIA GeForce GTX 1080 Ti as the GPU, and 64 GB of memory.
2. *Hardware on the Inference Phase:* The original models and the optimized models utilize the same hardware as in the training phase. It is noteworthy that original models are able to use both the CPU and GPU, while the optimized

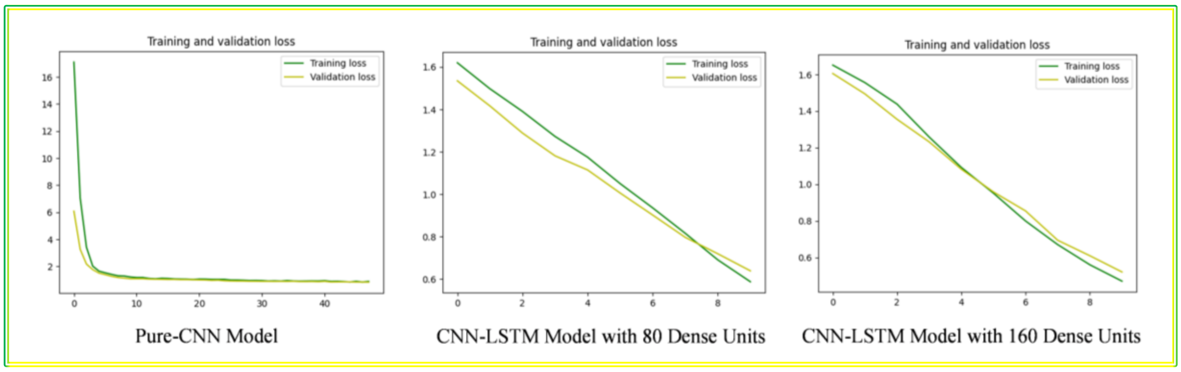

Fig. 8. Training and validation loss of various models

models are able to use the CPU and OpenVINO version 2021.4.

A. *Training Performance*

An assessment of performance tests with respect to precision, recall, f1-score, and accuracy are conducted once the model training is complete. The results are generated by applying the *classification_report* function. Fig. 7 – 8 visualize the training and validation accuracies as well as the losses for all the models.

The dataset split shown in Table I is used to train the pure-CNN model and produce the results in Table V, whereas the dataset split shown in Table II is used to train the proposed CNN-LSTM models and produce the results in Table VI.

TABLE V
TRAINING PERFORMANCE OF THE PURE-CNN MODEL

Disaster Label	*Precision*	*Recall*	*F1-score*	*Support*
Cyclone	0.60	0.56	0.58	270
Earthquake	0.72	0.79	0.75	270
Flood	0.60	0.71	0.65	270
Non-disaster	0.89	0.79	0.84	870
Wildfire	0.66	0.77	0.71	270
Accuracy	74 %			

TABLE VI
TRAINING PERFORMANCE OF THE PROPOSED CNN-LSTM MODELS WITH 80 OR 160 DENSE UNITS

Disaster Label	Precision	Recall	F1-score	Support
Cyclone	$1.00^1,1.00^2$	$0.89^1,1.00^2$	$0.94^1,1.00^2$	$9^{1,2}$
Earthquake	$1.00^1,1.00^2$	$0.89^1,1.00^2$	$0.94^1,1.00^2$	$9^{1,2}$
Flood	$1.00^1,1.00^2$	$0.89^1,1.00^2$	$0.94^1,1.00^2$	$9^{1,2}$
Non-disaster	$0.94^1,1.00^2$	$1.00^1,1.00^2$	$0.97^1,1.00^2$	$29^{1,2}$
Wildfire	$0.90^1,1.00^2$	$1.00^1,1.00^2$	$0.95^1,1.00^2$	$9^{1,2}$
Accuracy	$95\%^1$, $100\%^2$			

^{1} CNN-LSTM model with 80 dense units, ^{2} CNN-LSTM model with 160 dense units.

Tables V–VI show the performance results of the TensorFlow model. Precision indicates how confident the model classifies a given sample as positive; recall indicates how well the model classifies a positive sample as positive [38]; F1-score is a combination of precision and recall scores in a harmonic mean form; support value is the number of test data that are used to produce the results in their class; and accuracy is the percentage of the correct predictions to the total predictions. Equations (1–4) show the mathematical formulas for obtaining the precision, recall, F1-score and accuracy. True positive (TP), true negative (TN), false positive (FP) and false negative (FN) are required [30] for the generation of the results.

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

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

$$F1-score = 2 \times \frac{Precision \times Recall}{(Precision+Recall)}, \quad (3)$$

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)}. \quad (4)$$

The training results show that our proposed CNN-LSTM models perform better than the pure-CNN model with an accuracy improvement of up to 26 %. Initially, 80 neurons in the dense layer are employed and obtained a significantly better accuracy when compared to the pure-CNN model. Subsequently, we intend to boost the accuracy scores, and by employing the dense layer with 160 neurons, the accuracy improvement is even better, achieving 100 % accuracy.

B. *Inference Performance on TensorFlow Environment*

Since the task of natural disaster classification is carried out in real-time, it is essential to realize the performance of the models in terms of FPS and latency. The number of images processed in a single second is known as throughput (FPS), indicating the processing speed of the entire process. The time taken to perform an inference for every input is known as latency [39]. Table VII presents the performance of the pure-CNN, CNN-RAP, and CNN-LSTM models, using the test dataset that is compiled into video format. Accuracy is not affected by the change in hardware, as the precision of the model remains the same.

This subsection evaluates the performance of the various models relating to throughput, latency, and accuracy. The accuracy of each deep learning model is calculated by averaging the accuracy scores in each disaster class as shown in Equation 5.

$$Accuracy = \frac{\sum_{n=1}^{m} x_n}{m}, \quad (5)$$

such that, $\forall n, m \in \mathbb{N}$,

where x is the accuracy scores of each disaster class and m is the upper limit of the disaster classes.

TABLE VII
PERFORMANCE OF THE VARIOUS MODELS USING TEST DATASET VIDEOS IN TENSORFLOW ENVIRONMENT

Deep Learning Model	*Throughput (FPS)*	*Latency (ms)*	*Accuracy[a] (%)*	*Accuracy[b] (%)*
Pure-CNN	8.66[2], 26.20[3]	115.10[2], 37.76[3]	73.86[2,3]	70.26[2,3]
CNN-RAP	8.56[2], 26.16[3]	116.48[2], 37.82[3]	99.93[2,3]	93.69[2,3]
CNN-LSTM with 80 DU	0.40[2], 9.38[3]	2,609.28[2], 103.24[3]	94.83[2,3]	88.30[2,3]
CNN-LSTM with 160 DU	0.40[2], 9.34[3]	2,610.20[2], 103.48[3]	100.00[2,3]	95.00[2,3]

[1] Dense Units (DU), [2] CPU Results, [3] GPU Results, [A] Dataset from [25], [B] Dataset from [25] + [47-51].

Pure-CNN model has a respectable throughput and latency but has the lowest accuracy, due to the prediction flickering effect among consecutive video frames, the severity of which can be lessened by the implementation of RAP. The CNN-RAP model achieves a similar throughput and latency to that of the pure-CNN model, with significant improvement in accuracy, achieving up to 99.9 %. The trend of high accuracy continues with the proposed CNN-LSTM models, with the model trained with 160 dense units achieving the highest accuracy of 100 %. Despite the excellent accuracy, the proposed CNN-LSTM models have 95.3 % lower throughput and over 22 times higher latency using the CPU, while 64.2 % lower throughput and over 2.7 times higher latency using the GPU, as compared to CNN-RAP model.

The CNN-RAP model has an extra computation step, that is to obtain the average results for the past 30 inferences, then only output a label that has the highest probability. Evidently, this additional computation step has a miniscule effect of 1.15 % lower throughput and 1.20 % higher latency using CPU; or 0.15 % lower throughput and 0.16 % higher latency using GPU, as compared to the pure-CNN model as demonstrated in the results. Besides that, the CNN-RAP model processes a single image per inference and the proposed CNN-LSTM models process 30 images per inference. As a consequence, the performance of the proposed CNN-LSTM models will be greatly hampered, resulting in poorer performance when compared to CNN-based models in terms of FPS and latency.

One additional analysis experiment is conducted to assess the generalization capability of the proposed solution. A total of 15 new samples (3 per class) are collected from [47-51] and added to the existing testing dataset. From Table VII, it can be observed that all accuracy performance drops as expected. Nevertheless, the proposed CNN-LSTM still yields the best performance with 95.0% accuracy, demonstrating its robustness.

C. Inference Performance on OpenVINO Environment

In this stage, we evaluate the performance of the optimized models with the same performance metrics used in the TensorFlow environment. Table VIII presents the performance of the optimized models using the same test dataset videos that are used in Section 4.2. It is noteworthy that OpenVINO does not support the NVIDIA GPU, hence the results in Table VIII show the results of various models running in the Intel CPU hardware.

TABLE VIII
PERFORMANCE OF THE VARIOUS MODELS USING TEST DATASET VIDEOS IN OPENVINO ENVIRONMENT

Deep Learning Model	*Throughput (FPS)*	*Latency (ms)*	*Accuracy[a] (%)*	*Accuracy[b] (%)*
Pure-CNN	14.00	71.12	70.80	67.94
CNN-RAP	13.96	71.26	97.70	93.13
CNN-LSTM with 80 DU	0.50	2,094.78	94.83	88.30
CNN-LSTM with 160 DU	0.50	2,095.76	100.00	95.00

[1] Dense Units (DU), [A] Dataset from [25], [B] Dataset from [25] + [47-51].

For a clear and direct comparison, the model inferences are compared using only CPU mode in both the TensorFlow environment and the OpenVINO environment. Referring to the model's performance in Table VIII and comparing with Table VII, the pure-CNN model achieves the performance improvement of 61.7 % higher FPS, 38.2 % lower latency, while losing 4.14 % in accuracy; the CNN-RAP model achieves 63.1 % higher FPS, 38.8 % lower latency, while sacrificing 2.23 % in accuracy; the proposed CNN-LSTM models achieve 25.0 % higher FPS and 19.7 % lower latency with lossless accuracy. A maximum accuracy of 100 % is achieved by the proposed CNN-LSTM model with 160 dense units.

OpenVINO optimization is done by removing or combining the layers of the model to produce an optimized model, hence the drop in the accuracy of the optimized model. Upon the optimization by OpenVINO, some layers are removed to produce an optimized model. For each of the models, five MaxPooling layers, one Flatten layer, and the Dropout layer have been removed as it is mostly used for model training purposes only. The comparisons demonstrate the performance enhancements across all of the models upon OpenVINO optimization compared to TensorFlow's models in terms of frame rate and latency during inferencing. Not only that, the comparisons express the sturdiness of the proposed CNN-LSTM models that its accuracy remains consistent after OpenVINO optimization, either with test dataset from [25] alone or [25] with [47-51]. The reason is that the proposed CNN-LSTM models learn and know the temporal relationship between video frames, resulting in a more stable and robust model.

Overall, the proposed CNN-LSTM models achieve the best accuracy for inferencing in the OpenVINO environment, albeit with low throughput and latency. Moreover, with the implementation of 30-frame RAP to the pure-CNN model, the performance of the CNN-RAP model improves in terms of accuracy, but it is still lower than the proposed CNN-LSTM models.

V. CONCLUSION

Natural disasters continue to pose significant threats to humanity. With the advancement in technology, the integration of AI and DL algorithms have shown promising results in disaster classification performance, paving the way for better disaster management and response. This work developed three types of DL models that accurately classifies four types of natural disaster and non-disaster events. The presented inference results demonstrate that our proposed CNN-LSTM models achieve up to 41.2 % accuracy improvement in classifying different types of disaster events compared to the CNN-RAP model and pure-CNN model. Furthermore, the inference performance of the models achieves up to 63.1 % higher throughput (FPS) and 38.8 % lower latency in the OpenVINO platform compared to the native TensorFlow platform using CPU. Hence, the finding is expected to substantially improve real-time disaster monitoring applications. Nevertheless, the enhancements in throughput and latency achieved through OpenVINO optimization for the CNN-LSTM model are comparatively less substantial when compared to the improvements observed in the CNN-RAP model and the pure-CNN model. This difference can be attributed to the computational intensity of the LSTM layer, which necessitates the processing of extracted features from 30 images generated by the CNN model, resulting in minimal gains in terms of throughput and latency performance. Also, for this very reason, a positive aspect of this circumstance is that the CNN-LSTM model is able to achieve lossless accuracy following the optimization process. Future research directions will be on federated learning for the training phase and distributed learning for the inference phase.

Acknowledgements

This research was supported by the Ministry of Higher Education, Malaysia (MoHE) through Fundamental Research Grant Scheme (FRGS/1/2023/ICT08/UTAR/02/1). The work was also supported by the Universiti Tunku Abdul Rahman (UTAR), Malaysia, under UTAR Research Fund (UTARRF) (IPSR/RMC/UTARRF/2021C1/T05).

References

[1] H. Ritchie, P. Rosado, and M. Roser. Natural Disasters. Our World in Data. 2022. Available online: https://ourworldindata.org/natural-disasters (accessed on 21 March 2023).

[2] K. Okamoto, T. Mochida, D. Nozaki, Z. Wen, X. Qi, and T. Sato, “Content-Oriented Surveillance System Based on ICN in Disaster Scenarios,” *Int. Symp. Wirel. Pers. Multimed. Commun. WPMC*, vol. 2018-November, pp. 484–489, Jul. 2018, doi: http://doi.org/10.1109/WPMC.2018.8712852.

[3] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, “Large-scale video classification with convolutional neural networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1725–1732, Sep. 2014, doi: http://doi.org/10.1109/CVPR.2014.223.

[4] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015. - Conf. Track Proc.*, Sep. 2014, doi: http://doi.org/10.48550/arxiv.1409.1556.

[5] M. Tan and QV. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *International Conference on Machine Learning*, pp. 6105–6114, 2019, doi: https://arxiv.org/abs/1905.11946.

[6] F. Alam, F. Ofli, M. Imran, T. Alam, and U. Qazi, “Deep Learning Benchmarks and Datasets for Social Media Image Classification for Disaster Response,” *Proc. 2020 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Mining, ASONAM 2020*, pp. 151–158, Dec. 2020, doi: http://doi.org/10.1109/ASONAM49781.2020.9381294.

[7] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: http://doi.org/10.1162/NECO.1997.9.8.1735.

[8] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: http://doi.org/10.1109/TNNLS.2016.2582924.

[9] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, “Performance Evaluation of Deep neural networks Applied to Speech Recognition: Rnn, LSTM and GRU,” *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 235–245, Oct. 2019, doi: http://doi.org/10.2478/JAISCR-2019-0006.

[10] S. Mangal, P. Joshi, and R. Modak, “LSTM vs. GRU vs. Bidirectional RNN for script generation,” Aug. 2019, doi: http://doi.org/10.48550/arxiv.1908.04332.

[11] “Model Optimizer Developer Guide - OpenVINO^{TM} Toolkit.” https://docs.openvino.ai/2021.2/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html (accessed Mar. 15, 2023).

[12] N. T. Sze Yang, M. L. Tham, S. Y. Chua, Y. Loong Lee, Y. Owada, and S. Poomrittigul, “Efficient Device-Edge Inference for Disaster Classification,” *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, vol. 2022-July, pp. 314–319, 2022, doi: http://doi.org/10.1109/ICUFN55119.2022.9829668.

[13] K. Muhammad, T. Hussain, M. Tanveer, G. Sannino, and V. H. C. De Albuquerque, “Cost-Effective Video Summarization Using Deep CNN with Hierarchical Weighted Fusion for IoT Surveillance Networks,” *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4455–4463, May 2020, doi: http://doi.org/10.1109/JIOT.2019.2950469.

[14] V. B. Gadhavi, S. Degadwala, and D. Vyas, “Transfer Learning Approach For Recognizing Natural Disasters Video,” *Proc. 2nd Int. Conf. Artif. Intell. Smart Energy, ICAIS 2022*, pp. 793–798, 2022, doi: http://doi.org/10.1109/ICAIS53314.2022.9743035.

[15] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” May 2015, doi: http://doi.org/10.48550/arxiv.1506.00019.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” Dec. 2014, doi: http://doi.org/10.48550/arxiv.1412.3555.

[17] R. Cahuantzi, X. Chen, and S. Güttel, “A comparison of LSTM and GRU networks for learning symbolic sequences,” Jul. 2021, doi: http://doi.org/10.48550/arxiv.2107.02248.

[18] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, “Translating Videos to Natural Language Using Deep Recurrent Neural Networks,” *NAACL HLT 2015 - 2015 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Proc. Conf.*, pp. 1494–1504, Dec. 2014, doi: http://doi.org/10.3115/v1/n15-1173.

[19] J. Donahue *et al.*, “Long-term recurrent convolutional networks for visual recognition and description,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June-2015, pp. 2625–2634, Oct. 2015, doi: http://doi.org/10.1109/CVPR.2015.7298878.

[20] G. Van Houdt, C. Mosquera, and G. Nápoles, “A review on the long short-term memory model,” *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, doi: http://doi.org/10.1007/S10462-020-09838-1.

[21] A. Sahar and D. Han, “An LSTM-based indoor positioning method using Wi-Fi signals,” *ACM Int. Conf. Proceeding Ser.*, Aug. 2018, doi: http://doi.org/10.1145/3271553.3271566.

[22] M. Abdullah, M. Ahmad, and D. Han, “Facial Expression Recognition in Videos: An CNN-LSTM based Model for Video Classification,” *2020 Int. Conf. Electron. Information, Commun. ICEIC 2020*, Jan. 2020, doi: http://doi.org/10.1109/ICEIC49074.2020.9051332.

[23] V. Sharma, M. Gupta, A. Kumar, and D. Mishra, “Video Processing Using Deep Learning Techniques: A Systematic Literature Review,” *IEEE Access*, vol. 9, pp. 139489–139507, 2021, doi: http://doi.org/10.1109/ACCESS.2021.3118541.

[24] Y. H. Yang, J. S. Xu, Y. Gordienko, and S. Stirenko, “Abnormal Interference Recognition Based on Rolling Prediction Average Algorithm,” *Adv. Intell. Syst. Comput.*, vol. 1247 AISC, pp. 306–316, 2021, doi: http://doi.org/10.1007/978-3-030-55506-1_28.

[25] F. Alam, F. Ofli, M. Imran, T. Alam, and U. Qazi, “Deep Learning Benchmarks and Datasets for Social Media Image Classification for Disaster Response,” *Proc. 2020 IEEE/ACM Int. Conf. Adv. Soc. Networks Anal. Mining, ASONAM 2020*, pp. 151–158, Nov. 2020, doi: http://doi.org/10.48550/arxiv.2011.08916.

[26] M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg, “AIDR: Artificial intelligence for disaster response,” *WWW 2014 Companion - Proc. 23rd Int. Conf. World Wide Web*, pp. 159–162, Apr. 2014, doi: http://doi.org/10.1145/2567948.2577034.

[27] F. Alam, F. Ofli, and M. Imran, “CrisisMMD: Multimodal Twitter Datasets from Natural Disasters,” *Proc. Int. AAAI Conf. Web Soc. Media*, vol. 12, no. 1, pp. 465–473, Jun. 2018, doi: http://doi.org/10.1609/ICWSM.V12I1.14983.

[28] 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," *IEEE Access*, vol. 10, pp. 115930–115944, 2022, doi: http://doi.org/10.1109/ACCESS.2022.3218655.

[29] M. F. Mushtaq *et al.*, "BHCNet: Neural Network-Based Brain Hemorrhage Classification Using Head CT Scan," *IEEE Access*, vol. 9, pp. 113901–113916, 2021, doi: http://doi.org/10.1109/ACCESS.2021.3102740.

[30] Capra, M., Bussolino, B., Marchisio, A., Masera, G., Martina, M., & Shafique, M. (2020). Hardware and Software Optimizations for Accelerating Deep Neural Networks: Survey of Current Trends, Challenges, and the Road Ahead. *IEEE Access*, *8*, 225134–225180. https://doi.org/10.1109/ACCESS.2020.3039858.

[31] A. M. R. Abdali and R. F. Al-Tuma, "Robust Real-Time Violence Detection in Video Using CNN and LSTM," *SCCS 2019 - 2019 2nd Sci. Conf. Comput. Sci.*, pp. 104–108, Mar. 2019, doi: http://doi.org/10.1109/SCCS.2019.8852616.

[32] Arif, M. A. Amin, A. A. Ali, and A. K. M. M. Rahman, "Visual attention-based comparative study on disaster detection from social media images," *Innov. Syst. Softw. Eng.*, vol. 16, no. 3–4, pp. 309–319, Dec. 2020, doi: http://doi.org/10.1007/S11334-020-00368-1.

[33] D. R. Hartawan, T. W. Purboyo, and C. Setianingsih, "Disaster victims detection system using convolutional neural network (CNN) method," *Proc. - 2019 IEEE Int. Conf. Ind. 4.0, Artif. Intell. Commun. Technol. IAICT 2019*, pp. 105–111, Jul. 2019, doi: http://doi.org/10.1109/ICIAICT.2019.8784782.

[34] S. Montaha, S. Azam, A. K. M. R. H. Rafid, M. Z. Hasan, A. Karim, and A. Islam, "TimeDistributed-CNN-LSTM: A Hybrid Approach Combining CNN and LSTM to Classify Brain Tumor on 3D MRI Scans Performing Ablation Study," *IEEE Access*, vol. 10, pp. 60039–60059, 2022, doi: http://doi.org/10.1109/ACCESS.2022.3179577.

[35] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: http://doi.org/10.1162/NECO_A_01199.

[36] "Converting a Model to Intermediate Representation (IR) - OpenVINO^{TM} Toolkit.", https://docs.openvino.ai/2021.2/openvino_docs_MO_DG_prepare_model_convert_model_Converting_Model.html (accessed Mar. 15, 2023).

[37] "Converting a Model Using General Conversion Parameters - OpenVINO^{TM} Toolkit.", https://docs.openvino.ai/2021.2/openvino_docs_MO_DG_prepare_model_convert_model_Converting_Model_General.html (accessed Mar. 15, 2023).

[38] Q. M. Areeb, Maryam, M. Nadeem, R. Alroobaea, and F. Anwer, "Helping Hearing-Impaired in Emergency Situations: A Deep Learning-Based Approach," *IEEE Access*, vol. 10, pp. 8502–8517, 2022, doi: http://doi.org/10.1109/ACCESS.2022.3142918.

[39] S. Bernabe, C. Gonzalez, A. Fernandez, and U. Bhangale, "Portability and Acceleration of Deep Learning Inferences to Detect Rapid Earthquake Damage from VHR Remote Sensing Images Using Intel OpenVINO Toolkit," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 14, pp. 6906–6915, 2021, doi: http://doi.org/10.1109/JSTARS.2021.3075961.

[40] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017, Accessed: Jan. 22, 2024. [Online]. Available: https://arxiv.org/abs/1704.04861v1.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 770–778, Dec. 2015, doi: http://doi.org/10.1109/CVPR.2016.90.

[42] U. Elordi, N. Aranjuelo, L. Unzueta, J. L. Apellaniz, and I. Arganda-Carreras, "Optimizing Video Analytics Deployment for In-Flight Cabin Readiness Verification," *IEEE Access*, vol. 11, pp. 92985–92995, 2023, doi: http://doi.org/10.1109/ACCESS.2023.3309050.

[43] "TensorFlow Lite | ML for Mobile and Edge Devices." https://www.tensorflow.org/lite (accessed Jan. 22, 2024).

[44] M. Lebedev and P. Belecky, "A Survey of Open-source Tools for FPGA-based Inference of Artificial Neural Networks," *Proc. - 2021 Ivannikov Meml. Work. IVMEM 2021*, pp. 50–56, 2021, doi: http://doi.org/10.1109/IVMEM53963.2021.00015.

[45] J. Prieto *et al.*, "A Light Vehicle License-Plate-Recognition System Based on Hybrid Edge–Cloud Computing," *Sensors 2023, Vol. 23, Page 8913*, vol. 23, no. 21, p. 8913, Nov. 2023, doi: http://doi.org/10.3390/S23218913.

[46] "openvino2tensorflow PyPI.", https://pypi.org/project/openvino2tensorflow/ (accessed Jan. 22, 2024).

[47] "Disaster Images Dataset.", https://www.kaggle.com/datasets/varpit94/disaster-images-dataset/data (accessed Jan. 22, 2024).

[48] "Strong wind and rain blowing big green tree in bad weather 9933089 Stock Video at Vecteezy." https://www.vecteezy.com/video/9933089-strong-wind-and-rain-blowing-big-green-tree-in-bad-weather (accessed Jan. 22, 2024).

[49] "Aerial view of flooding in a residential area in northern Thailand. River water overflows after heavy rains and floods agricultural area and villages. 11756195 Stock Video at Vecteezy.", https://www.vecteezy.com/video/11756195-aerial-view-of-flooding-in-a-residential-area-in-northern-thailand-river-water-overflows-after-heavy-rains-and-floods-agricultural-area-and-villages (accessed Jan. 22, 2024).

[50] "Beautiful aerial view of nature on the hill of agricultural tourism, Terasering Panyaweuyan, in Majalengka, West Java-Indonesia 9278994 Stock Video at Vecteezy." https://www.vecteezy.com/video/9278994-beautiful-aerial-view-of-nature-on-the-hill-of-agricultural-tourism-terasering-panyaweuyan-in-majalengka-west-java-indonesia (accessed Jan. 22, 2024).

[51] "Forest Fires Near the City 2017963 Stock Video at Vecteezy." https://www.vecteezy.com/video/2017963-forest-fires-near-the-city (accessed Jan. 22, 2024).

[52] G. Xu, T. Ren, Y. Chen, and W. Che, "A One-Dimensional CNN-LSTM Model for Epileptic Seizure Recognition Using EEG Signal Analysis, *Front. Neurosci.*, vol. 14, 2020, doi: http://doi.org/10.3389/fnins.2020.578126.

[53] T. Bao, S. A. R. Zaidi, S. Xie, P. Yang, and Z. -Q. Zhang, "A CNN-LSTM Hybrid Model for Wrist Kinematics Estimation Using Surface Electromyography," *IEEE Trans Instrum Meas*, vol. 70, pp. 1-9, 2021, doi: http://doi.org/10.1109/TIM.2020.3036654.

[54] I. Priyadarshini, and C. Cotton, "A novel LSTM–CNN–grid search-based deep neural network for sentiment analysis," *J Supercomput*, vol. 77, pp. 13911-13932, 2021, doi: https://doi.org/10.1007/s11227-021-03838-w.

Nathaniel Sze Yang Tan received his BEng degree in Electrical and Electronics Engineering from Universiti Tunku Abdul Rahman in 2022. He is currently a Master student at Universiti Tunku Abdul Rahman. His research interests focus on computer vision, machine learning and neural networks.

Mau-Luen Tham received the Bachelor of Engineering and Doctor of Philosophy in the field of Telecommunication Engineering from the University of Malaya. He is currently an Associate Professor with Universiti Tunku Abdul Rahman. His research interests include the IoT, machine learning/deep learning/deep reinforcement learning, and beyond-5G communications.

Sing Yee Chua received the BEng degree in Electrical and Electronics Engineering from the University of Technology Malaysia and the PhD degree in engineering from Monash University. She is currently an Assistant Professor at Universiti Tunku Abdul Rahman. Her research interests include computer vision, signal and image processing, and optical engineering.

Ying Loong Lee received the B.Eng. (Hons.) degree in electronics majoring in telecommunications and the Ph.D. degree from Multimedia University in 2012 and 2017, respectively. He is currently an Assistant Professor in the Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman. His current research interests include wireless communications for 5G and 6G, fixed-mobile convergence and applications of artificial intelligence in telecommunications.