Soft Computing (2025) 29:2363–2378
https://doi.org/10.1007/s00500-025-10512-3

NEURAL NETWORKS AND INTELLIGENT SYSTEMS

A lightweight CNN model for UAV-based image classification

Xinjie Deng[1,2] · Michael Shi[3] · Burhan Khan[1] · Yit Hong Choo[1] · Fazal Ghaffar[1] · Chee Peng Lim[4]

Accepted: 8 November 2024 / Published online: 28 February 2025

Abstract

For many unmanned aerial vehicle (UAV)-based applications, especially those that need to operate with resource-limited edge networked devices in real-time, it is crucial to have a lightweight computing model for data processing and analysis. In this study, we focus on UAV-based forest fire imagery detection using a lightweight convolution neural network (CNN). The task is challenging owing to complex image backgrounds and insufficient training samples. Specifically, we enhance the MobileNetV2 model with an attention mechanism for UAV-based image classification. The proposed model first employs a transfer learning strategy that leverages the pre-trained weights from ImageNet to expedite learning. Then, the model incorporates randomly initialised weights and dropout mechanisms to mitigate over-fitting during training. In addition, an ensemble framework with a majority voting scheme is adopted to improve the classification performance. A case study on forest fire scenes classification with benchmark and real-world images is demonstrated. The results on a publicly available UAV-based image data set reveal the competitiveness of our proposed model as compared with those from existing methods. In addition, based on a set of self-collected images with complex backgrounds, the proposed model illustrates its generalisation capability to undertake forest fire classification tasks with aerial images.

Keywords Lightweight CNNs · MobileNetV2 · UAV images · Forest fire classification · Ensemble learning

1 Introduction

Unmanned aerial vehicles (UAVs) are pilot-less aircraft that find applications across diverse civilian and military sectors. When equipped with computer vision and remote sensing capabilities, UAVs provide fast, mobile, and cost-effective solutions for tackling various problems, e.g., emergency and disaster response, agriculture, traffic management, public safety, wildlife observation, infrastructure inspection and land surveying (Otto et al. 2018). UAVs derive their data collection capabilities from onboard remote sensors, e.g., optical cameras, near-infrared, multi-spectral and hyper-spectral cameras, thermal infrared cameras, laser scanners, and synthetic-aperture radar (SAR) to light detection and ranging (LiDAR) devices (Bouguettaya et al. 2019; Colomina and Molina 2014). As a result, various data types can be collected, e.g., images, videos, and point clouds, to support a multitude of applications. With the aid of navigation sensors and inertial measurement units, geographic coordinate information can be integrated into UAV applications (Yuan et al. 2015).

One key application of UAVs is to perform monitoring of the environment and notify the relevant authorities during

✉ Xinjie Deng
dengxin@deakin.edu.au

Michael Shi
michael.shi789@gmail.com

Burhan Khan
burhan.khan@deakin.edu.au

Yit Hong Choo
y.choo@deakin.edu.au

Fazal Ghaffar
s221455639@deakin.edu.au

Chee Peng Lim
cplim@swin.edu.au

1 Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, 75 Pigdons Road, Waurn Ponds, VIC 3216, Australia

2 Chongqing Jianzhu College, Lihuadadao, Nanan 400072, Chongqing, China

3 Institute for Sustainable Industries and Liveable Cities Victoria University, PO Box 14428, Melbourne, VIC 8001, Australia

4 Department of Computing Technologies, Swinburne University of Technology, John Street, Hawthorn, Victoria 3122, Australia

emergency situations, such as floods, bushfires, hurricanes, volcano hazards, and landslides, mitigating harm to both humans and the environment. In this respect, it is critical to equip UAVs with onboard image/video processing capability to allow real-time monitoring of the environment for the authorities to make early and informed decisions. In recent years, deep learning (DL) has become a popular method for image processing. However, the computational demand of DL models presents a bottleneck for image processing with edge-computing devices, preventing their deployment in UAVs for time-critical missions. As such, lightweight DL models are important for UAVs in carrying out real-time image processing tasks. In this study, we aim to design and develop a lightweight CNN-based model for forest fire detection with UAV images.

As a vital ecosystem component in our environment, forests play a crucial role in sustaining ecological balance. Every year, forest fires devastate millions of hectares of our valuable resources, which not only destroy our ecosystem but also cause significant economic and human resource loss. In 2019, the Amazon rainforest fire burned more than 2 million acres and cost up to 900 billion (Symonds 2019). In 2023, a fire in Hawaii destroyed 17,000 acres, took 98 lives, and caused economic destruction exceeding 5.5 billion (Thebault et al. 2023). As such, governments around the world put great significance on preventing forest fires. In this respect, early and accurate forest fire detection is critical to minimise damage, save lives, and reduce firefighting efforts. If early and accurate information on the onset of a forest fire event can be obtained, firefighters can strive to bring the fire under control before it spreads to a large area, therefore minimising cost and damage. UAV-based surveillance systems offer many advantages for environmental monitoring, which include the ability for data collection on a wide region, even for inaccessible and/or dangerous areas, fast response time, and low running costs.

In forest fire detection, the main obstacles entail the complex background of a forest and the lack of forest fire imagery repositories for DL model training and development. In response to these challenges, this study devises a MobileNetV2 (a lightweight CNN) variant to undertake UAV-based forest fire classification tasks. To reduce the computational load, we employ the MobileNetV2 architecture as the baseline model and enhance it with an attention mechanism. An ensemble framework is also developed to improve classification performance. Our research contributions are three-fold:

- propose an enhanced MobileNetV2 variant with an attention mechanism to enhance its capability in image processing;
- adopt a transfer learning strategy to accelerate the training process and improve the performance of the enhanced MobileNetV2 variant;
- devise an ensemble framework with several enhanced MobileNetV2 variants on a parallel platform to further boost accuracy and speed for forest fire scene classification using UAV-based imagery.

The remaining part of this article is organized as follows: Section 2 reviews CNN models including lightweight CNNs for UAV-based image classification and their applications to forest fire detection. Section 3 explains the proposed model in detail. Section 4 presents the experimental settings and evaluates the model performance using a benchmark data set. Section 5 extends the evaluation to real-world images using a self-collected date set. Section 6 provides the concluding remarks and suggestions for future work.

2 Related studies

The related studies for this research are reviewed and discussed from two perspectives: CNN-based image classification and forest fire detection with CNNs using UAV-based imagery.

2.1 CNN-based image classification

Advances in hardware, such as GPUs and TPUs, and the availability of big data have propelled the recent success and popularity of DL. Indeed, DL models excel at processing large volumes of data and automatically extracting features. This capability makes DL models an effective approach for undertaking various tasks, such as image recognition, natural language processing, etc. There are many reviews on DL models for computer vision, particularly on image analysis and classification (Geetha et al. 2021; Khan et al. 2020; Voulodimos et al. 2018). CNNs are a class of DL models designed to process data with a grid-like topology. Three primary types of layers make up a CNN: convolution layers, pooling layers, and fully connected (FC) layers. Each type of layer serves a distinct function (Raschka and Mirjalili 2017). Many CNNs, such as AlexNet, VGGNet, GoogleNet, ResNet, and DenseNet, have been commonly employed for image classification (Deng et al. 2023; Li et al. 2019; Rawat and Wang 2017).

While deep, wide, and complex networks are proficient at feature extraction, their dependence on extensive hardware resources and high computational demands make them unsuitable for use with devices having limited capacity. For many real-world applications where visual recognition tasks need to be performed in real-time on resource-constrained

mobile devices, the underlying processing models need to be lightweight and efficient (Mehta and Rastegari 2022).

Lightweight networks, such as Xception(Chollet 2017), MobileNets (Howard et al. 2019; Sandler et al. 2018), ShuffleNets (Ma et al. 2018; Zhang et al. 2018), SqueezeNet (Iandola et al. 2016), NASNet Mobile (Zoph et al. 2018), GhostNet(Han et al. 2020), PP-LCNet (Cui et al. 2021), have been widely used in the image classification domain (Kai Heng Lua et al. 2022). There are also network compression strategies such as pruning, low-rank factorization, quantization, and knowledge distillation to compress a model. These lightweight CNNs are versatile and easy to train. They can replace the heavy-weight backbones in existing task-specific CNN models, leading to a reduction in network size and improvement in latency (Mehta and Rastegari 2022).

2.2 Forest fire classification using UAV-images

Benefiting from the capacity to extract high-representative features from images, CNNs have demonstrated promising performances in image processing. Several studies have utilised CNNs for classifying UAV-based forest fire imagery (Bouguettaya et al. 2019; Geetha et al. 2021).

Lee et al. (2017) conducted a comparison of five different CNNs for the classification of forest fires based on images captured by UAVs. These CNNs include AlexNet, GoogLeNet, a modified GoogLeNet (with half the channel number of GoogLeNet), VGG13, and a modified version of VGG (with half the channel number of the original model). They achieved 94.8%, 99%, 86.2%, 96.9%, and 96.2% accuracy rates, respectively. Chen et al. (2019) developed two models for forest fire classification. One used the texture features extracted by local binary patterns (LBP) for smoke detection with a support vector machine (SVM). The other consisted of a 17-layer CNN for flame detection. A real-time forest fire surveillance system using fog computing and UAVs was described by Srinivas and Dua (2020). They adopted an architecture similar to AlexNet for the classification task. Based on several public data sets, the system successfully estimated 95.07% of the fire disasters. Sousa et al. (2020) utilised a pre-trained InceptionV3 model for forest fire classification. Using several public forest fire datasets, they adopted k-fold cross-validation and data augmentation strategies to enlarge the data set to facilitate testing of their method.

Instead of UAV-based images, Govil et al. (2020) proposed a wildfire detection system using camera images. They used InceptionV3 to classify smoke and non-smoke images in wildfire detection. Tang et al. (2021) developed an DL system based on ResNet50 for forest fire classification. In their study, all the training data was labeled as fire, smoke, and others. Another study proposed a multi-label classification model for forest fire detection (Park et al. 2021). The method leveraged transfer learning with several architectures including VGG16, ResNet50, and DenseNet121 for classifying various elements in images, including flames, smoke, non-fire scenarios, and other objects. Sun et al. (2021) proposed a CNN model for classifying forest fire smoke. Batch normalization (BN) and multi-convolution kernels are used by the proposed model to optimise and enhance classification accuracy. They experimented with three multi-channel convolution kernels (3×3, 5×5, and 7×7) in different convolution layers. A self-collected image data set consisting of scene photographs and internet images was used for evaluation (Sun et al. 2021). Zhang Wang et al. (2022) devised a ResNet50-based model for forest fire classification. The activation functions in stages three and four were changed from the rectified linear unit (ReLU) to Mish.

Ghali et al. (2022) proposed a wildfire classification method based on deep ensemble learning using EfficientNetB5 and DenseNet201. The computed feature maps from both models are concatenated before being fed into an average pooling(AP) layer. The images were classified in the layer using a sigmoid function. Khan et al. (2022) explored the performance of a self-collected data set on different machine learning algorithms, including a pre-trained VGG19-based transfer learning model. They assembled a fire data set (i.e., DeepFire), containing 1 900 colour images. The developed method outperformed several machine learning models, including k-nearest neighbours, random forest, naive Bayes, SVM, and logistic regression. YOLO-based models have been primarily designed for object detection tasks, where the goal is to locate and classify objects within an image (Jiang et al. 2022). With some modifications, YOLOV5 can be adapted for image classification. Mahdi and Mahmood (2022) used YOLOV5 with an edge computing infrastructure for a fire and non-fire classification task. They created a wildfire image data set from internet videos. They randomly initialised the weight within the range [0,1] instead of the pre-trained weights from COCO. A dropout layer was used as the last layer to reduce the training time. All the training, validation, and test images were handled at the edge computing platform. Images with fire were uploaded to the cloud server, which served as the platform for making decisions (Mahdi and Mahmood 2022).

While DL models have been used for forest fire detection, only a few of the reported studies emphasize lightweight CNNs. Shamsoshoara et al. (2021) introduced a public aerial image data set, i.e., Fire Luminosity Airborne-based Machine Learning Evaluation (FLAME) and used the Xception network to perform a forest fire classification task. It reached an accuracy rate of 76.23%. Wu et al. (2020) employed the MobileNetV2 model to apply transfer learning for classifying forest fires based on aerial images. The MobileNetV2 model is pre-trained using the ImageNet data set as the wildfire classifier to accelerate training and tackle the lack of large

data sets for DL model training. Data augmentation strategies were also developed, encompassing operations such as flipping, changing brightness, blurring, and adding Gaussian noise to images.

Khan and Khan (2022) designed a MobileNetV2-based model named FFireNet. The model combined a pre-trained MobileNetV2 and FC layers. It was trained and tested on a self-collected data set earning an accuracy rate of 98.42%. NASNet Mobile was evaluated on a forest fire data set containing 1900 self-collected images (Reis and Turk 2023). Transfer learning and fine-tuning techniques were used to improve its performance, as compared with those from other DL models including InceptionV3, DenseNet121, ResNet50V2, VGG19, and traditional machine learning algorithms (SVM and random forest). Xception, MobileNetV2, and ResNet50 were integrated into an ensemble model for feature extraction by Bahhar et al. (2023). A global average pooling (GAP) layer was added to tailor the model for UAV-based forest fire classification.

The United Nations Environment Program - World Conservation Monitoring Centre (UNEP-WCMC) devised a forest category classification system(FAO and UNEP 2020). It categorizes forests in the world into 26 major types, reflecting both climatic zones and major tree species. Owing to diversity in the ecosystem, forest fire detection poses a challenging task. This research contributes towards an accurate forest fire classification solution using UAV-based aerial imagery. It is very important to ensure effectiveness (in terms of performance) and efficiency (in terms of processing time) when designing a model for use in situations where the computing power is limited, such as onboard of UAVs. As such, the focus on this research is on lightweight CNN-based models for forest fire classification using images captured by UAVs. While various CNNs have been used in UAV-based image fire classification, there are limited research studies on lightweight models suitable for implementation in resource-constrained computing platforms. Moreover, most of the existing studies often rely on proprietary data sets, which can affect reproducibility of the results. Therefore, we design and develop lightweight methods that can be effectively deployed on edge computing devices with limited computational resources.

3 Methodology and the proposed model

According to the literature review, CNNs have demonstrated strong performances in aerial image classification tasks. Lightweight CNNs, such as MobileNetV2, are especially notable for their application to resource-constrained platforms, including edge and embedded devices. This study improves MobileNetV2 by incorporating attention mechanisms into its algorithm. The following section presents the methodologies and techniques used in the proposed model.

3.1 Background information on MobileNets

MobileNets are specialised CNNs designed for applications with limited resources, such as edge or embedded devices. The main components of a MobileNet model are discussed as follows.

3.1.1 Convolutional layers

Convolutional layers are composed of a set of convolution kernels for data processing (D'Angelo et al. 2021). An image is first divided into small slices to facilitate the extraction of feature maps. The kernel size influences the receptive field which represents the size of local information to be aggregated. A kernel convolutes with the input image using a specific set of weights by multiplying each element with the corresponding element of the receptive field. A two-dimensional convolution operation with the stride of one is as follows:

$$Y[i,j] = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} X[i+m, j+n]\cdot K[m,n] + b \quad (i = 0, 1, ..., H-1;\quad j = 0, 1, ..., W-1) \tag{1}$$

where, $Y[i, j]$ is the value at position (i, j) in the output feature map, while W and H are the width and height of the output feature image Y. The value at position $(i + m, j + n)$ in the input feature map X is defined as $X[i + m, j + n]$. The height and width of the kernel K are M and N, respectively. The weight at position (m, n) in the convolution kernel is $K[m, n]$; and term b denotes the bias.

3.1.2 Batch normalization

BN improves the training of a neural network by stabilising the distributions of the layer inputs. This is achieved by normalizing the inputs to each layer to control the first two moments (the mean and variance) of the distributions (Ioffe and Szegedy 2015). In a mini-batch $\mathbf{C}$ of size m, there are m values in the mini-batch, where,

$$\mathbf{C} = \{x_1, x_2, \ldots, x_m\}$$

The normalized values are denoted as $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m\}$ and their linear transformations are $\{y_1, y_2, \ldots, y_m\}$. The BN transform can be indicated as, $BN_{\gamma,\beta} : \{x_1, x_2, \ldots, x_m\} \rightarrow \{y_1, y_2, \ldots, y_m\}$ where the mean and variance of this mini-batch are as follow:

$$\mu_{\mathbf{C}} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2}$$

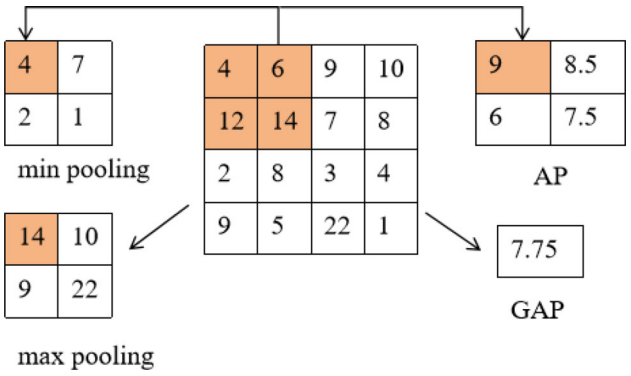


Fig. 1 Four types of pooling layer

$$\sigma_{\mathbf{C}}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathbf{C}})^2 \tag{3}$$

The normalization procedure operates as follows,

$$\hat{x_i} = \frac{x_i - \mu_{\mathbf{C}}}{\sqrt{\sigma_{\mathbf{C}}^2 + \varepsilon}} \tag{4}$$

where ε is a constant added to the mini-batch variance for numerical stability. The final result of the normalization process, i.e., $BN_{\gamma,\beta}$, is computed by applying a linear transformation with a pair of trainable parameters, i.e., γ and β, as follows:

$$y_i = \gamma\hat{y}_i + \beta \equiv BN_{\gamma,\beta} \tag{5}$$

where the adjustment of γ and β helps the model to achieve its optimal distribution for each hidden layer.

3.1.3 Pooling layers

Pooling layers sub-sample the feature maps generated by the convolution operations. A pooling layer shrinks a large-sized feature map to a smaller one. Several types of pooling methods are available, and the most frequently used include max pooling, min pooling, GAP, and AP. Figure 1 illustrates these four pooling operations.

3.1.4 Fully connected layers

The FC layer works as a classifier and is typically positioned at the end of the network. Within an FC layer, each neuron is connected to every neuron in the previous layer. The input of the FC layer comes from the final pooling or convolution layer. Each neuron has an activation function. The quantity of FC layers and their respective neurons rely on the complexity of the research domain and the data set. The number of neurons in the last FC layer matches the count of target classes.

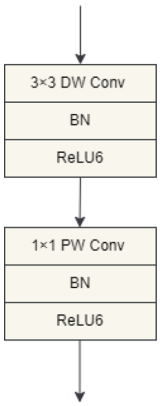


Fig. 2 Depthwise separable convolution with DW, PW, and BN layers and ReLU6

3.1.5 Depthwise separable convolution layers

A depthwise separable convolution (DSC) splits a standard convolution layer into two sub-layers, i.e., the depthwise convolution (DW) layer and pointwise convolution(PW) layer. The DW layer conducts separate convolution operations for each input channel while the pointwise convolution (PW) layer applies a 1×1 convolution layer to combine the DW outputs. Similar to many other network architectures, MobileNets utilise DSC layers. Figure 2 illustrates a DSC layer with DW, PW, followed by BN and ReLU. This factorization reduces the computation time and model size (Howard et al. 2017). Specifically, the computational cost of a standard convolution is as follows,

$$P_{conv} = D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F. \tag{6}$$

where the input feature map size is $(D_F \cdot D_F \cdot M)$, and the output feature map size is $(D_F \cdot D_F \cdot N)$; M and N represent the number of input and output channels, respectively. The kernel is assumed to be square with spatial dimension D_K.

The DSC computational cost is as follows:

$$P_{DSC} = D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F. \tag{7}$$

Equation 7 involves both the DW and PW operations. Converting the standard convolution into a two-step filtering and combination process leads to a reduction in computation cost, i.e.,

$$\frac{P_{DSC}}{P_{conv}} = \frac{1}{N} + \frac{1}{{D_K}^2}. \tag{8}$$

where N is greater than D_K. Therefore, the training cost of a standard 2D convolution layer is ${D_K}^2$ times higher than that of a DSC. In other words, if a 3×3 DSC is used, it consumes eight to nine times lower computational cost than that of a standard convolution layer.

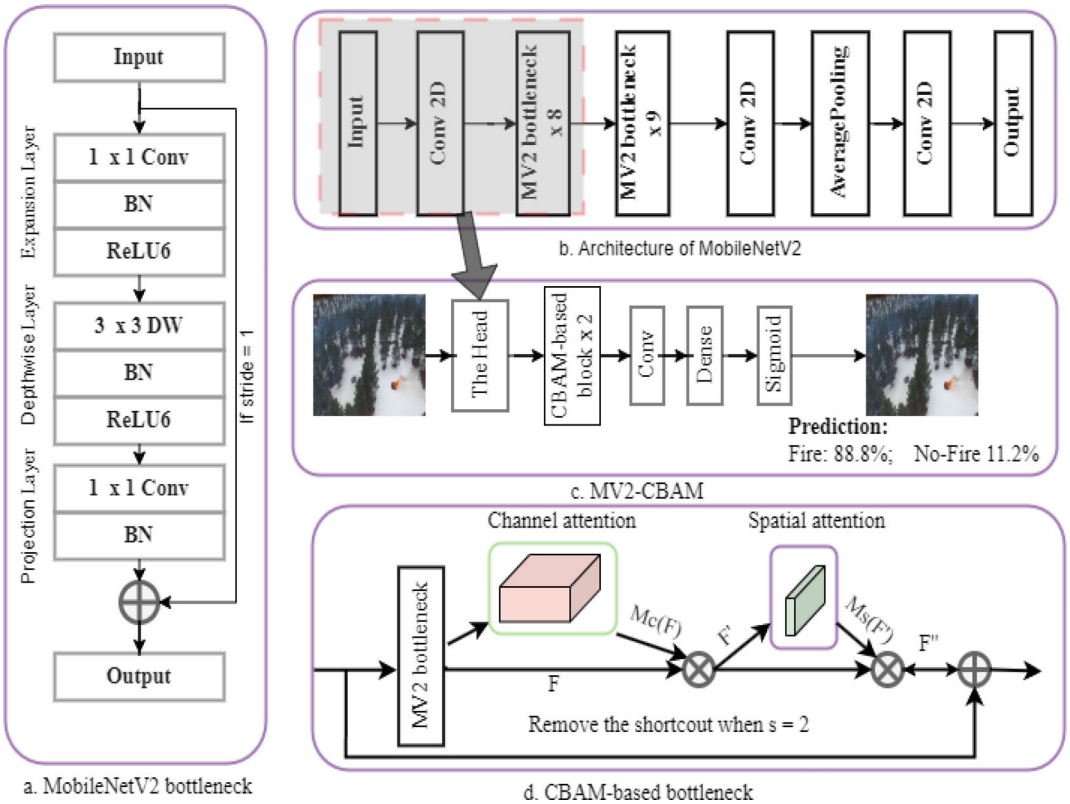


Fig. 3 The proposed architecture

3.1.6 Inverted residual blocks

Besides DSC, MobileNetV2 introduced a linear bottleneck to improve the information-destroying problem about the non-linear layers in the convolution blocks (Sandler et al. 2018). A new structure named Inverted Residuals is designed to preserve the information. This strategy helps MobileNetV2 to be more efficient for implementation in mobile devices. The linear bottleneck is similar to the residual block with a tiny modification. After the DW convolution process and before the projection step, there's a PW convolution layer where the data sample is expanded into a high-dimensional form. This expanded layer is then passed through a linear bottleneck layer, which is essentially a PW convolution layer without a non-linear activation function such as ReLU. By avoiding non-linearity in this stage, the model preserves more information, ensuring that the subsequent projection (which reduces the dimensions) retains as much meaningful information as possible. Unlike traditional residual blocks that first expand the number of channels and then reduce them, Inverted residuals do the opposite. The idea is to first expand the number of channels, process them with lightweight DW convolution, and then project them into a low-dimensional space with a PW (1×1) convolution. This inversion captures more information while using less computation (Sandler et al. 2018). The architecture of MobileNetV2 and the inverted residual blocks are illustrated in Fig. 3.

3.1.7 Activation functions

Activation functions serve to transform input values into corresponding output values. The input value is determined by computing the weighted sum of the neuron's input along with its bias. Activation functions decide whether to free a neuron with respect to a particular input by creating the relevant output. Within a CNN, various activation functions, such as ReLU, facilitate the learning of intricate tasks. ReLU is the most commonly used function in the CNN context. It preserves positive inputs and maps negative inputs to zero, enhancing non-linear mapping. This functional property increases the model's ability to capture and process complex patterns, which enhances its ability to solve problems. The mathematical representation of ReLU is shown in Eq. 9.

$$f(x)_{ReLU} = max(0, x) \tag{9}$$

Notice that ReLU6 is a modified version of ReLU, whereby it limits the output to a maximum value of six. The

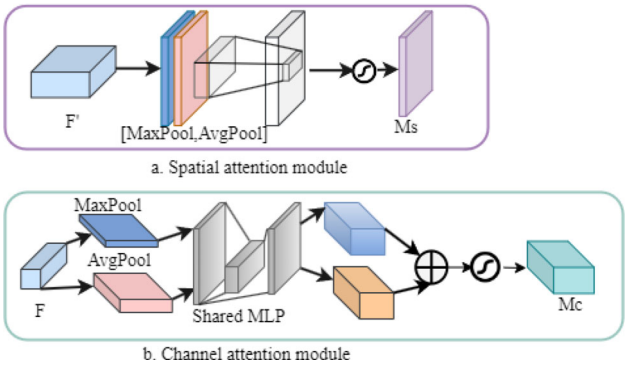


Fig. 4 Diagram of CAM and SAM of CBAM

mathematical representation of ReLU6 is given in Eq. 10.

$$f(x)_{ReLU6} = min(max(0, x), 6) \tag{10}$$

Specifically, ReLU6 provides non-linearity in MobileNetV2, leading to its robustness in low-precision computation.

3.2 Attention mechanism

The attention mechanism enables DL to focus on the most critical information. It has been widely exploited to improve DL performance in computer vision, yielding impressive results in a variety of visual and related tasks, such as image classification, object detection, image segmentation, and self-supervised learning (Guo et al. 2022; Niu et al. 2021). Several lightweight attention mechanism blocks have been developed, including SENet (Hu et al. 2018), style-based recalibration module (Lee et al. 2019), bottleneck attention module (Park et al. 2018), convolutional block attention module (CBAM) (Woo et al. 2018), coordinate attention (CA) module, and efficient channel attention (ECA) (Wang et al. 2020). These mechanisms enhance the model performance at a slightly higher computational cost, making them particularly beneficial in applications with limited computational resources. As a lightweight and general block, CBAM can be integrated into the CNN architectures seamlessly with minimum overheads. It consists of a channel attention module (CAM) and a spatial attention module (SAM), as shown in Fig. 4.

Specifically, the CAM considers the inter-channel relationship of the input features by adopting AP and max pooling outputs with a shared network. The SAM considers the inter-spatial relationships of the input features and generates a spatial attention map through AP and max pooling outputs. Both CBAM modules are arranged sequentially to achieve a good performance. Given an intermediate feature map, the CBAM module sequentially generates the attention maps along with the channel and spatial dimensions. Then, the attention maps are multiplied with the input feature map for adaptive feature refinement. The CBAM formulation is illustrated in Eqs. 11 and 12.

$$F^{'} = M_C(F) \bigotimes F \tag{11}$$

$$F^{''} = M_S(F^{'}) \bigotimes F^{'} \tag{12}$$

where F represents the intermediate feature map and $F^{''}$ denotes the final refined output; M_C is a 1D channel attention map, M_S is a 2D spatial attention map inferred by the CBAM, while $\bigotimes$ indicates element-wise multiplication. From the literature, CBAM-based MobileNet is able to enhance the top-1 classification accuracy on ImageNet by 2.38% and 0.96% compared with those from the original MobileNet and SE-based MobileNet, respectively (Woo et al. 2018). As such, CBAM is useful for boosting model accuracy when it is integrated into MobileNets. In this study, we use CBAM is used to optimise our proposed model, as detailed in the next section.

Table 1 Comparison of MobileNetV2 and MobileNetV3

Network	Madds	Params
V3-Large 1.0	219	5.4M
V3-Large 0.75	155	4.0M
V2 1.0	300	3.4M
V3-Small 1.0	66	3.9M
V3-Small 0.75	44	1.9M
V2 0.35	59.2	1.6M

3.3 The proposed MV2-CBAM model

In this study, we enhance MobileNetV2 for UAV-based imagery classification. We select MobileNetV2 for its smaller size and simpler training mechanism, as compared with the latest MobileNetV3 model. Specifically, MobileNetV2 with a depth multiplier of 1.0 and 0.35 have approximately 3.4 million and 1.6 million parameters, respectively. In contrast, MobileNetV3 Large (1.0) and MobileNetV3 Small (1.0) have approximately 5.4 million and 2.9 million parameters, respectively.

MobileNetV2 has a smaller size, with MobileNetV2 (1.0) and MobileNetV2 (0.35) having sizes of approximately 14 MB and 5 MB, respectively (Howard et al. 2019; Sandler et al. 2018). Comparatively, MobileNetV3 is larger, with MobileNetV3 Large (1.0) and MobileNetV3 Small (1.0) having sizes of approximately 24 MB and 12 MB, respectively. These values highlight the trade-off between model size and the number of parameters. As such, MobileNetV2 is suitable for use with resource-constrained devices. Table 1 summarises the comparison. Furthermore, MobileNetV3 adopts an automated network architecture search (NAS) strategy to

fine-tune the architecture and optimize the model for specific tasks, therefore increasing its computational load during training and making it less appropriate for use in resource-limited environments. In contrast, MobileNetV2 offers a smaller size to facilitate lightweight applications. As such, MobileNetV2 is leveraged as the base model in this study.

MV2-CBAM contains the input, hidden, and output layers. The input layer size is $224 \times 224 \times 3$, and the values of RGB channels are scaled to the range of [0.0, 1.0]. The output layer adopts a sigmoid function for the classification task, as follows,

$$S(x) = \frac{1}{1 + e^{(-x)}} \tag{13}$$

where $S(x)$ represents the output of the sigmoid function S with x as its input. Note that $S(x)$ always lies between 0 and 1, which indicates the probability of object detection. We incorporate two enhancements into the original MobileNetV2 model. The first is to reduce the model depth. The original MobileNetV2 model has been tailored to the 1000-category classification task of ImageNet. Since our research focuses on binary classification, we reduce its depth by decreasing the number of bottleneck layers from 17 down to 10. We denote the resulting model as "MV2-Trim". Secondly, an attention mechanism is introduced. Considering the capabilities of CBAM, i.e., allowing a model to focus on the target object and having a lightweight structure, we integrate it into the last two bottlenecks of the MV2-Trim model, leading to a CBAM-based inverted residual block and the proposed "MV2-CBAM" model, which is shown in Fig. 3. The operational procedure of the proposed CBAM-based inverted residual block is detailed in Algorithm 1.

Algorithm 1 Procedure of the CBAM-based inverted residual block

Input: Feature map $\mathbf{F}_0 \in \mathbb{R}^{H \times W \times C}$
Output: Refined feature map $\mathbf{F}'' \in \mathbb{R}^{H \times W \times C}$
Step 1: $\mathbf{F} \leftarrow$ Inverted residual block($\mathbf{F}_0$)
Step 2: Channel Attention Module
Compute channel attention map $\mathbf{M}_c$

$$\mathbf{M}_c(\mathbf{F}) \leftarrow \sigma(\text{MLP}(\text{AvgPool}(\mathbf{F})) + \text{MLP}(\text{MaxPool}(\mathbf{F})))$$

(Woo et al. 2018)
Refine the input feature map $\mathbf{F}' \leftarrow \mathbf{M}_c(\mathbf{F}) \odot \mathbf{F}$ (Eq. 11)
Step 3: Spatial Attention Module
Compute spatial attention map $\mathbf{M}_s$

$$\mathbf{M}_s(\mathbf{F}') \leftarrow \sigma(\text{Conv}([\text{AvgPool}(\mathbf{F}'); \text{MaxPool}(\mathbf{F}')]))$$

(Woo et al. 2018)
Refine the feature map $\mathbf{F}'' \leftarrow \mathbf{M}_s(\mathbf{F}') \odot \mathbf{F}'$ (Eq. 12)
if Stride == 2 **then**
    $\mathbf{F}'' \leftarrow \mathbf{F}_0 + \mathbf{F}''$
end if
Return: Refined feature map $\mathbf{F}''$

4 Evaluation and discussion

The goal of the experiments presented in this section is to demonstrate and evaluate the effectiveness of the proposed MV2-CBAM model and its ensemble framework for detecting forest fires using UAV-based images.

4.1 Data set and data augmentation

A CNN model requires an extensive number of data samples to optimise the network parameters and weights during its training procedure. Insufficient training data can lead to an over-fitting issue which leads to poor classification results. The number of aerial wildfire images available online is still limited. In this study, the FLAME database (Shamsoshoara et al. 2021) is used. Specifically, a subset of 39,375 RGB frame images, captured by the Zenmuse X4S camera, is employed for training. This subset comprises 25,018 "fire" samples and 14,357 "non-fire" samples. The test data set consists of 8,617 images captured by a Phantom 3 camera, with 5,137 "fire" samples and 3,480 "non-fire" samples, respectively.

Augmentation techniques are widely used to increase the number of samples in a data set. Horizontal flipping takes both rows and columns of an image matrix and flips them horizontally. The outcome of horizontal flipping can be treated as a mirror version of the original image. Rotation augmentation involves rotating images between 1 and 359 degrees on an axis. The degree of augmentation depends on the rotation degree parameter. Slight rotations (1 to 20 degrees) can benefit digit recognition tasks (Shorten and Khoshgoftaar 2019). However, as the degree of rotation increases, label preservation diminishes, which affects the interpretability and performance of the model. In this study, we utilise a standard data augmentation technique, i.e., horizontal flipping and random rotation, to increase the diversity of the training input samples and to further improve generalisation of the model. The objective is to mitigate the effects of unbalanced data samples. Figure 5 shows several samples in the FLAME data set and the augmented images.

The images in the training data set contain RGB channels and each pixel has an intensity value ranging from 0 to 255. All the input images are re-scaled to a range between 0 and 1, aiming to accelerate training and enhance stability.

4.2 Experimental setting

4.2.1 Ensemble framework

Formulating a general and strong learner for tackling different problems is challenging. In contrast, multiple weak learners can be built with the use of ensemble learning to combine their predictions, enabling weak learners to function

(a) Fire data sample

(b) Samples using rotation and augmentation

Fig. 5 Samples in FLAME and sample augmentation

as one strong learner. Indeed, the effectiveness of ensemble learning has been successfully demonstrated in various areas such as medicine and text mining(Müller et al. 2022). In this respect, majority voting is one of the most common and intuitive prediction combination methods in forming a useful ensemble framework (Mienye and Sun 2022). For a binary classification task, the prediction of the $i-th$ classifier is assumed to be either d_{i0} or d_{i1} for each class, respectively. As such, $\sum_{i=0}^{n}(d_{i0})$ and $\sum_{i=0}^{n}(d_{i1})$ indicates the voting score of the two classes, respectively, where $d_i \in (0, 1)$, and n denotes the number of classifiers. The class with the highest score is selected as the final prediction from the ensemble framework. As such, this study formulates individual MV2-CBAM models and their ensemble framework for UAV-based forest fire classification. The procedure of ensemble framework is illustrated in Algorithm 2.

Algorithm 2 Procedure of the ensemble framework

Input: X: the image to be classified
Prediction:
for $i \leftarrow 0$ to n **do**
  $d_{i0}, d_{i1} \leftarrow CNN_i(X)$ ▷ Weights assigned to class 0 and class 1 by classifier i
end for
Ensemble Learning: Weighted Majority Voting
$score_0 \leftarrow \sum_{i=0}^{n} d_{i0}$ ▷ Total score for class 0
$score_1 \leftarrow \sum_{i=0}^{n} d_{i1}$ ▷ Total score for class 1
$label_p \leftarrow \arg\max\{score_0, score_1\}$ ▷ Select the class with the highest total score
Output: X should be classified to be $label_p$

4.2.2 Create voters of the the ensemble framework

According to Glorot and Bengio (2010) and He et al. (2015), the use of random weight initialisation can affect model training. Variation in weight initialisation can lead different models to different local optimal solutions, even when the model structures are identical (Glorot and Bengio 2010; He et al. 2015). Dropout is a common regularisation strategy used to prevent over-fitting during training (Srivastava et al. 2014). It works by randomly dropping a subset of neurons along with their connections from the model during training. Different neurons are dropped in each training iteration, which introduces variability into the learning process. Extensive studies corroborate that dropout enhances the model generalisation capability. Mitigating over-fitting potentially allows the model to adapt to varied data and elevates the model's performance. However, this advantage comes at a cost. The training process can become unstable as random dropout makes the model architecture different after each iteration. In this study, an empirical evaluation is conducted to explore the impact of varying initial weights and the dropout strategy on model training. Specifically, the proposed model is trained for one epoch, and the process is repeated twenty times with random initial weights. The trained models show variation in performance, even when using the same training data set and identical hyperparameters. Figure 6 illustrates the diverse accuracy rates from the twenty models on the same test data set, indicating the influence of random initial weight settings on the model performance.

Since random weight initialisation combined with dropout can produce diverse models, finding an appropriate initial weight setting for model training remains a challenge. However, the variation induced by random weight initialisation and dropout can be exploited to generate an ensemble framework comprising a set of varied models. As individual models can produce different predictions, this diversity is harnessed to create an ensemble framework capable of providing more accurate and reliable predictions. The training procedure for the proposed MV2-CBAM model as the voters is depicted in Algorithm 3.

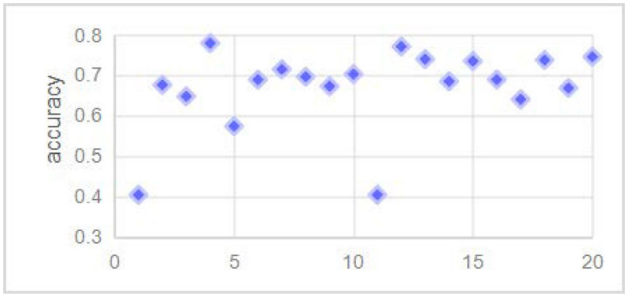


Fig. 6 The performance of identical models trained with random initial weights

Algorithm 3 Training procedure for MV2-CBAM models

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$
Input image: x_i
Related label: $y_i \in \{0, 1\}$
0 : Fire, 1 : No-fire
Initialization:
- x_i resize $\rightarrow$ (224, 224, 3)
- Normalize pixel values (0, 255) $\rightarrow$ (0, 1)
- Data augmentation $\leftarrow$ Figure 6
- Split into train/validation datasets

Hyperparameters:
- Divide D into batches $B = \{B_1, B_2, \ldots, B_m\}$
- Number of epochs E
- Learning rate η
- Total run number R

for $Run \leftarrow 1$ to R **do**
  Randomly initial weight $\rightarrow$ MV2-CBAM
  for $epoch \leftarrow 1$ to E **do**
    for each batch B_i in B **do**
      Train the CNN model on batch B_i
- $B_i \rightarrow$ ConvBNReLU Block
- $\rightarrow$ Eight Inverted Residual Blocks
- $\rightarrow$ Two CBAM-based blocks (Algorithm 1)
- $\rightarrow$ Conv $\rightarrow$ AP$\rightarrow$ Conv $\rightarrow$ Dense
- $\rightarrow$ Sigmoid function
- $\rightarrow$ Get y_{pred} for samples in B_i

    end for
    Loss $\leftarrow \{(y_{pred_i}, y_i)\}_{i=1}^{N}$ (Eq. 14)
    Gradient descent $\leftarrow$ {Adam, SGD, RMSProp,...} & η
    Update the weights
  end for
  Get CNN_{Run} with $(Weight_{Run})$
end for
Output: Trained CNN model $\{(CNN_i)\}_{i=1}^{R}$

4.2.3 Transfer learning

Training a DL model from scratch is time-consuming and demanding. It requires large labeled data sets and extensive computational resources. It often faces over-fitting and convergence issues, requiring iterative adjustments in architecture and parameters (Tajbakhsh et al. 2016). In transfer learning, a model is initially developed and trained for a specific task (source domain) and is then reused for a new, similar problem (target domain) (Park et al. 2021). This method leverages the knowledge gained from initial training and optimise it for a different context. It can produce more successful results than networks trained from scratch. The most common transfer learning procedure in the context of DL comprises four steps:

- Extract a number of layers from a previously trained model;
- Freeze them to preserve the existing information;
- Add new and trainable layers on top of the frozen layers to learn and adapt the old features for prediction in response to a new data set;
- Train the new layers with the new data set.

Transfer learning is usually accomplished for tasks where the target domain does not have sufficient data samples to train a full-scale model from scratch. The fine-tuning strategy is to unfreeze the entire or part of the model and re-train it with the new data set using a low learning rate. As such, transfer learning can achieve improvement by incrementally adapting the pre-trained features to the new data set (Kaya and Gürsoy 2023).

4.2.4 Experimental strategy

An empirical evaluation using the FLAME data set is conducted according to the following procedure. Original and MV2-Trim models, each with random initial weights, are trained with 40 training epochs. The performance after each epoch is computed and the best-performing model is identified. Two ensemble frameworks are formed, comprising three or five individual models. The majority voting scheme is used to combine the predictions from the individual models. The ensemble results and average results from individual models are computed for comparison.

All the experiments are conducted using the National Computational Infrastructure (NCI)[1]. GPU (type: Tesla V100-SXM2-32GB) is used for implementation. The training batch size is set to 32, while the learning rate is set to 0.0001. An early stopping strategy is used to generate the individual models.

A binary cross-entropy loss function is adopted to find the weights of all neurons in the model. The binary cross-entropy measure for a single data sample is shown in Eq. 14.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))\Big) \tag{14}$$

[1] https://nci.org.au/

where y is the target class, i.e., fire ($y = 0$) and non-fire($y = 1$); p is the predicted probability of class with label 1. The training process utilises the Adam optimiser to minimise the loss function and determine the optimal weights.

4.3 Evaluation metrics

To evaluate the developed classification model, the key performance indicators are accuracy and F1-score metrics. F1-score takes both precision and recall rates into account to produce a weighted harmonic mean. Accuracy and F1-score results are calculated as follows(D'Angelo et al. 2023):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$Precision = \frac{TP}{TP + FP} \tag{16}$$

$$Recall = \frac{TP}{TP + FN} \tag{17}$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{18}$$

where TP, TN, FP, and FN denote True Positive, True Negative, False Positive, and False Negative, respectively. In this study, the precision rate is related to the incorrect classification of "non-fire" samples as "fire" samples. The recall rate is the opposite, i.e., incorrect classification of "fire" samples as "non-fire" samples. In other words, precision and recall indicate false-positive and false-negative rates. The F1-score takes into account both precision and recall rates, producing a weighted harmonic mean of the precision and recall metrics.

4.4 Results and analysis

4.4.1 Models performance

Three models are evaluated using the FLAME data set, i.e., original MobileNetV2, MV2-Trim, and MV2-CBAM. Specifically, the initial weights of all three models are based on the weights pre-trained with ImageNet. Each model is further trained for 40 epochs using the FLAME data set. The test results are presented in Table 2. In general, the proposed MV2-CBAM model achieves the best performance, demonstrating that the introduced enhancements are useful for tackling this forest fire imagery classification task.

Table 3 summarises the average accuracy rates and the associated 95% confidence interval (CI) from 30 individual models. The same training and test data sets have been evaluated using Xception (Shamsoshoara et al. 2021), ResNet50 (Zhang Wang et al. 2022), and ensemble EfficientNet-B5 and DenseNet201 (Ghali et al. 2022). For comparison, the published results are extracted from the literature and presented in Table 3.

Referring to Table 3, the mean accuracy score (85.82%) from 30 MV2-CBAM models is better than those published in the literature, namely from 76.23% to 85.12%. Note that MV2-CBAM is smaller in size (i.e., lightweight) as compared with those models in the literature. To assess the ensemble framework performance, we randomly select three individuals from the pool of thirty MV2-CBAM networks as voters and combine their predictions using the majority voting scheme. By repeating the process 1000 times, the mean accuracy rate and its 95% CI are computed. The same approach is conducted with five individual MV2-CBAM networks as voters, which is again repeated 1000 times. The 3-voter and 5-voter ensemble results are shown in Table 3. It is evident from that result that the best performance is from the ensemble framework with 5 voters, i.e. 88.05% mean accuracy. Notice that the 95% CI of the 5-voter ensemble framework (between 87.97% and 88.07%) does not overlap with those from the 3-voter ensemble framework as well as 30 individual MV2-CBAM networks, indicating its stability in performance from the statistical perspective. Figure 7 shows the confusion matrix of the best individual MV2-CBAM model, the ensemble models, and the reported models in the literature.

4.4.2 Inference time

Reliable performance with fast inference time is critical in UAV-based imagery applications. In this study, we adopt lightweight architectures to boost the inference speed. Since we employ an ensemble framework to improve performance, it is important to ensure its computational process is fast enough for UAV-based settings. As such, parallel computing is exploited to improve the computational efficiency. In this respect, two strategies are explored: data parallelism and model parallelism.

In model parallelism, each MV2-CBAM network is allocated to one processor. The results from multiple MV2-CBAM networks across multiple processors are then combined to form an ensemble framework. To use data parallelism for training one MV2-CBAM network, the training sets firstly divided into multiple blocks. Each training data block is handled by one processor. The locally computed results from multiple blocks produced by multiple processors are combined after each training epoch to generate a complete trained model. Note that data parallelism is model-agnostic and can be applied to different network architectures (Shallue et al. 2019). The workflow of data parallelism is illustrated in Fig. 8. In this study, we compare the computational time of sequential and parallel training schemes. The results are presented in Table 4.

Sequential training consumes 1.85 h to train an individual MV2-CBAM model repeatedly in ten epochs. In contrast, it only takes 0.67 h when data parallelism with three GPUs is adopted for single model training, i.e., almost one-third of

Table 2 Comparison of MobileNetV2-based models

Model	Freeze the top	Accuracy	Loss	F1-score	Precision	Recall	Params(M)	Trainable Params(M)
Original	Yes	0.8305	0.560	0.855	0.875	0.835	3.72	1.56
MobileNetV2	No	0.8942	0.397	0.911	0.915	0.907	3.72	3.68
MV2-Trim	Yes	0.6837	3.330	0.783	0.663	0.957	1.12	0.82
	No	0.893	0.164	0.910	0.909	0.912	1.12	1.11
MV2-CBAM	Yes	0.7759	1.170	0.837	0.740	0.964	1.52	1.33
	No	0.9062	0.304	0.959	0.881	0.918	1.52	1.50

Table 3 Performance evaluation of ensemble framework in FLAME data set

Model	Accuracy	
Shamsoshoara et al. (2021)	76.23%	
Ghali et al. (2022)	85.12%	
Zhang Wang et al. (2022)	79.48%	
Proposed model	**Mean Accuracy**	**95% CI of Accuracy**
Average of 30 MV2-CBAM models	85.82%	(85.08%, 86.56%)
Three-voter ensemble of MV2-CBAM	87.36%	(87.28%, 87.44%)
Five-voter ensemble of MV2-CBAM	88.05%	87.96%, 88.07%)

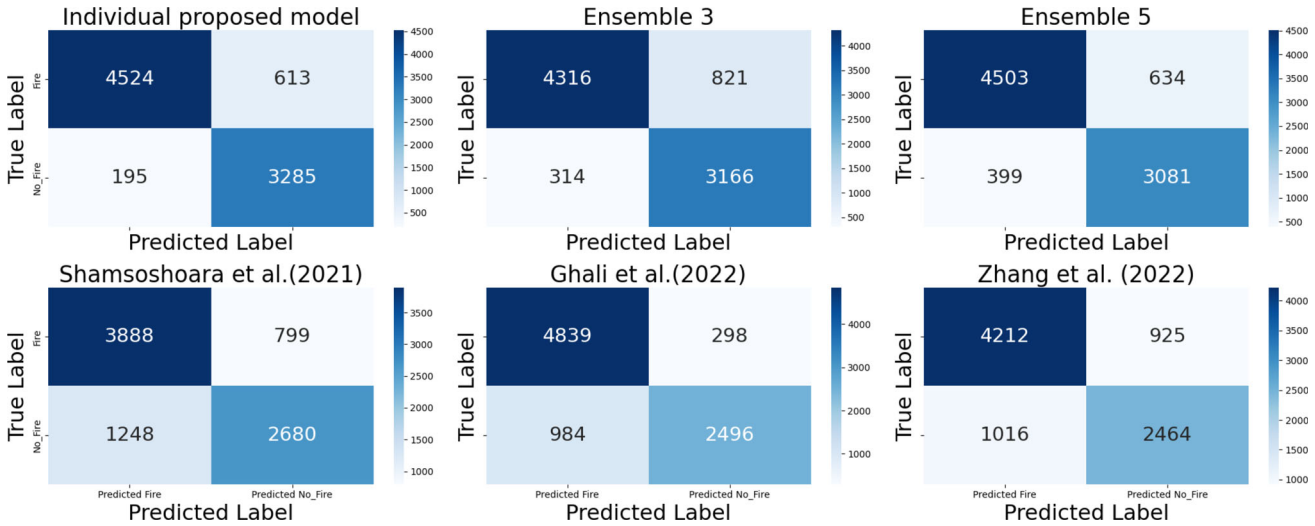


Fig. 7 Confusion matrices for the FLAME

Table 4 Parallel and sequential training time

Setting	Sequential training	Parallel training	
		Data parallelism	Model parallelism
Number of GPUs	1	3	3
Number of MV2-CBAM	1	1	3
Training Time (h)	1.85	0.67	1.37

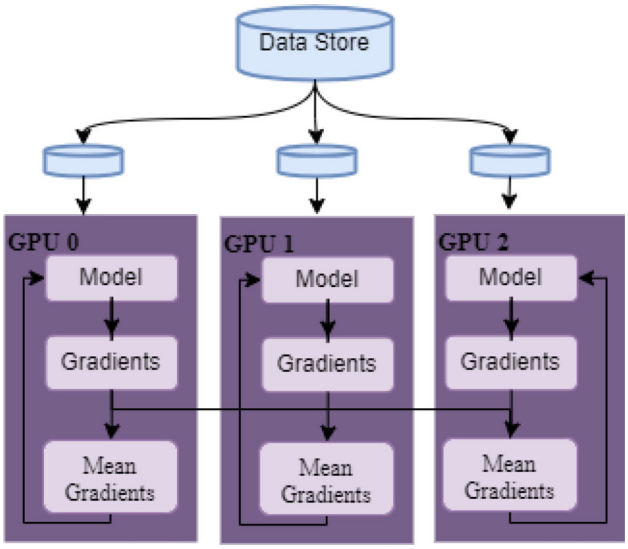


Fig. 8 The procedure of data parallelism

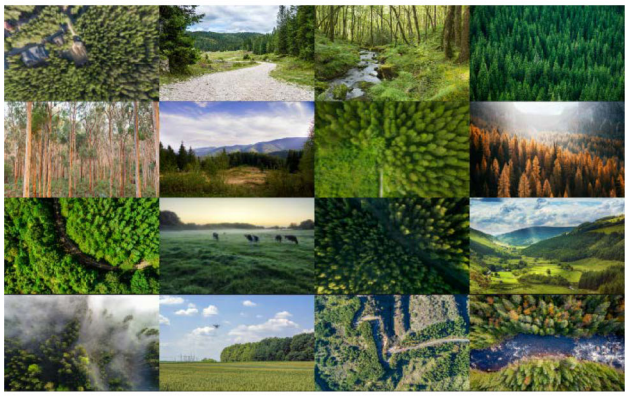
sequential training time. We also perform model parallelism training for the three-voter ensemble framework. As such, three GPUs are used to train three MV2-CBAM models concurrently, and this process consumes 1.37 h. As expected, the results indicate that parallel training offers a practical means for the ensemble framework to ensure its fast processing time while improving performance about the use of lightweight models in UAV-based imagery tasks.

5 Evaluation with self-collected images

To further demonstrate the MV2-CBAM generalisation capability, a set of self-collected images from the Internet is used for evaluation. There are 626 and 578 fire and non-fire samples, respectively, with complex backgrounds ranging from deep forests to wild fields. Figure 9 depicts several fire and non-fire samples. A total of 60% of the images are randomly chosen for training and validation, while the remaining 40% are allocated for test. The original MobileNetV2 and MV2-Trim models are executed for performance comparison with MV2-CBAM. All the experiments are conducted based on network weights pre-trained on the FLAME data set, utilising the early stopping strategy.

Table 5 shows the overall results. All models achieve over 85% accuracy rates. With the pre-trained weight and dropout strategy, MV2-CBAM demonstrates its robustness in classifying this self-collected data set, as compared with the original (87.37%) and trimmed (85.68%) MobileNetV2 models. Specifically, the 30 individual MV2-CBAM models yield 88.65% mean accuracy. Both ensemble frameworks outperform the individual models. For the three-voter ensemble framework, the mean accuracy rate of 1000 repetitions is

(a) Fire samples



(b) No_Fire sample

Fig. 9 Samples in custom data set

Table 5 Performance evaluation on the self-collected data set

Model	Accuracy
Original MobileNetV2	87.37%
MV2-Trim	85.68%
Proposed model	**Mean accuracy**
Thirty individual models	88.65%
Three-voter ensemble of MV2-CBAM	88.79%
Five-voter ensemble of MV2-CBAM	88.83%

88.79%. Meanwhile, for the five-voter ensemble framework, the mean accuracy rate of 1000 repetitions is 88.83%.

It is worth noting that all models can accurately predict non-fire samples, while there are differences in performance in the prediction of fire samples. This observation suggests that the data set is inadequate, particularly in the case of fire samples. And that the models may not yet have converged, making their predictive abilities for unseen images unstable. The results underscore the effectiveness of MV2-CBAM model and the developed ensemble framework for classifying forest fire images with different backgrounds.

It is worth noting that all models can accurately predict non-fire image samples, while there are differences in performance toward predicting fire image samples. A total of 44 fire images with the highest error rates, where more than 2/3 of the ensemble models make incorrect predictions are selected for analysis. As can be observed, in addition to more complex backgrounds than those in the FLAME data set, most of these error-prone fire images contain strong smoke. In contrast, FLAME images do not have such scenes. As such, it is necessary to further improve MV2-CBAM for forest fire classification with complex scenes.

6 Conclusions and future work

In this study, we have enhanced the MobileNetV2 model by integrating an attention mechanism into its operation utilising CBAM blocks. The proposed MV2-CBAM model efficiently captures fire-related features while maintaining a lightweight architecture. To further improve the classification performance, we have formed an ensemble framework comprising multiple MV2-CBAM models with random initial weights. The developed ensemble framework has been evaluated using the FLAME data set and a self-collected real-world forest fire imagery data set. A comparative evaluation against other CNN models has revealed that our MV2-CBAM ensemble framework is able to significantly enhance the accuracy of forest fire imagery classification tasks.

Despite the promising results in fire/smoke detection, several challenges need to be addressed in future research. For future work, it is necessary to:

- enhance the robustness and accuracy of the developed fire/smoke detection model by training with broader and more diverse real-world forest fire images. This will help address the challenge in detecting fires in complex and varied forest environments.
- improve the ensemble framework by incorporating diverse advanced models, including transformer-based models, and CNN-transformer hybrid models. This will help increase the overall detection performance.
- assess the practical applicability of the developed model with UAV-based hardware platforms, such as the NVIDIA Jetson Nano. This will demonstrate and ascertain the effectiveness of the developed solution for forest fire classification in real-world environments.

Data Availability Access to the data supporting the research findings can be obtained by contacting the corresponding author for further information or inquiries.

Declarations

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. This work was supported by the Natural Science Foundation of Chongqing Science and Technology Commission (Grant No. CSTB2023NSCQ-MSX0478) and the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJQN202304315 and KJZD-K202404303).

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Bahhar C, Ksibi A, Ayadi M, Jamjoom MM, Ullah Z, Soufiene BO, Sakli H (2023) Wildfire and smoke detection using staged yolo model and ensemble cnn. Electronics 12(1):228

Bouguettaya A, Kechıda A, Taberkıt AM (2019) A survey on lightweight cnn-based object detection algorithms for platforms with limited computational resources. Int J Inf Appl Math 2(2):28–44

Chen Y, Zhang Y, Xin J, Wang G, Mu L, Yi Y, Liu D (2019) Uav image-based forest fire detection approach using convolutional neural network. 2019 14th ieee conference on industrial electronics and applications (iciea) pp 2118-2123

Chollet F (2017, July) Xception: Deep learning with depthwise separable convolutions. Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)

Colomina I, Molina P (2014) Unmanned aerial systems for photogrammetry and remote sensing: a review. ISPRS J Photogramm Remote Sens 92:79–97. https://doi.org/10.1016/j.isprsjprs.2014.02.013

Cui C, Gao T, Wei S, Du Y, Guo R, Dong S, Ma Y (2021) Pp-lcnet: A lightweight cpu convolutional neural network. Retrieved from https://arxiv.org/abs/2109.15099

Deng X, Khan B, Lim CP, Liao M-Y (2023) A mobilenetv2-cbam-based model for forest fire classification using uav imagery. 2023 ieee 4th international conference on pattern recognition and machine learning (prml) pp 141–146

D'Angelo G, Ficco M, Robustelli A (2023) An association rules-based approach for anomaly detection on can-bus. International conference on computational science and its applications pp 174–190

D'Angelo G, Palmieri F, Robustelli A (2021) Effectiveness of video-classification in android malware detection through apistreams and cnn-lstm autoencoders. International symposium on mobile internet security pp 171–194

FAO and UNEP (2020) The state of the world's forests 2020. forests, biodiversity and people. Rome: FAO and UNEP

Geetha S, Abhishek C, Akshayanat C (2021) Machine vision based fire detection techniques: a survey. Fire Technol 57:591–623. https://doi.org/10.1007/s10694-020-01064-z

Ghali R, Akhloufi MA, Mseddi WS (2022) Deep learning and transformer approaches for uav-based wildfire detection and segmentation. Sensors, 22 (5), https://doi.org/10.3390/s22051977 Retrieved from https://www.mdpi.com/1424-8220/22/5/1977

Glorot X, Bengio Y (2010, 13–15 May) Understanding the difficulty of training deep feedforward neural networks. YW Teh, Titterington M (eds), Proceedings of the thirteenth international conference on artificial intelligence and statistics Vol. 9, pp 249–256. Chia Laguna Resort, Sardinia, Italy: PMLR

Govil K, Welch ML, Ball JT, Pennypacker CR (2020) Preliminary results from a wildfire detection system using deep learning on remote camera images. Remote Sensing, 12(1), https://doi.org/10.3390/rs12010166 Retrieved from https://www.mdpi.com/2072-4292/12/1/166

Guo M-H, Xu T-X, Liu J-J, Liu Z-N, Jiang P-T, Mu T-J, Hu S-M (2022) Attention mechanisms in computer vision: a survey. Comput Vis Med 8(3):331–368. https://doi.org/10.1007/s41095-022-0271-y

Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2020) Ghostnet: More features from cheap operations. Proceedings of the ieee/cvf conference on computer vision and pattern recognition pp 1580-1589

He K, Zhang X, Ren S, Sun J (2015, December) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the ieee international conference on computer vision (iccv)

Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, Adam H (2019, October) Searching for mobilenetv3. Proceedings of the ieee/cvf international conference on computer vision (iccv)

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. Retrieved from https://arxiv.org/abs/1704.04861

Hu J, Shen L, Sun G (2018, June) Squeezeand- excitation networks. Proceedings of the ieee conference on computer vision and pattern recognition (cvpr)

Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. Retrieved from https://arxiv.org/abs/1602.07360

Ioffe S, Szegedy C (2015, 07–09 Jul) Batch normalization: Accelerating deep network training by reducing internal covariate shift. Bach F, Blei D (eds), Proceedings of the 32nd international conference on machine learning Vol. 37, pp 448–456. Lille, France: PMLR. Retrieved from https://proceedings.mlr.press/v37/ioffe15.html

Jiang P, Ergu D, Liu F, Cai Y, Ma B (2022) A review of yolo algorithm developments. Proc Comput Sci 199:1066–1073. https://doi.org/10.1016/j.procs.2022.01.135

Kai Heng Lua W, ChunYu Yau P, Kiat Seow C, Wong D (2022) Lightweight cnn-based deep neural networks application in safety measurement. 2022 5th international conference on pattern recognition and artificial intelligence (prai) pp 455-459

Kaya Y, Gürsoy E (2023) A mobilenet-based cnn model with a novel fine-tuning mechanism for covid-19 infection detection. Soft Comput 27(9):5521–5535

Khan A, Hassan B, Khan S, Ahmed R, Abuassba A (2022) Deepfire: A novel dataset and deep transfer learning benchmark for forest fire detection. Mob Inf Syst 2022:5358359. https://doi.org/10.1155/2022/5358359

Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53:5455–5516

Khan S, Khan A (2022) Ffirenet: Deep learning based forest fire classification and detection in smart cities. Symmetry 14(10):2155. Retrieved from https://www.mdpi.com/2073-8994/14/10/2155

Lee H, Kim H-E, Nam H (2019, October) Srm: A style-based recalibration module for convolutional neural networks. Proceedings of the ieee/cvf international conference on computer vision (iccv)

Lee W, Kim S, Lee Y-T, Lee H-W, Choi M (2017) Deep neural networks for wild fire detection with unmanned aerial vehicle. 2017 ieee international conference on consumer electronics (icce) pp 252-253

Li S, Song W, Fang L, Chen Y, Ghamisi P, Benediktsson JA (2019) Deep learning for hyperspectral image classification: an overview. IEEE Trans Geosci Remote Sens 57(9):6690–6709. https://doi.org/10.1109/TGRS.2019.2907932

Ma N, Zhang X, Zheng H-T, Sun J (2018, September) Shufflenet v2: Practical guidelines for efficient cnn architecture design. Proceedings of the european conference on computer vision (eccv)

Mahdi AS, Mahmood SA (2022) An edge computing environment for early wildfire detection. Ann Emerg Technol Comput (AETiC) 6(3)

Mehta S, Rastegari M (2022) Mobilevit: Light-weight, general-purpose, and mobilefriendly vision transformer. Retrieved from https://arxiv.org/abs/2110.02178

Mienye ID, Sun Y (2022) A survey of ensemble learning: concepts, algorithms, applications, and prospects. IEEE Access 10:99129–99149. https://doi.org/10.1109/ACCESS.2022.3207287

Müller D, Soto-Rey I, Kramer F (2022) An analysis on ensemble learning optimized medical image classification with deep convolutional neural networks. IEEE Access 10:66467–66480. https://doi.org/10.1109/ACCESS.2022.3182399

Niu Z, Zhong G, Yu H (2021) A review on the attention mechanism of deep learning. Neurocomputing 452:48–62. https://doi.org/10.1016/j.neucom.2021.03.091

Otto A, Agatz N, Campbell J, Golden B, Pesch E (2018) Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: a survey. Networks 72(4):411–458

Park J, Woo S, Lee J-Y, Kweon IS (2018) Bam: Bottleneck attention module. Retrieved from https://arxiv.org/abs/1807.06514

Park M, Tran DQ, Lee S, Park S (2021) Multilabel image classification with deep transfer learning for decision support on wildfire response. Remote Sens 13(19), https://doi.org/10.3390/rs13193985 Retrieved from https://www.mdpi.com/2072-4292/13/19/3985

Raschka S, Mirjalili V (2017) Python machine learning: machine learning and deep learning with python, 2nd edn. Packt Publishing, Birmingham

Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput 29(9):2352–2449. https://doi.org/10.1162/neco_a_00990

Reis HC, Turk V (2023) Detection of forest fire using deep convolutional neural networks with transfer learning approach. Appl Soft Comput 143:110362

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the ieee conference on computer vision and pattern recognition pp 4510-4520

Shallue CJ, Lee J, Antognini J, Sohl- Dickstein J, Frostig R, Dahl GE (2019) Measuring the effects of data parallelism on neural network training. J Mach Learn Res 20(112):1–49. Retrieved from http://jmlr.org/papers/v20/18-789.html

Shamsoshoara A, Afghah F, Razi A, Zheng L, Fulé PZ, Blasch E (2021) Aerial imagery pile burn detection using deep learning: the flame dataset. Comput Netw 193:108001

Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):1–48

Sousa MJ, Moutinho A, Almeida M (2020) Wildfire detection using transfer learning on augmented datasets. Expert Syst Appl 142:112975. https://doi.org/10.1016/j.eswa.2019.112975

Srinivas K, Dua M (2020) Fog computing and deep cnn based efficient approach to early forest fire detection with unmanned aerial vehicles. In: Smys S, Bestak R, Rocha Á (eds) Inventive compu-

tation technologies. Springer International Publishing, Cham, pp 646–652
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
Sun X, Sun L, Huang Y (2021) Forest fire smoke recognition based on convolutional neural network. J For Res 32(5):1921–1927. https://doi.org/10.1007/s11676-020-01230-7
Symonds A (2019) Amazon rainforest fires: here's what's really happening. Times, New York
Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, Liang J (2016) Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE Trans Med Imaging 35(5):1299–1312. https://doi.org/10.1109/TMI.2016.2535302
Tang Y, Feng H, Chen J, Chen Y (2021) Forestresnet: A deep learning algorithm for forest image classification. J Phys: Conf Ser 2024(1):012053. https://doi.org/10.1088/1742-6596/2024/1/012053
Thebault R, Brulliard K, Slater J (2023) Hawaii's worst fires leave lahaina in ruins as death toll rises to 53. The Washington Post
Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) Deep learning for computer vision: a brief review. Comput Intell Neurosci. https://doi.org/10.1155/2018/7068349
Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q (2020) Eca-net: Efficient channel attention for deep convolutional neural networks. Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr)
Woo S, Park J, Lee J-Y, Kweon IS (2018, September) Cbam: Convolutional block attention module. Proceedings of the european conference on computer vision (eccv)
Wu H, Li H, Shamsoshoara A, Razi A, Afghah F (2020) Transfer learning for wildfire identification in uav imagery. 2020 54th annual conference on information sciences and systems (ciss) pp 1–6
Yuan C, Zhang Y, Liu Z (2015) A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. Can J For Res 45(7):783–792
Zhang Wang M, Fu Y, Ding Y (2022) A forest fire recognition method using uav images based on transfer learning. Forests, 13(7), https://doi.org/10.3390/f13070975 Retrieved from https://www.mdpi.com/1999-4907/13/7/975
Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: An extremely efficient convolutional neural network for mobile devices. 2018 ieee/cvf conference on computer vision and pattern recognition pp 6848–6856
Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. Retrieved from https://arxiv.org/abs/1707.07012