Emergency Management and Disaster Response: Aerial Image Classification for Drone-Based Emergency Monitoring using Atrous Convolutional Feature Fusion

Daoudi Amir Salah Eddine[*]
Higher School of Computer Science
Sidi Belabbes

Tbahriti Mohammed
Higher School of Computer Science
Sidi Belabbes

June 7, 2024

Abstract

Deep Learning based algorithms can provide state-of-the-art accuracy for remote sensing technologies such as Unmanned Aerial Vehicles (UAVs)/drones, potentially enhancing their remote sensing capabilities for many emergency response and disaster management applications. In particular, UAVs equipped with camera sensors can operating in remote and difficult to access disaster-stricken areas, analyze the image and alert in the presence of various calamities such as collapsed buildings, flood, or fire in order to faster mitigate their effects on the environment and on human population. However, the integration of deep learning introduces heavy computational requirements, preventing the deployment of such deep neural networks in many scenarios that impose low-latency constraints on inference, in order to make mission-critical decisions in real-time. To this end, this paper focuses on the efficient aerial image classification from on-board a UAV for emergency response/monitoring applications. Specifically, a dedicated Aerial Image Database for Emergency Response (AIDER) applications is introduced and a comparative analysis of existing approaches is performed. Through this analysis a lightweight convolutional neural network (CNN) architecture is proposed, referred to as *AsphaltNet*, based on atrous convolutions to process multiresolution features and capable of running efficiently on low-power embedded platforms achieving upto $20\times$ higher performance compared to existing models with minimal memory requirements with less than 1% accuracy drop compared to state-of-the-art models.

***K*eywords** Deep Learning, Convolutional Neural Networks, Emergency Monitoring, Unmanned Aerial Vehicles, Drones, Image Processing, Video Processing, Remote Sensing

1 Introduction

Over the past few years Unmanned Aerial Vehicles (UAVs)/drones have gained considerable interest as a remote sensing platform for various practical applications, such as traffic monitoring [23], search and rescue [31], and precision agriculture [29], and satelite imagery processing [48]. Recent technological advances such as the integration of camera sensors provide the opportunity for new UAV applications such as detect, monitor and analyze passive and active threats and hazards at incident scenes (e.g., fire spots in forested areas, flooding threat, road collisions, landslide prone areas) [2]. In addition, due to their small size UAVs offer fast deployment and can thus be in-the-loop of mission critical decisions to better manage the available resources and improve risk assessment, prevention, and mitigation [32]. However, there is a unique set of constraints that need to be addressed due to the fact that a UAV has to operate in disaster-stricken areas

[*]spartaamir19@gmail.com, https://sparta-portfolio.netlify.app/

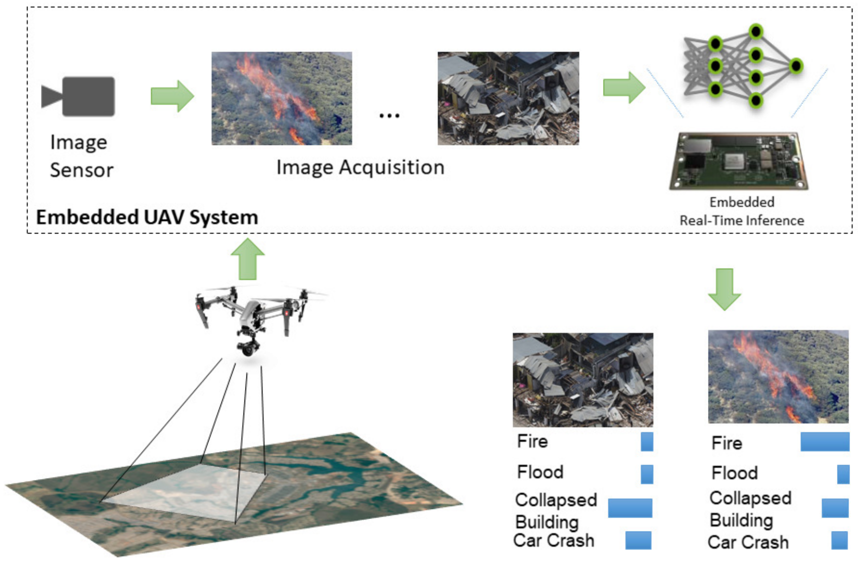


Figure 1: Application scenario for use of UAVs that utilize algorithms based on deep learning models that analyse video footage in real-time to characterizing the current situation and alert in the presence of any danger.

where remote communication to cloud services may not be possible to be established and high-end infrastructure is not available. As a result, a higher level of autonomy is required to ensure operational efficiency and real-time analysis. In such cases an autonomous UAV relies heavily on its on-board sensors and microprocessors to carry out a given task without requiring the feed to be send to a central ground station. Furthermore, the autonomous operation of UAVs by combining path planning algorithms with automated on-board visual processing enables them to cover a larger area in less time [31]. However, on-board processing comes with its own set of challenges due to the limited computational resources and low-power constraints which are necessitated by the low-payload capabilities of UAVs. As such, the computational efficiency of the underlying computer vision algorithm plays a key role in enabling autonomous UAVs to detect hazards at incident scenes in real-time.

Deep learning algorithms such as Convolutional Neural Networks (CNNs) have been widely recognized as a prominent approach for many computer vision applications (image/video recognition, detection, and classification) and have shown remarkable results in many applications [27, 10, 6]. Hence, there are many benefits stemming from using deep learning techniques in emergency response and disaster management applications to retrieve critical information in a timely-fashion and enable better preparation and reaction during time-critical situations and support in-the-loop decision-making processes [30]. Prior works have demonstrated how deep learning approaches can overcome traditional machine learning methods with hand-crafted features through the use of transfer learning where a pretrained convolutional neural network is used as a feature extractor and one or more layers are added on top to perform the classification for the new task [34]. Even though CNNs are increasingly successful at classification tasks, their inference speed on embedded low-power platforms such as those found on-board UAVs is hindered by the high computational cost that they incur [45, 37], especially when considering the need to run multiple vision tasks on the same platform. As such, for many applications local embedded processing near the sensor is preferred over the cloud due to privacy or latency concerns, or operation in remote areas where there is limited or even no connectivity. In addition, purpose built small networks can provide the necessary accuracy and performance for niche applications where abundant data is not available and computational constraints are imposed. Also, beyond the computational efficiency they are faster to go through training iterations and more easily updatable over-the-air. Hence, using a small CNN that is amicable for near-sensor (edge) processing to perform the aerial scene classification on board a UAV becomes a very attractive and sensible alternative to standard approaches.

This work addresses the problem of on-board aerial scene classification which is to automatically assign a semantic label to characterize the aerial image that the UAV captures [46]. With respect to emergency response applications these labels correspond to a danger or hazard that has occurred. Such a system can be deployed into UAVs for automated monitoring and inspection to enhance preparedness and provide rapid situational awareness. The specific use-case under consideration is that a UAV will follow a predetermined path as shown in [31]) and will continuously analyze the frames it receives from the camera through its embedded platform and alert for any potential hazards or dangerous situations that it recognizes as shown in Fig. 1. The objective of this work is to enhance the real-time perception capabilities in such scenarios through the development of a CNN model that provides the best trade-off between accuracy and performance and can operate on embedded hardware that is on-board the UAV or its mobile control station. We propose a lightweight CNN, referred to as *AsphaltNet*, based on depthwise atrous convolutions enabling it to gather multi-resolution features in a computationally efficient manner thus managing to provide adequate trade-off between accuracy and speed for the problem of emergency response monitoring with UAVs. Further, the dataset is significantly enlarged and perform an extended comparison with other models and also propose further optimizations to improve performance in certain use-cases.

The main contributions of this work are summarized as follows:

- A dedicated database for the application of aerial image classification for emergency response which contains a larger number of images compared to existing datasets.
- A novel and computationally efficient CNN (referred to as *AspahltNet*) is proposed that combines multi-resolution depthwise convolutions to simultaneously provide near state-of-the-art accuracy ($\sim 95.7\%$) while being up to $\sim 20\times$ faster,and suitable for low-cost low-power devices.
- Study and analyze the impact of different CNN architectures for the task of aerial scene classification of disasters and evaluation in terms of accuracy, inference speed, and memory.
- In the future: Evaluation of the different models using an actual experimental setup consisting of a UAV with two processing options an embedded computing platform and a mobile ground station.

The rest of this report is structured as follows. Section 2 provides the background on convolutional neural network architectures and outlines previous work on the area of aerial image classification for emergency response and disaster management. Section 3 provides details on the constructed dataset for aerial image classification of disasters as well as the techniques used to develop the proposed network. Section 4 presents an analysis of the different models and techniques as well as evaluation on a real experimental UAV platform. Finally, Section 5 provides concluding remarks and discusses directions for future work in this area.

2 Background and Related Work

2.1 Convolutional Neural Network Architectures and design Approaches

Convolutional neural networks (CNNs) are a biologically inspired machine learning algorithm that can be trained to perform a variety of image detection, recognition and segmentation tasks. CNNs are composed of multiple layers that are trained using stochastic gradient descent with back-propagation in order to learn hierarchical represenations of visual data. In the last decade, a lot of progress has been made on CNN-based classification systems. Numerous architectures have been proposed by the deep learning community fuelled by the need to perform even better in image classification tasks such as the ImageNet Large Scale Visual Recognition Competition (ILSVRC). Typically, the structure of a CNN comprises a feature extractor stage followed by a classification layer. Some of the most important architectures are highlighted next. The reader is referred to the individual papers for more details.

VGG16 [39]: The VGG network has become a popular choice when extracting CNN features from images. This particular network contains 16 CONV/FC layers and appealingly, is characterized by its simplicity. It is comprised only of 3×3 convolutional layers stacked on top of each other with an increasing depth of 2 with pooling layers in between to reduce the feature map size by a factor of 2; and with 2 fully-connected layers at the end, each with $4,096$ neurons. A final dense layer is equal to the number of classes is used for the final classification. A downside of the VGGNet is that it is more expensive to evaluate, and uses a lot of parameters and consequently memory (~ 140MB).

ResNet [9]: This network introduced the idea of residual learning in order to train even deeper CNNs, where the input to a convolution layer is propagated and added to the output of that layer after the operation, thus the network effectively learns residuals. However, it's gain in accuracy comes at a cost of both memory demands as well as execution time. (~ 102MB)

Inception [42] & Xception [7]:The main contribution of this architecture is that it combines many different convolution filters (e.g., $1 \times 1, 3 \times 3, and 5 \times 5$) into a multi-level feature extractor. The output of these filters at the same network level are stacked along the channel dimension before being fed into the next layer. This module is referred to as *inception* module in the network. The specific architecture referred to as *Inception V3* comes form the work of Szegedy et al. [41] which proposed updates to the original inception module to further boost accuracy. The weights for Inception V3 are smaller than both VGG and ResNet, coming in at ~ 96MB but can still be considered too large for embedded applications. For this reason a more optimized variant of the inception family was proposed called *Xception* where the idea of separable convolutions was proposed in an attempt to decrease the computational complexity. A convolution is separated to a depth-wise separable convolution where a filter is applied independently on each channel and then these results are combined through point-wise convolutions. Herein, the focus will be on the *Xception* architecture due to its comparatively higher computational efficiency [7].

MobileNet [13, 35, 12]: Utilizing the idea of separable convolutions the MobileNet family of models manage to offer state-of-the-art performance with reduced computational cost. Essentially, the architecture is a streamlined version of the Xception network that applies a single filter at each input in contrast to the more complicated inception module. Thus is designed can be easily parametrized and optimized for mobile applications. In the latest iteration of MobileNets [12] the authors utilize neural search techniques to further optimize the model architecture, especially for semantic segmentation tasks.

SqueezeNet [14]: This is a CNN designed to be parameter efficient. It utilizes a fire module that combines the output of different convolutional layers. It also fetures a bottleneck layer that reduces the dimensionality of the input feature map to subsequently reduce the computational cost.

ShuffleNet [26]: This approach relies on utilizing different groups of filters each working on a separate channel group of the input feature map which reduces the parameter count of the network. In order to then relate the different channels together it utilizes a process called channel shuffling.

EfficientNet [43]: A new CNN scaling approach is proposed in this work that uniformly scales all dimensions of depth/width/resolution in a principled manner in order to achieve better accuracy and efficiency. The approach can be utilized with any backbone architecture such as *MobileNets* and *ResNets*. In addition, autoML framework is used to automatically search for an optimized architecture. Overall, efficientNets have found to surpass the accuracy of larger models such as *VGG* with an order of magnitude fewer parameters and FLOPS. However, the evaluation of EfficientNets was limited to cloud-centric processing systems.

2.2 Related work on image classification for emergency response and disaster management

In this section some relevant works for the problem of aerial image classification for emergency response and disaster management are described, some of which also target remote sensing with UAVs. Different methods have been proposed over the years to detect various disasters in images such as image-processing-based with thresholds to perform pixel-level classification [47], Gaussian mixture models which require empirical tuning [44], and Support Vector Machines which are cosnidered slow for real-time applications [20]. The success of deep learning and CNNs in particular for different kinds of image analysis tasks has also led the research community to investigate their suitability for such applications. Such approaches have first been proposed for ground robots such as the work in [24] and later also used to interpret aerial images [28].

Deep learning has gained a prominent role as an approach for aerial image classification for emergency response and disaster management applications due to its higher classification accuracy and generalization capabilities. In [18] the authors propose a cloud based deep learning approach for fire detection with UAVs. The detection using a custom convolutional neural network (similar in strcuture to VGG16) which is trained to discriminate between fire and non-fire images of 128×128 resolution. The system works by transmitting the video footage from a UAV to a workstation with an NVIDIA Titan Xp GPU where the algorithm is executed. Of course, in scenarios with limited connectivity there would be difficulties in applying this approach. Overall, the proposed approach achieves an accuracy in the range of $81 - 88\%$ for this task.

In [3] a method is proposed for detecting objects of interest in avalanche debris using the pretrained inception Network for feature extraction and a linear Support Vector Machine for the classification. They also propose an image segmentation method as a preprocessing technique that is based on the fact that the object of interest is of a different color than the background in order to separate the image into regions using a sliding window. In addition, they apply post-processing to improve the decision of a classifier based on hidden Markov models. The application is executed on a desktop computer and not on an embedded device, with clock speed of 3GHz and 8GB RAM average a performance of 5.4 frames per second for 224×224 images. The accuracy was between $72 - 97\%$.

Similarly, the work in [38] also targets fire detection application with deep learning. Specifically, two pretrained convolutional neural networks are used and compared, namely VGG16 [39] and Resnet50 [9] as base architectures to train fire detection systems. The architectures are adapted by adding fully connected layers after the feature extraction to measure the classification accuracy. The different models average an accuracy of $\sim 91\%$ for a custom database with an average processing time of 1.35 seconds on an NVIDIA GeForce GTX 820 GPU.

The work in [49] proposes an approach for wildfire detection from UAV platform. The overall approach comprises of a convolutional neural network called *Fire_Net* consisting of 15 layers with an architecture similar to the *VGG16* network with 8 convolutional, 4 max-pooling, and 2 fully connected layers for recognizing fire in 128×128 resolution images. It is accompanied by a region proposal algorithm that extracts image regions from larger resolution images so that they can be classified by the neural network. The training of the system was performed on an NVIDIA GeForce 840M GPU, while the overall accuracy is $\sim 98\%$ and the average performance is 24 frames-per-second on the particular GPU platform for 128×128 resolution images and without considering the overhead of the region proposal and region selection algorithms.

Perhaps the most relevant related work in terms of application domain is that of Kamilaris et al [17] where a deep convolutional neural network is trained to classify aerial photos in one of 5 classes corresponding to natural disasters. The VGG [39] network is used as the base feature extraction and a fully connected is placed on top of it to perform the transfer learning for the new task. An accuracy of 91% is achieved for a custom test set and on average less than 3 seconds are needed to process an image of 224×224 on an Intel Core i7 machine. Even though the application is similar, in this work the UAV on-board system is considered as the processing platform which has additional constraints with the objective of increasing its autonomy and real-time processing capabilities.

From the literature analysis it is clear that existing approaches targeting aerial scene classification for emergency response and disaster management with UAVs have shown really good results using deep CNN architectures as the main classification approach. In their majority they use existing pretrained networks which adapt through transfer learning for the classification of a single event and primarily utilize desktop-class systems as the main computational platform that remotely process the UAV footage on GPUs. However, in certain scenarios the communication latency and connectivity issues may hinder the performance of such systems necessitating higher autonomy levels for the UAV and on-board processing capabilities [40]. Moreover, the computing limitations of embedded platforms constitute the use of existing algorithms targeting desktop-class systems infeasible. To that end the contribution of this work with regards to state-of-the-art is the evaluation of existing approaches for the task of aerial scene classification for emergency response and disaster management and the introduction of an efficient convolutional neural network suitable for embedded platforms such as UAVs that is capable for on-board classification of multiple disaster events.

3 Deep Learning for Aerial Disaster-Event Classification

This section outlines the process of developing an efficient convolutional neural network suitable for embedded platforms for classifying aerial images from a UAV for emergency response and disaster management applications. Specifically, it details how the training set for this problem was collected, and the different networks used for analysis and comparison both through transfer learning of pretrained networks and custom networks along with the design choices made to develop them. Finally, the details of the training process are provided.

3.1 Dataset Collection

Training a CNN for aerial image classification for emergency response and disaster management applications first requires collecting a suitable dataset for this task. To the best of our knowledge there is only one open sourcet used and publicly available dataset for emergency response applications, this dataset was collected as follows. As such, a dedicated database for this task is constructed referred to as *AIDER* (**A**erial **I**mage **D**ataset for **E**mergency **R**esponse Applications). The dataset construction involved manually collecting all images for four disaster events, namely *Fire/Smoke*, *Flood*, *Collapsed Building/Rubble*, and *Traffic Accidents*, as well as one class for the *Normal* case. Visually similar images such as for example active flames and smoke are grouped together. A finer-level classification is possible but is left as future work.

The aerial images for the disaster events were collected through various online sources (e.g. google images, bing images, youtube, news agencies web sites, etc.) using the keywords "Aerial View" or "UAV" or"Drone" and an event such as "Fire","Earthquake","Highway accident", etc. Images are initially of different sizes but are standardized prior to training. All images where manually inspected to first contain the event that was of interested and then to have the event centered at the image so that any geometric transformations during augmentation would not remove it from the image view. During the data collection process the various disaster events were captured with different resolutions and under various condition with regards to illumination and viewpoint. Finally, to replicate real world scenarios the dataset is imbalanced in the sense that it contains more images from the *Normal* class. Of course, this can make the training more challenging, however, an appropriate strategy is followed to combat this during training which will be detailed in the following sections.

The operational conditions of the UAV may vary depending on the environment, as such it is important that the dataset does not contain only "clean" and "clear" images. In addition, data-collection can be time-consuming and expensive. Hence to further enhance the dataset a number random augmentations are probabilistically applied to each image prior

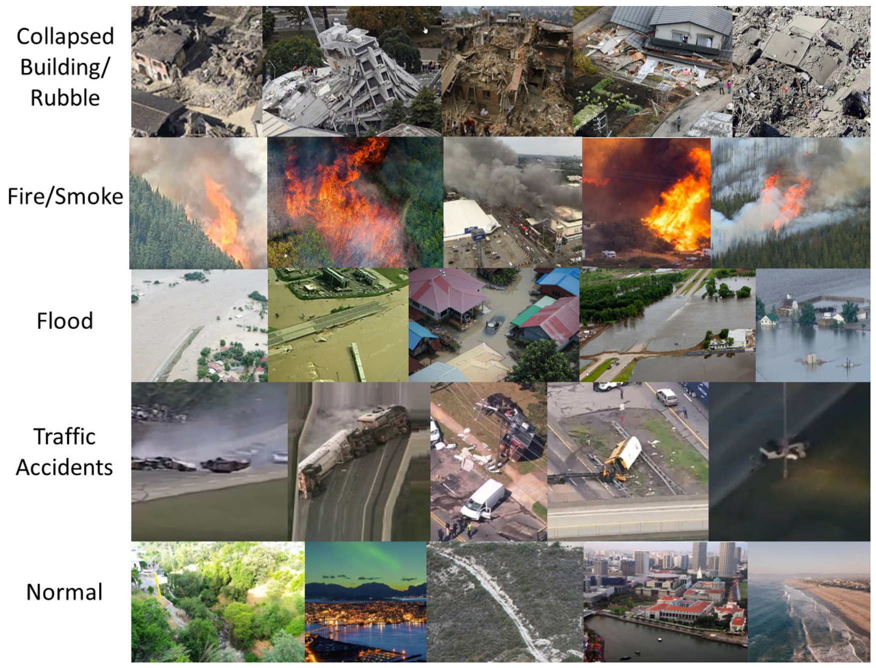


Figure 2: **A**erial **I**mage **D**ataset for **E**mergency **R**esponse (AIDER) Applications: Example images from the database by class.

Table 1: Summary of the **A**erial **I**mage **D**ataset for **E**mergency **R**esponse Applications (AIDER)

Class	**Train Set**	**Validation Set**	**Testing Set**	**Total Per Class**
Collapsed Building/Rubble	400	100	200	700
Fire/Smoke	420	110	220	740
Flood	400	100	200	700
Traffic Accidents	400	100	200	700
Normal	2700	1000	2000	5700
Total per Set	4320	1410	2810	**Overall: 8540**

to adding it to the batch for training. Specifically these are geometric transformations such as rotations, translations, horizontal axis mirroring, cropping and zooming, as well as image manipulations such as illumination changes, color shifting, blurring, sharpening, and shadowing. In addition, sample pairing is also employed to mix images together and further enhance the training set [15]. Each transformation is applied with a random probability which is set in such as way to ensure that not all images in a training batch are transformed so that the network does not capture the augmentation properties as a characteristic of the dataset. The objective of all this transformations is to combat overfitting and increase the variability in the training size to achieve a higher generalization capability. Some samples from the dataset can be seen in Fig. 2. Overall, with respect to the related works that consider multiclass problems (e.g., [17]) almost 17$\times$ more data were collected. The dataset does not contain the amount of images found in common benchmarks such as ImageNet and CIFAR however, it is much more challenging to encounter such real-life events and capture adequate data. As such, augmentations techniques were utilized to enhance the initial dataset even further. This, in our opinion, appoints *AIDER* to a valuable complementary data source for developing and benchmarking data-driven methodologies for emergency response and disaster monitoring applications.

3.2 CNNs for Aerial Disaster Classification

To identify the best structure of the CNN that will perform the aerial image classification a number of different networks was developed using two different approaches. The overall objective of this process is to explore the performance-accuracy trade-offs between these networks. First, transfer learning is employed to train the networks outlined in Section 2.1 which correspond to the methodology used in prior works. Furthermore, new network structures are designed and

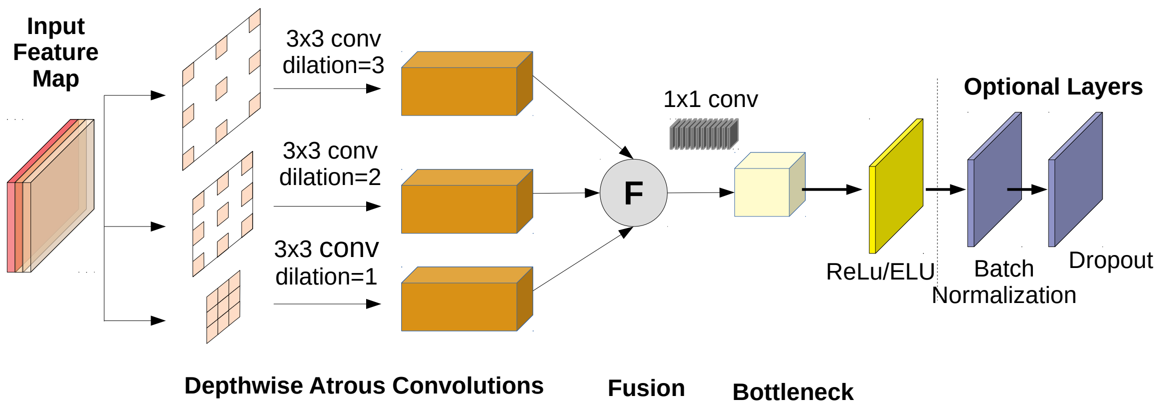


Figure 3: Basic components of multi-resolution Atrous Fusion Block.

trained from scratch specifically for this task. The reasoning behind the latter approach is that it allows making those design choices that lead to more efficient networks that are fast to execute on embedded platforms and at the same time maintain the accuracy of larger networks.

3.2.1 Transfer Learning Networks

For transfer learning different networks from the literature, *VGG16*[39], *ResNet50*[9] *FireNet*[49], *MobileNets*[13, 35, 12], *Xception*[7], *ShuffleNet*[26], *EfficientNet* [43], and *SqueezeNet*[14]. The majority of these networks hace also been used in related works [17, 38, 18].

The feature extraction part is frozen for each of these networks, applying all necessary preprocessing steps to the input image, and add a classification layer on top similar to prior works. In contrast to other works a global (per feature-map) average-pooling layer is applied prior to the dense layers followed by a softmax classification layer at the end. The average pooling reduces the parameter count and the subsequent computational and memory requirements and has shown to perform equally as well with the traditional approaches [25]. Hence, the pretrained models used for comparison are inherently more efficient in terms of memory and operations that the networks used in the literature for this task, which utilize fully connected layers.

3.2.2 Custom Networks

Fine-tuning pretrained networks has some critical limitations. The preterained networks have different degrees of sensitivity depending on how the dataset is similar to the large-scale dataset used (e.g., ImageNet) [21]. The larger and deeper networks attained through transfer learning may not be suited for resource-constraint systems such as UAV platforms, which impose limitations of the size of the platform, the weight, and its power envelope. Furthermore, it is inconvenient to change the architecture of existing networks since the pretraining should be re-conducted on the large-scale dataset (e.g., ImageNet), requiring high computational cost. For this reason there is a need to design specialized networks that are inherently computationally efficient to eliminate the aforementioned limitations [50].

The design space is explored by focusing on the layer configurations, type and connectivity. Consequently different networks are developed to better understand the trade-offs involved in the design choices. There are some systematic design choices that are made across the different network configurations. As a primary building block an architecture is developed that relies on fusing features produced by atrous convolutions of different degrees of dilation [5] and such architectures have primarily been used for segmentation tasks. They are employed herein as a means to simultaneously learn features at various resolutions more efficiently thus facilitating UAV applications which encounter areas of interest at different scales. The proposed architecture allows for flexible aggregation of the multi-scale contextual information while keeping the same resolution and reduced number of parameters. The Atrous Convolutional Feature Fusion (ACFF) block is detailed next.

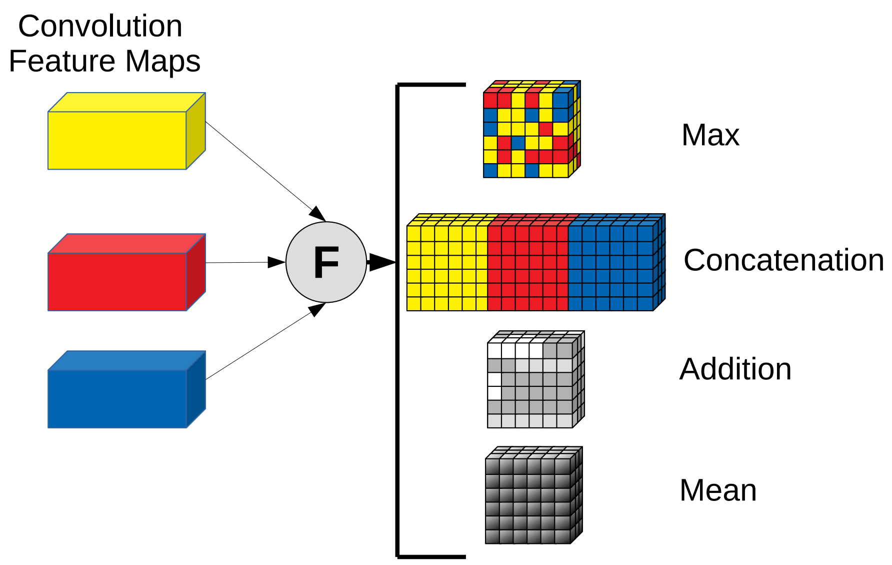


Figure 4: The different fusion schemes tested for the network configurations. The fusion nodes are instantiated after the atrous convolution blocks with element-wise add,mean,max operations, and a concatenation operation.

3.3 Atrous Convolutional Feature Fusion (ACFF)

Atrous (also called dilated) convolutions [5] can capture and transform images at different resolutions depending on the *dilation rate* which determines the spacing between the kernel points, effectively increasing their receptive field without increasing the parameter count. Hence, it can be used to incorporate larger context to the model. The proposed block (Fig. 3) computes multiple such atrous convolutional features ($\mathsf{U_d}$) for the same input map as shown in Eq. (1) across different dilation rates d. Each atrous convolution is factored into depth-wise convolution that performs light-weight filtering by applying a single convolutional kernel per input channel to reduce the computational complexity. Then some form of fusion takes place to merge the different features together as shown in Eq. 2. The intuition is to take advantage of the different dilation rates since one path may peek up features that another may have missed due to changes in object/region resolution. It is important to note that the weights are not shared between paths and each learns different weights $\mathbf{w_d}$ that may be more useful. Another advantage of using atrous convolutions stems from the fact that the same number of parameters and computations are needed regardless of the resolution. Each atrous convolution acts on the same feature map but at a different spatial resolution; starting from a dilation rate of 1 and filter size of 3×3 (i.e., no spacing) and going up to 3 which is equivalent to 7×7 receptive field effectively looking at the same area with different kernels but with less number of parameters utilized as shown in Table 2.

An essential part of optimized CNNs is reducing not only the spatial size of feature maps but also the channel dimensions. Hence, prior to the atrous convolutions the input feature map channels are halved. This makes it possible to have multiple branches for atrous convolution without significantly impacting the performance. The depth reduction factor is a hyperparameter that can be further tuned depending on the requirements. Therefore, a bottleneck layer is utilized to reduce the number of channels after reshaping without changing the spatial size of the feature maps.

The atrous convolutional features at different dilation rates need to be combined together to allow the unit to learn from representations from a large effective receptive field. Four different fusion maps are examined *maximum*, *addition*, *concatenation*, and *averaging* as shown in Fig. 5. The fusion mechanism is then followed by 1×1 convolutions and activation that non-linearly combine channel features together and projects them into a higher dimensional space. Finally, each atrous convolutional block is followed by batchnormalization and an activation.

$$\mathsf{U_d} = \sum_{i=1}^{M}\sum_{j=1}^{N} \mathsf{X}(m + d \times i, n + d \times j) \odot \mathbf{w_d}(i,j) \tag{1}$$

$$\mathsf{Z} = F_{\substack{1 \le m \le M \\ 1 \le n \le N \\ 1 \le d \le N_d}}(\{\mathsf{U_d}(m,n)\}) \tag{2}$$

$$d \in [1, \ldots, N_d]$$

Table 2: Atrous Vs Normal Convolution Savings

Type	Effective Receptive Field	Parameters
Normal Depthwise	3×3	9
Atrous Depthwise	3×3	9
Normal Depthwise	5×5	25
Atrous Depthwise	5×5	9
Normal Depthwise	7×7	49
Atrous Depthwise	7×7	9
Normal Depthwise	11×11	121
Atrous Depthwise	11×11	9

3.4 Macro-Architecture Design Choices

The ACFF macro block is used as a starting point to build a deep neural network that is characterized by low-computational complexity and is suitable for embedded platforms. The following design choices are made for the overall network structure which is illustrated in Table 3.

- **Reduced Cost of First Layer**: The first layer typically incurs the higher computational cost since it is applied across the whole image. Hence, a relatively small number of filters is selected (16) with spatial resolution of 3×3. A standard convolution layer is used here to better capture low-level image features. A stride of 2 is used to reduce the computations.
- **Early downsampling**: Downsampling is performed at all the initial layers. A combination of stride and max-pooling layers are used in an effort to reduce the loss of information by aggressive striding, but still reduce the spatial resolution. It was empirically found that downsizing the feature maps in the latter stages resulted in decreased accuracy hence, the downsampling is performed in the first four layers.
- **Canonical Architecture:** To keep the representational expressiveness a pyramid-shaped form is adopted for the CNN configuration, which means a progressive reduction of spatial resolution of the feature maps at each layer with an increase of their depth. It is quite typical for large networks to have even thousands of filters at each layer, however, for embedded applications this adds considerable overhead. Hence, the first layer has 16 filters, which are then increased thereafter but does not go beyond 256 which is the final layer prior to the classification part.
- **Fully Convolutional Architecture:** A simple and effective trick is utilized to massively reduce the parameter count and computational cost by avoiding the use of fully connected layers. Instead, a channel-wise 1×1 convolution is used to reduce the channels to the number of classes, followed by a global average pooling operation that summarized the per class feature maps and which is fed to a softmax layer.
- **Capped leaky relu:** A capped version of ReLU is used, similar to [35], as the main non-linearity across the network due to its robustness when used with low-precision computation. Specifically, the output is upper bounded at 255 which can be approximated with an 8 bit integer. Even though quantization is not not exploited here, by utilizing this ReLU variant, the network is amicable to further improvements. Moreover, the modified capped leaky ReLU has two modes of operation. During training phase it allows the small fraction of gradient to flow, whereas in the inference phase that part is zeroed out allowing for quantization if necessary.
- **Regularization:** Due to the relatively small size of the dataset compared to databases such as ImageNet; additional regularization techniques are also incorporated beyond augmentation to combat overfitting. In particular, *batch normalization* [16] is used after the atrous convolutions, 1×1 convolutions, as well as depth reduction operations. A *dropout* layer with rate 0.2 is applied after the layers which have the highest number of parameters. As with other works L2 weight regularization with a parameter of 5×10^{-4} was found to be effective in helping the network learn faster and achieve lower error rates.ca
- **Network Depth:** Deep networks are necessary to build strong representations but are also predicated on having a huge amount of data. Also very deep networks incur a higher computational cost. Given these two factors it was empirically found that a network size of 6 ACFF blocks was sufficient to achieve comparable accuracy to the sate-of-the-art, while increasing it did not result in significant accuracy improvements but incurred higher computational cost.
- **Skip Connection Fusion:** A significant amount of information can be lost during the aggressive downwsampling process. To preserve some of the initial information the input feature map is also included into the fusion process.

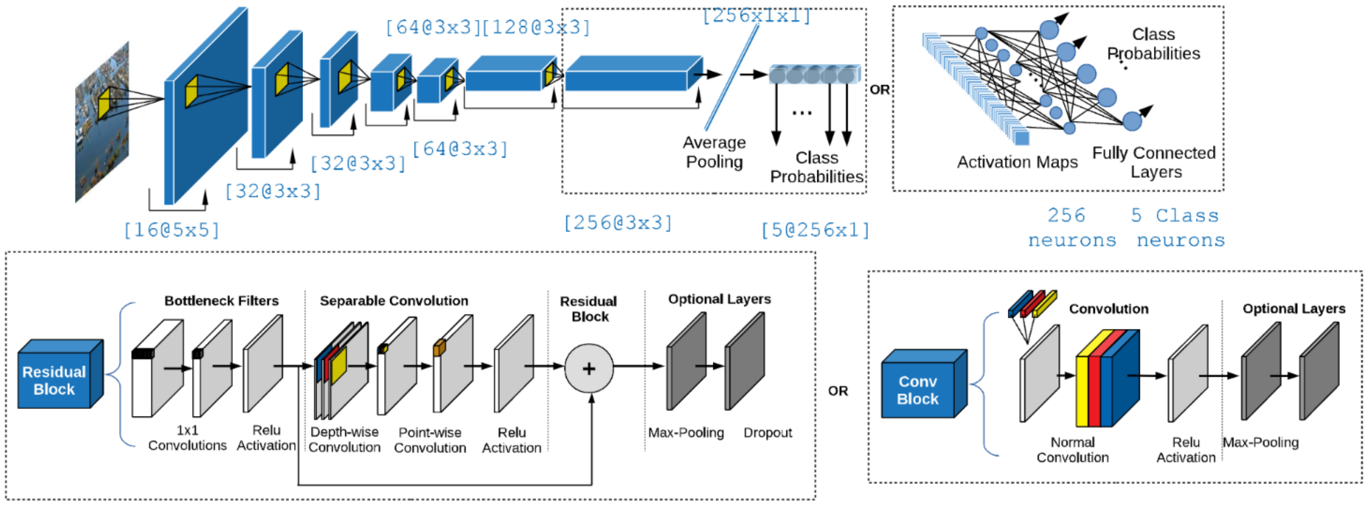


Figure 5: The different Configurations used to develop our model AsphaltNet.

Table 3: AsphaltNet Model Structure

Layer	**Output Size**	**Receptive Field**	**Number of Filters**	**Stride**
Input image	240×240			
Convolution	120×120	3	16	2
ACFF Block 1	120×120	$3, 5, 7$	64	1
MaxPooling 1	60×60	2		2
ACFF Block 2	60×60	$3, 5, 7$	96	1
MaxPooling 2	30×30	2		2
ACFF Block 3	30×30	$3, 5, 7$	128	1
MaxPooling 3	15×15	2		2
ACFF Block 4	15×15	$3, 5, 7$	128	1
ACFF Block 5	15×15	$3, 5, 7$	128	1
ACFF Block 6	15×15	$3, 5, 7$	256	1
Convolution	15×15	1	5	1
Global Pooling	5			
Softmax	5		5	

The aforementioned configurations are combined to build the base *AsphaltNet* architecture shown in Table 3. Different modes of fusing together the feature maps have been evaluated along with base implementations of standard convolution, depth-wise convolution networks, and spatially-separable convolutions in order to compare and contrast the trade-offs. These results are presented in section 4.2.

3.5 Training

All the networks are developed and tested through the same training framework so as to have the same conditions and a fair comparison during the inference phase. The Keras deep learning framework [8] is used which has available all models with Tensorflow [1] running as the backend. The same image size is used for all networks where possible (except for the *mobileNet* V1 and V2 models which specifically require a smaller image size). Consequently, before augmenting and adding an image to the batch it is first resized to the appropriate image size depending on the network (default is 240×240 pixels which is a typical size for training CNNs). It should be noted that it is possible to use larger image sizes at a cost of slower inference time, however in this work the image size space is not explored but rather focus is on the network design.

Algorithm 1 Algorithm pseudocode for balanced training

```
 1: Inputs: $batch\_size = 64, num\_of\_classes = 5, cnn\_model$
 2: Output: Trained CNN Model
 3: function Train($cnn\_model, batch\_size, num\_of\_classes$)
 4: $amount = \lceil \frac{batch\_size}{num\_of\_classes} \rceil$
    { % When the $batch\_size$ is not divisible to the $num\_of\_classes$ some class will be chosen each round to have an
    additional sample at random.}
 5: while Training do
 6:    $Batch = []$
 7:    for class $i$ do
 8:       $class\_images \leftarrow$ get_class_samples($i$,$amount$)
 9:       $class\_images \leftarrow$ augment_images($class\_images$)
10:       $Batch \leftarrow$ add_to_batch($class\_images$)
11:    end for
12:    cnn_model$\leftarrow$ fit($cnn\_model$,$Batch$)
13: end while
14: return cnn_model
```

The first step in the training process is to split the dataset into training, validation, and test sets. The bulk of the data are allocated to the training set and the rest between the other two sets in a $0.5, 0.2, 0.3$ ratio. As mentioned prior, the *Normal* class is the majority class and thus is over-represented in the dataset. This reflects real-world conditions, however, if not addressed, it can potentially lead to problems where the network overfits and thus classifies everything as the majority class. To avoid issues due to the dataset imbalance the simultaneous use of majority class undersampling with oversampling of the minority classes within the same batch is performed. To do this the process outlined in Algorithm 1 is performed to select the same number of images form each class to form a batch and this way all cases are equally represented. Another reason why this process is preferred is that the majority of images were extracted from videos. Even if the videos are sparsily sampled there exists a dependency between images in close temporal proximity. Hence, through random sampling it is less frequent to encounter visually similar images during the training process.

All the networks where trained using a GeForce RTX 3070 Overclocked GPU, on a PC with an Intel $Ryzen76800H$ processor, and 32GB of RAM. The standard Adam optimization method [19] was used for training with a cosine learning rate annealing schedule, and Leaky RELU for the Standard Activation Function. The initial learning rate was set to 0.1 and the total amount of epochs is 300, thus the learning rate LR at time *t* is calculated as shown in Eq. 3. During training label smoothing is also applied with an *epsilon* of 0.1. Each epoch runs for 60 iterations with a batch size of 64 resulting in a total of $3,840$ augmented images per epoch.

$$LR(t) = 0.5 * (1 + \cos(((t * \pi)/(300)))) * 0.1 \tag{3}$$

4 Experimental Evaluation and Results

In this section, the experimental evaluation of the proposed methodology is discussed with results from the experimental evaluation of the approach on an actual embedded platform. First the improvements over existing networks are validated on the developed dataset and qualitative results of the learning process are presented. Real experiments have also been conducted in two different settings: (i) On-board embedded processing where all the computations are performed on-board the resource-constrained UAV platform, on an embedded device. (ii) Remote-based processing in which the UAV transmits the captured video to the controller ground station for processing on an Android tablet that controls the UAV. It is worth noting that the primary interested is in single image processing speed and as such the evaluation phase is carried our with a batch size is 1, since this is common in real-time streaming applications where the camera outputs each frame sequentially.

4.1 Performance Metrics

An important performance metric for real-time applications is the resulting frame-rate or frames-per-second (FPS) achieved by each model, which is inversely proportional to the time needed to process a single image frame from a video/camera stream. In addition, since the prior distribution over classes is significantly nonuniform a simple accuracy measure (percentage of correctly classified examples) which is used in related works, may not be appropriate for the specific problem considered in this work since usually the normal case is much more frequent than other classes. To avoid this bias in our results the F1 score [11] is employed as the learning performance metric instead. The key metrics are summarized below:

4.1.1 Frames-Per-Second (FPS)

The rate at which a classifier is capable of processing incoming camera frames, where t_i is the processing time of a single image.

$$FPS_{model} = \frac{1}{\frac{1}{N_{test\_samples}} \times \sum_{i=1}^{N_{test\_samples}} (t_i)}, \tag{4}$$

4.1.2 Mean F1 Score ($\overline{F1}$)

The F1 score metric finds the mean accuracy across all classes weighted by the number of test instances for each label denoted as $|C_i|$. F1 score conveys the balance between the precision and the recall. In general a good F1 score means that you have low false positives and low false negatives. This handles the label imbalance problem and provides a more appropriate measure of the performance across classes.

$$\overline{F1}_{model} = \frac{2}{N_{classes}} \times \sum_{i=1}^{N_{classes}} \frac{prec_i \times sens_i}{prec_i + sens_i}, \tag{5}$$

4.2 Overall Performance Analysis and Comparison

Initially, the custom models with different fusion mechanisms with the main architecture presented in Section 3.2.2 are compared against vanilla CNN implementations of standard convolution networks as well as networks composed of depthwise- and spatially- separable convolutions. The different networks are compared against parameter size which determines the learning complexity of the model, memory for storage of the model weights, and accuracy with the results summarized in Table 4.

First the standard convolutional neural network has the higher amount of parameters and subsequently memory demands compared to other models. It achieved the highest accuracy amongst the standard networks with the highest demands for memory and parameter count. The depthwise-separable convolution network reduces these demands however, it has a considerable accuracy drop. The spatially-separable network increases the accuracy but with additional parameters needed. On the other hand, the proposed networks with different fusion mechanisms provide higher or on par accuracy. due to the limited number of parameters they are computationally and memory efficient while the combination of multiple features from different kernel sizes manages to provide a considerable accuracy increase. The *maximum* and *add* fusion networks perform better for this particular dataset, amongst the ACFF-based networks. This can be attributed to the fact that they give more emphasis to specific spatial locations which helps in the learning rather than *average* fusion which suppresses information and may reduce some important features. Also they do not increase the memory and processing requirements which is the case in *concatenation* fusion. The best performing network with *add* fusion is selected as the main architecture of *AsphaltNet* which is used for comparison with other networks and further experimental results.

The results for all networks are summarized in Table 5. For these networks comparisons are also made with respect to Floating Point Operations (FLOPS) to measure complexity as a platform independent metric. First, with regards to the accuracy of the pretrained models it is observed that *VGG16* outperforms all of them with a 96.4% F1 accuracy score with *ResNet50* closely following with 96.1%. This is in line with what has been reported in prior works using such networks achieving accuracies between $81 - 98\%$ for different applications and scenarios. However, both networks have very high demands for computational and storage requirements making them unsuitable for resource constraint systems and real-time use. The latest iteration V3[2] achieves the higher accuracy of the mobilenet family of networks it has the lowest FLOPs from the pretrained networks and achieves a score of 95.3%. It requires an order of magnitude more parameters and memory, however. The other mobilenet versions (V1 and V2) operate on smaller image resolutions (224×224), however still have higher FLOP requirements. *EfficientNet*[3] provides a high accuracy of 96.0% due to its

[2]The smaller varient of MobileNet V3 was used as it is more computationally efficient.

[3]The optimized version B0 is used

Table 4: Comparison of the different fusion mechanisms and base models

Model	**Parameters**	**Memory** (MB)	**F1 Score** (%)
Standard Convolutional	$719,737$	3.08	93.1
Depthwise-Separable	$95,849$	0.383	90.5
Spatially-Separable	$627,913$	2.511	92.5
Max-Fusion	$90,892$	0.363	95
Average-Fusion	$90,892$	0.363	93.9
Add-Fusion	$90,892$	0.363	95.7
Concatenate Fusion	$222,881$	0.891	94.3

Table 5: Comparison with existing approaches

Model	**Parameters**	**Memory (MB)**	**F1 Score (%)**	**FLOPS** ($\times 10^6$)
AsphaltNet	90,892	0.368	95.7	57
ResNet50	24,113,541	96.4	96.1	4,533
VGG16	14,849,349	59.39	96.4	17,620
Xception	21,387,309	85.549	95.3	419
MobileNet V1	3,492,549	13.9	95	550
MobileNet V2	2,587,205	10.3	95.2	279
MobileNet V3	3,046,037	12.1	95.3	60
SqueezeNet	698,917	2.7	91.5	833
ShuffleNet	4,282,425	17.1	91.1	524
EfficientNet (B0)	4,378,785	17.5	96.0	420
Fire_Net	5,235,860	5.2	90.5	1577

elaborate architecture. However, the parameter count and memory requirements are higher than AsphaltNet. Other networks fail to provide adequate accuracy and require a higher number of FLOPs with more parameters. It is clear from this analysis that it is worth investigating tailored made solutions for constrained applications in order to provide an improvement across all design aspects.

AsphaltNet achieves the same accuracy with fewer FLOPs and parameters than the most competitive architectures. It manages to achieve 95.7% F1 score accuracy which is very close to the pretrained network approaches with minimal memory requirements at $\sim 360KB$ which is $\sim 30\times$ smaller than *MobileNetV3*. Overall, *AsphaltNet* has comparable performance to more recent models such as *EfficientNet* and *MobileNetV3* in terms of accuracy with much less parameters. In addition, the significantly smaller model size of *AsphaltNet* compared to other models illustrates its efficacy for greatly reducing the memory requirements for leveraging image classification for real-time embedded emergency response applications, which also makes it suitable for on-chip storage on low-power platforms with limited memory as well as more specialized computing platforms such as FPGAs which can have limited on-chip storage [22].

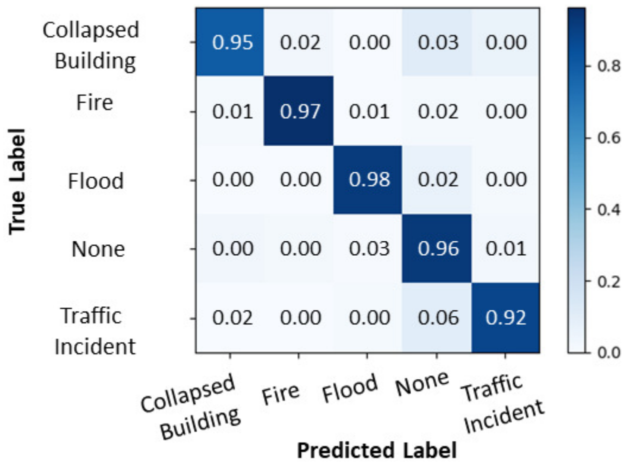
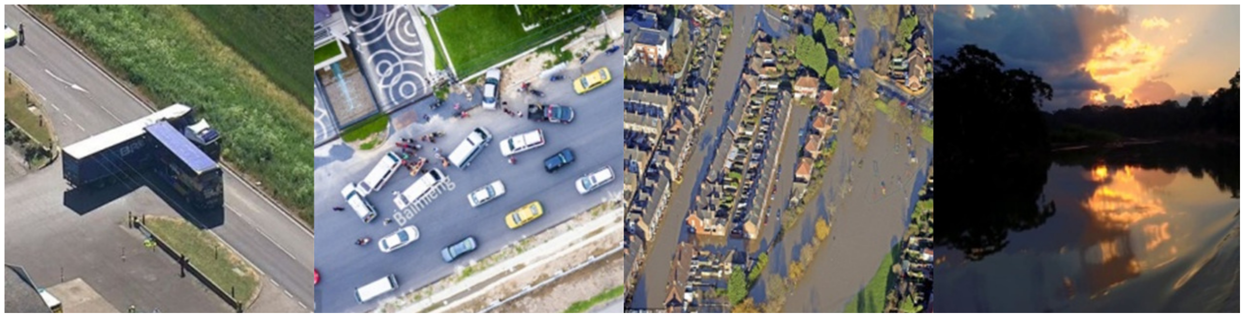


Figure 6: AsphaltNet Confusion Matrix

The confusion matrix in Fig. 6 of *AsphaltNet* allows us to draw some conclusions regarding the difficulty of the task and also highlight some potential research opportunities. Some areas of improvement can come from reducing the majority of errors attributed to classifying images as being in the *normal* class. This is to be expected since there are some images where the incident does not occupy a large image area and the viewpoint may not be ideal. To atone for such cases it is a common practice to process multiple crops from the input images. However, this has negative implications with regards to the processing time. Also for the *traffic incident* class it is observed that overall it produced the lowest accuracy. This is because some of the images the "incident" was not clear either because of the viewpoint or because in some cases the label was ambiguous. Some examples of interesting classification outcomes are shown in Fig. 7. It is worth noting that these behaviours were consistent across all networks, agreeing with recent studies that in difficult circumstances more contextual information is needed in order for CNNs to make better predictions.



(a) (b) (c) (d)

Figure 7: Interesting Classification Cases:(a & b) In these cases the incident let that be crash or damage is not clearly visible and as a results the models mis-classify the images. (c) Image of flood not classified correctly as the hue is similar to road. (d) The orange-yellow hue does not confuse the model as it correctly classifies the image

4.3 Qualitative Evaluation of Learning

In this section a closer look is taken into what features and image regions influence the prediction of the neural network and what it has learned to respond to in the various cases in order to come to a classification decision. To this end, the approach in [4] is used to find parts of the image that produce the highest activations. Through this approach in each layer, the activations of the feature maps are averaged and are scaled up to the size of the map of the preceding layer. The up-scaled averaged map from an upper level is then multiplied with the averaged map from the layer below. These steps are repeated until the input is reached. In addition, the Gradient-weighted Class Activation Mapping (Grad-CAM) approach [36] is also employed to visualizing the regions of input that are important for class-specific predictions and uses gradient information flowing into the final convolutional layer rather than activation output. These visualizations mask shows which regions of the input image contribute most to the output of the network. Figure 8 indicates some images which have been correctly classified and the regions corresponding to the highest activation's of the CNN (the

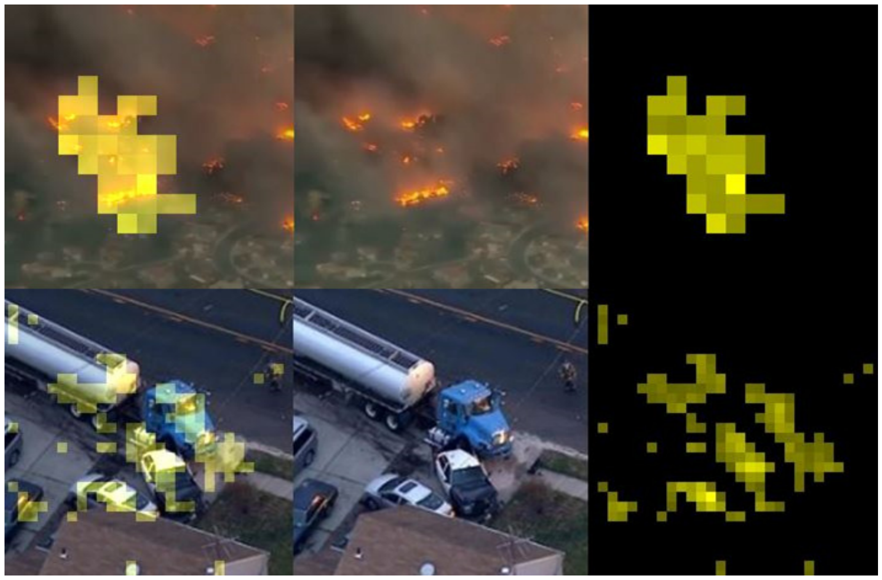


Figure 8: (Left) Feature Maps overlaid on input images (center) input images (right) High Activation feature Maps: Image regions that correspond to the highest activations in the network. These areas are mainly concentrated on the main characteristics of each event. In all cases the network focuses on important cues within the image to makes its prediction. For example in the first image it focuses on the red-orange glow of fire as well as the rubble of the collapsed building in the third image.

AsphaltNet model in particular). Fig. 10 indicates some images which have been correctly classified and the produced heat-map indicating important regions in the CNN's decision making. Through both approaches it is evident that the CNN uses important cues in order to come to a decision such the red-yellow glow fire in Fig. 8 and the building rubble in Fig. 10. The heat-maps consistently show that the pixels corresponding to the disaster event dominate the importance of the classification result. This indicates that the network indeed learns how to spot important incidents for emergency situations.

4.3.1 Disaster Recognition Network Architecture DirectNet (related Work)

To enhance the operation of a UAV in emergency response, it is necessary to have lightweight algorithms that provide a good trade-off of complexity and accuracy. To this end and to motivate more work towards this area, we proposed the design of a custom CNN designed from scratch to be efficient by tailoring the use of convolutional layers and kernel sizes. The custom CNN called DiRecNet consists of four main blocks, making it feasible for the model to learn hierarchical feature representations without reducing the feature map resolution too much. On the first two blocks, we use normal convolutional layers to extract richer low level features, while on the last two blocks we utilized separable convolutions to account for the fact that the channel size increases and has more efficient computations with a reduced number of operations and parameters. In more detail, the model first passes the scaled images onto two consecutive normal convolutional layers. The former with a kernel size of 7×7 pixels and 16 filters, while the latter with 5×5 pixels and 16 filters. This follows modern network trends that apply larger kernels [16]. The smaller channel number is used to offset the larger kernel size. Batch normalization is applied just after these convolutions, with a max-pooling operation of stride 2×2 after that. The data points are then passed to the next block of two convolutional layers with kernel size of 3×3. The first convolution involves 32 filters, while the second has 64 convolution filters. Again, batch normalization is applied before the next max pooling layer. The third block involves two separable convolutions with 128 and 256 filters respectively and 3×3 size, followed by a batch normalization layer. A max-pooling operation is also applied with a pool size of 2×2. The last block is designed with two identical separable convolutions of the 512 filter and the size 3×3. Finally, a global average pooling layer is applied to flatten the features. These are then passed to a fully connected layer of 1024 neurons, before a dropout of 0.7. Another fully connected layer is applied with 512 neurons with a dropout of 0.5. The last layer of the model is a fully connected layer of size 4 as the number of classes. An overview of the model is depicted in Fig.9.

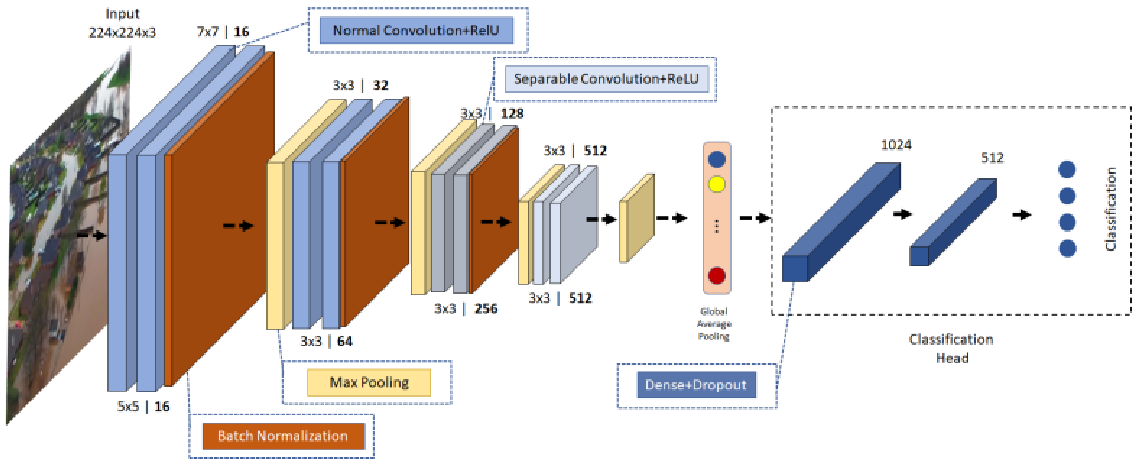


Figure 9: Proposed Convolutional Neural Network Architecture For The DirecNet Model

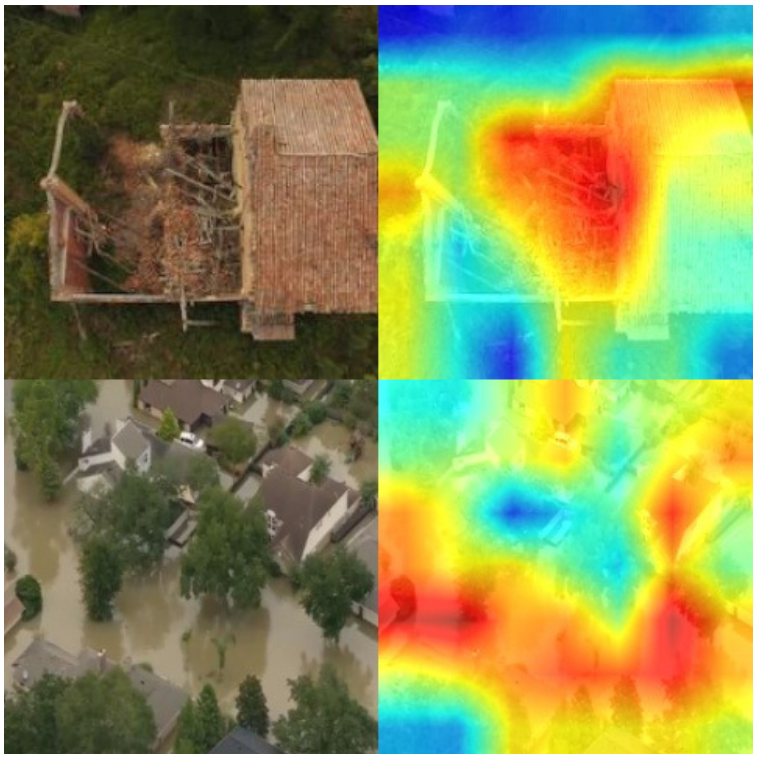


Figure 10: Images classified correctly and the corresponding class activation map. In all cases the visualization shows that the network focuses on important cues within the image to make a decision. (top) Focuses primarily on the rubble of the collapsed building. (bottom) Focuses on the flooded area.

4.4 Embedded Application Results

Unfortunately , we couldn't test our model in a real world scenario , however, we found an experiment of one of the existing works that is close to our use case, so we'll cover their experiment in the following section .

This section presents the evaluation of *DirecNet* in real use-cases. Edge devices with limited computational resources and restrictive energy overhead are targetted such as ARM-based mobile devices with 10-150 MFLOPs. The processing speed and subsequent frame-rate are measured for the proposed network, *DirecNet*, along with other state-of-the-art

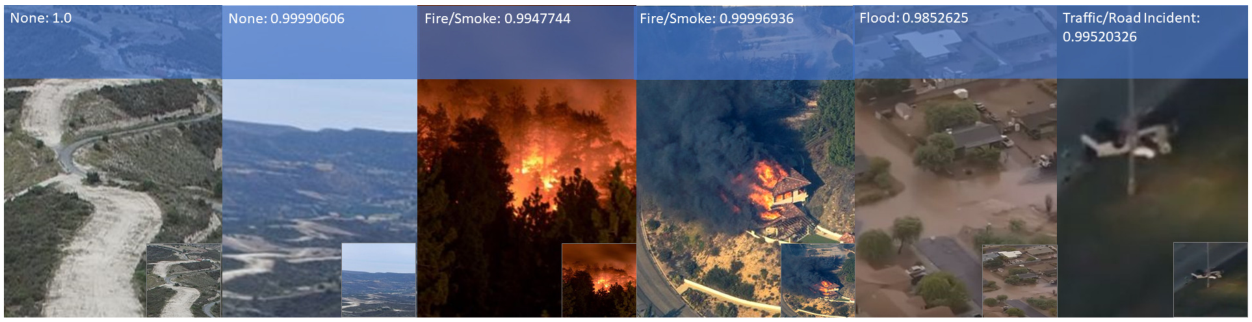
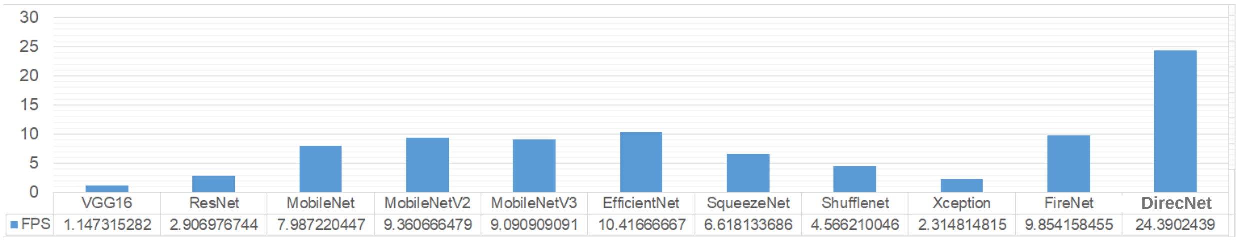


Figure 11: Experimental embedded platforms: Screenshots from Android mobile ground station. Neural network confidence and prediction outcome at the top of each image.



	VGG16	ResNet	MobileNet	MobileNetV2	MobileNetV3	EfficientNet	SqueezeNet	Shufflenet	Xception	FireNet	DirecNet
■ FPS	1.147315282	2.906976744	7.987220447	9.360666479	9.090909091	10.41666667	6.618133686	4.566210046	2.314814815	9.854158455	24.3902439

Figure 12: Frames-per-second when running on the ARM-based platform.

networks. Two different scenarios are targetted. The first concerns an on-board processing platform and the second a tablet which acts as the UAV mobile ground station that is connected to the UAV controller to process the input image. Both options are easily deployable and thus suitable for remote monitoring in emergency scenarios. Experimental setup using a DJI Matrice 100 UAV and experimental results for the two use-cases are shown in Fig. 13 and Fig. 11 respectively.

4.4.1 UAV On-board Processing

For the on-board processing a platform featuring with a quad-core ARM Cortex-A57 (Fig. 13) is used which is powerful and energy-efficient and comes in a small form factor suitable for UAVs. As shown in Fig. 12 the *DirecNet* model achieves ~ 25 FPS on this platform which is already far more practically applicable than other state of the art models that achieve at most ~ 9 FPS. It is important to note that the reduced FLOPs obtained by *MobileNetV3* does not directly translate to improved performance on the specific CPU-based experimental platform. This further emphasizes the need to optimize with the specific characteristics of the processing platform in mind. Overall, a speedup of $2 - 20\times$ is observed compared to other models. However, it is possible to provide further gains by applying quantization and bit reduction techniques, which *DirecNet* is already well suited for, to further improve performance for CPU platforms and potentially run at even higher frame-rates.

4.4.2 Processing on UAV Mobile Ground Station

Evaluation is also performed on a more conventional system where the UAV controller is connected to a tablet which acts as a mobile control station that is fed with the UAV camera image (Fig. 11). In this setup the video feed can be directly processed on the tablet without impacting the UAV platform in terms of weight and power. The *DirecNet* network manages to offer higher frame-rates ~ 19 FPS compared to state-of-the-art networks which remain below 10 FPS. Notice, that in real-world cases, interference can be observed between the UAV and the controller thus negatively impacting the reliability of the visual task. Hence the former case can be a more robust option.

4.5 Discussion and Further Improvements

Overall, the results are encouraging for the DirecNet Model , Knowing that our developped model is better in more accurate and less complex in terms of computational power , so in that the *AsphaltNet* network is able to provide accuracy comparable to state-of-the-art models at a fraction of memory and computation . There are still improvements that are possible particulary if the computing platform allows for them. The large margins afforded by *AsphaltNet* allows us to process even higher resolution images. Particularly, by combining the efficient model with tiling approaches

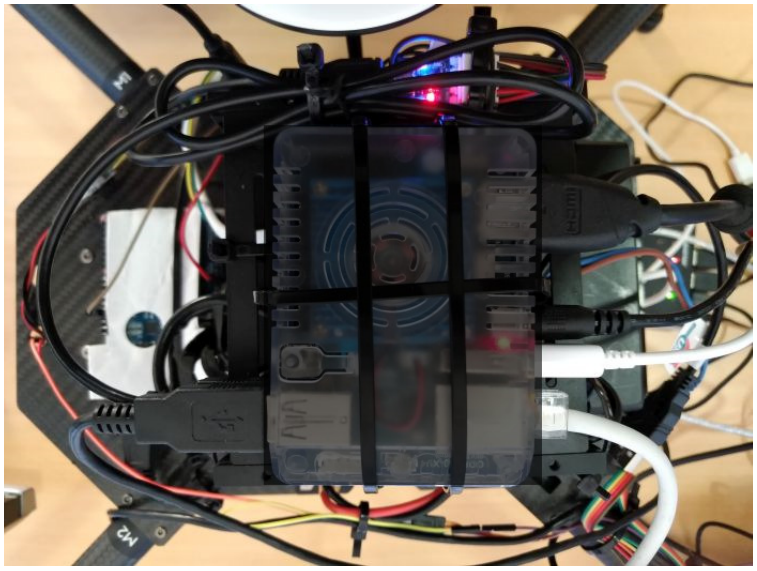


Figure 13: Experimental embedded platforms on-board a DJI Matrice 100 UAV

that break larger images into smaller regions that can be processed individually by a model. Especially, by combining them into a single batch and utilizing parallelization capabilities it is possible to offer comparable frame-rates for higher resolution images [33].

With regards to video streams the overall classification for EmergencyNet Model performance can be further improved in our Model AsphaltNet when deployed through the assumption that subsequent frames in a video are correlated with respect to their semantic contents. By measuring the distance between the output vectors prior to the classification layer it is possible to infer the similarity between different frames and accordingly weight the predictions of the past frames in order to remove any classification flickering and smooth out the predictions. In our test videos this simple yet effective technique managed to reduce prediction flickering.

The feature fusion architecture was found to provide good representational power with reduced parameter count. Hence, it would be beneficial to exploit recent advancements in automated model search which are being increasingly deployed in recent works[12, 43] as a logical next step to further improve the feature fusion architecture.

Through the analysis some difficult cases have been identified that require further research in order to develop better solutions. For example, the *traffic incident* case can cover a broad range of scenarios and breaking down into subclasses can be beneficial. Another important challenge was the difficulty of collecting and gathering the data. For particular cases where it is not easy to gather loads of data through the internet it could be beneficial to explore the recent advances in generative models to learn the joint probability distribution for each class in order to generate novel realistic synthetic data.

Finally, the potential to combine *AsphaltNet* with algorithms that detect people and vehicles as well as additional modalities (e.g., infrared or thermal cameras) can lead to even more enhanced situational awareness that can provide valuable tool for emergency response and disaster management applications especially when integrated with geospatial applications for geo-tagging of the recognized events.

5 Concluding Remarks

This work has been a foundation study of on-board UAV processing for emergency response applications. An analysis on the design and implementation of an efficient deep learning system has been carried out to automatically recognize and classify disaster events in real-time from on-board a UAV. The proposed solution provides an adequate trade-off between accuracy, inference speed, and complexity and that it can be used as a building block towards use-cases with similar constraints. The experimental study validates the efficiency of the proposed method since *AsphaltNet* is up to $20\times$ faster, requires an order of magnitude less memory and provides similar or better accuracy to existing models. In addition, a dedicated aerial image dataset for emergency response applications is introduced which researchers can use to further advance the existing models. The dataset will be further expanded and enhanced with additional images and classes in order to further raise the awareness of the community towards such applications and improve on existing models and techniques.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[2] Abdulla Al-Kaff, David Martín, Fernando García, Arturo de la Escalera, and José María Armingol. Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Systems with Applications*, 92:447 – 463, 2018.

[3] Mesay Belete Bejiga, Abdallah Zeggada, Abdelhamid Nouffidj, and Farid Melgani. A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery. *Remote Sensing*, 9(2), 2017.

[4] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence D. Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *CoRR*, abs/1704.07911, 2017.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Computer Vision – ECCV 2018*, pages 833–851, Cham, 2018. Springer International Publishing.

[6] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han. When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative cnns. *IEEE Transactions on Geoscience and Remote Sensing*, 56(5):2811–2821, May 2018.

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[8] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[10] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 2018.

[11] M. Hossin and M.N Sulaiman. A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 5(2):1–11, November 2019.

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[14] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[15] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *CoRR*, abs/1801.02929, 2018.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015.

[17] Andreas Kamilaris and Francesc X. Prenafeta-BoldÃº. Disaster monitoring using unmanned aerial vehicles and deep learning. In *Disaster Management for Resilience and Public Safety Workshop, in Proc. of EnviroInfo2017*, Luxembourg, September 2017.

[18] S. Kim, W. Lee, Y. s. Park, H. W. Lee, and Y. T. Lee. Forest fire monitoring system based on aerial image. In *2016 3rd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–6, Dec 2016.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[20] Byoung Chul Ko, Kwang-Ho Cheong, and Jae-Yeal Nam. Fire detection based on vision sensor and support vector machines. *Fire Safety Journal*, 44(3):322 – 329, 2009.

[21] Simon Kornblith, Jon Shlens, and Quoc V. Le. Do better imagenet models transfer better? 2019.

[22] C. Kyrkou, C. S. Bouganis, T. Theocharides, and M. M. Polycarpou. Embedded hardware-efficient real-time classification with cascade support vector machines. *IEEE Transactions on Neural Networks and Learning Systems*, 27(1):99–112, Jan 2016.

[23] C. Kyrkou, S. Timotheou, P. Kolios, T. Theocharides, and C. G. Panayiotou. Optimized vision-directed deployment of uavs for rapid traffic monitoring. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, Jan 2018.

[24] L. Li, K. Ota, M. Dong, and W. Borjigin. Eyes in the dark: Distributed scene understanding for disaster management. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3458–3471, Dec 2017.

[25] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[26] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Computer Vision – ECCV 2018*, pages 122–138, Cham, 2018. Springer International Publishing.

[27] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):645–657, Feb 2017.

[28] Volodymyr Mnih and Geoffrey Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 203–210, USA, 2012. Omnipress.

[29] D. Murugan, A. Garg, and D. Singh. Development of an adaptive approach for precision agriculture monitoring with drone and satellite data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(12):5322–5328, Dec 2017.

[30] Tien Dat Nguyen, Shafiq R. Joty, Muhammad Imran, Hassan Sajjad, and Prasenjit Mitra. Applications of online deep learning for crisis response using social media information. *CoRR*, abs/1610.01030, 2016.

[31] P. Petrides, C. Kyrkou, P. Kolios, T. Theocharides, and C. Panayiotou. Towards a holistic performance evaluation framework for drone-based object detection. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1785–1793, June 2017.

[32] Petros Petrides, Panayiotis Kolios, Christos Kyrkou, Theocharis Theocharides, and Christos Panayiotou. Disaster prevention and emergency response using unmanned aerial systems. In *Smart Cities in the Mediterranean: Coping with Sustainability Objectives in Small and Medium-sized Cities and Island Communities*, pages 379–403, Cham, 2017. Springer International Publishing.

[33] George Plastiras, Christos Kyrkou, and Theocharis Theocharides. Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, ICDSC '18, pages 3:1–3:6, New York, NY, USA, 2018. ACM.

[34] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA, 2014. IEEE Computer Society.

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[36] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, Oct 2017.

[37] Muhammad Shafique, Theo Theocharides, Christos Bouganis, Muhammad Abdullah Hanif, Faiq Khalid, Rehan Hafiz, and Semeen Rehman. An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the iot era. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 03 2018.

[38] Jivitesh Sharma, Ole-Christoffer Granmo, Morten Goodwin, and Jahn Thomas Fidje. Deep convolutional neural networks for fire detection in images. In Giacomo Boracchi, Lazaros Iliadis, Chrisina Jayne, and Aristidis Likas, editors, *Engineering Applications of Neural Networks*, pages 183–193, Cham, 2017. Springer International Publishing.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[40] V. Sze, Y. Chen, J. Emer, A. Suleiman, and Z. Zhang. Hardware for machine learning: Challenges and opportunities. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–8, 2018.

[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[42] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[43] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[44] B. Uğur Töreyin, Yiğithan Dedeoğlu, Uğur Güdükbay, and A. Enis Çetin. Computer vision based method for real-time fire and flame detection. *Pattern Recogn. Lett.*, 27(1):49–58, January 2006.

[45] Y. Wang, Z. Quan, J. Li, Y. Han, H. Li, and X. Li. A retrospective evaluation of energy-efficient object detection solutions on embedded devices. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 709–714, March 2018.

[46] Y. Wang, L. Zhang, X. Tong, F. Nie, H. Huang, and J. Mei. Lrage: Learning latent relationships with adaptive graph embedding for aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2):621–634, Feb 2018.

[47] C. Yuan, Z. Liu, and Y. Zhang. Uav-based forest fire detection and tracking using image processing techniques. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 639–643, June 2015.

[48] W. Zhao, S. Du, and W. J. Emery. Object-based convolutional neural network for high-resolution imagery classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(7):3386–3396, July 2017.

[49] Yi Zhao, Jiale Ma, Xiaohui Li, and Jie Zhang. Saliency detection and deep learning-based wildfire identification in uav imagery. *Sensors*, 18(3), 2018.

[50] Rui Zhu, Shifeng Zhang, Xiaobo Wang, Longyin Wen, Hailin Shi, Liefeng Bo, and Tao Mei. Scratchdet: Training single-shot object detectors from scratch. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2268–2277. Computer Vision Foundation / IEEE, 2019.