Deep-Learning-Based Aerial Image Classification for Emergency Response Applications using Unmanned Aerial Vehicles

Christos Kyrkou[1] and Theocharis Theocharides[2]
KIOS Research and Innovation Center of Excellence[1,2],
Department of Electrical and Computer Engineering[2],
University of Cyprus,1 Panepistimiou Avenue,Nicosia, Cyprus
{kyrkou.christos, ttheocharides}@ucy.ac.cy

Abstract

Unmanned Aerial Vehicles (UAVs), equipped with camera sensors can facilitate enhanced situational awareness for many emergency response and disaster management applications since they are capable of operating in remote and difficult to access areas. In addition, by utilizing an embedded platform and deep learning UAVs can autonomously monitor a disaster stricken area, analyze the image in real-time 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. To this end, this paper focuses on the automated aerial scene classification of disaster events from on-board a UAV. 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 developed, capable of running efficiently on an embedded platform achieving $\sim 3\times$ *higher performance compared to existing models with minimal memory requirements with less than* 2% *accuracy drop compared to the state-of-the-art. These preliminary results provide a solid basis for further experimentation towards real-time aerial image classification for emergency response applications using UAVs.*

1. Introduction

Over the past few years Unmanned Aerial Vehicles (UAVs) have gained considerable interest as a remote sensing platform for various practical applications, such as traffic monitoring [12] and search and rescue [17]. Recent technological advances such as the integration of camera sensors provide the opportunity for new UAV applications such as monitoring and identifying hazards and disasters in emergency situations (e.g., fire spots in forested areas, flooding threat, road collisions, landslide prone areas) [2] by means of analyzing the captured aerial images in real-time. In addition, due to their small size UAVs offer fast deployment and can thus provide a unique tool to rapidly assess a situation and improve risk assessment mitigation [16]. 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 which often have limited connectivity and visibility to the operators. 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 [17]. The challenge in such cases is to enable the efficient visual processing on-board the UAV given that the available hardware may have limitations in terms of computing power and memory.

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 [14, 7, 4]. 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 the decision-making processes [15]. Even though CNNs have been increasingly successful at various classification tasks through transfer learning [18], their inference speed on embedded platforms such as those found on-board UAVs is hindered by the high computational cost that they incur and the model size of such networks is prohibitive from a memory perspective for such embedded devices [25, 20]. However, for many applications local embedded processing near the sensor is preferred over the cloud due to privacy and latency concerns, or operation in remote areas where there is limited or even no connectivity.

This work addresses the problem of on-board aerial scene classification for emergency response applications which is to automatically assign a semantic label to characterize the aerial image that the UAV captures [26]. These labels correspond to a danger or hazard that has occurred. The specific use-case under consideration is that a UAV will follow a predetermined path as shown in[17]) 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. 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.

The main contributions of this work are summarized as follows:

- Construction of a dedicated database for the application of aerial image classification for emergency response and development of a suitable CNN training strategy.
- Development of a CNN (referred to as *ERNet*) with low-computational and suitable for low-cost low-power devices.

Through the analysis of the different models and techniques as well as evaluation on a real experimental UAV platform we demonstrate the effectiveness of this approach to simultaneously provide near state-of-the-art accuracy ($\sim 90\%$) while being $3\times$ faster on an embedded platform.

2. Background and Related Work

2.1. Convolutional Neural Network Architectures

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). Some of the most important architectures are highlighted next, whose components and ideas that will be used to develop an efficient CNN for embedded aerial scene classification with UAVs.

VGG16 [22]: 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 [6]: 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)

MobileNet [9]: Utilizing the idea of separable convolutions MobileNets manage to offer reduced computational cost with slight degradation in classification accuracy. It applies a single filter at each input channel and then linearly combines them. Thus is designed can be easily parametrized and optimized for mobile applications.

2.2. 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.

In [11] 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 missions 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 [24] 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 [21] also targets fire detection with deep learning. Specifically, two pretrained convolutional neural networks are used and compared, namely VGG16 [22] and Resnet50 [6] 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 [27] proposes an approach comprised 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. In [10] a deep convolutional neural network is trained to classify aerial photos in one of 5 classes corresponding to natural disasters. The VGG [22] 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.

From the literature analysis it is clear that existing approaches use existing pre-trained 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 [23]. Also, the computing limitations of embedded platforms constitute the use of existing algorithms targeting desktop-class systems infeasible.

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.

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 no widely used and publicly available dataset for emergency response applications. 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, 320 images of *Fire/Smoke*, 370 images for *Flood*, 320 images for *Collapsed Building/Rubble*, and 335 images for *Traffic Accidents*, as well as 1200 images for the *Normal* case. Visually similar images such as for example active flames and smoke are grouped together.

The aerial images of these disaster classes were collected from multiple sources such as the world-wide-web (e.g. google images, bing images, youtube, news agencies web sites, etc.), other databases of general aerial images, and images collected using our own UAV platform. 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, a certain 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 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. 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 these 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. 1. Overall, with respect to the related works that consider multiclass problems (e.g., [10]) almost 5× more data were collected. In addition, using augmentations the initial dataset was considerably expanded even further.

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 methodol-

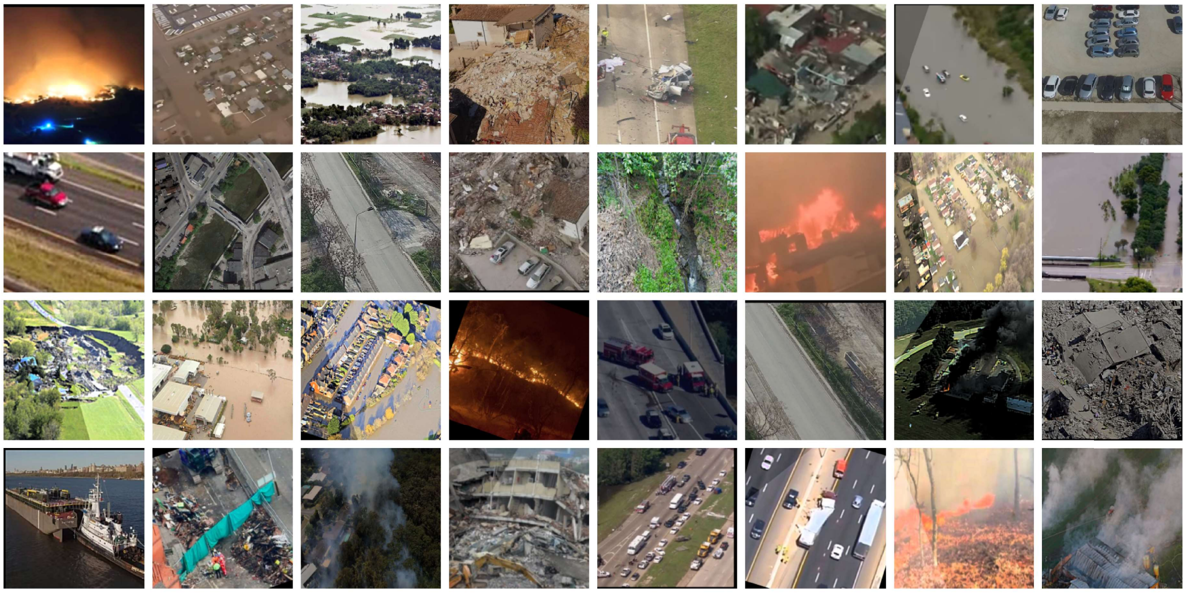

Figure 1: **A**erial **I**mage **D**ataset for **E**mergency **R**esponse (AIDER) Applications: Example images from the of Augmented Database

ogy used in prior works. Furthermore, new network structures are designed and 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 established networks are used, namely *VGG16* [22] and *ResNet50* [6] which have also been used in prior works outlined in Section 2.2 such as [10, 21, 11] as well as networks more suited to embedded domains such as *MobileNet*[9]. 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 [13]. 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

The larger and deeper networks attained through transfer learning may not be suited for resource-constrained systems such as UAV platforms, which impose limitations of the size of the platform, the weight, and its power envelope. For this reason there is a need to design specialized networks that are inherently computationally efficient. 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.

- **Reduced Cost of First Layer**: The first layer typically incurs the higher computational cost since it is applied on the whole image. Hence, a relatively small number of filters is selected (16) with higher spatial resolution of 5×5 compared to the latter layers. Overall, using an increased filter size resulted in an increase in accuracy. The improvement is attributed to the fact that since a relatively small number of filters is used compared to other works (e.g., 64) more image information by needs to be captured by increasing the filter size. Experiments were also performed starting with a filter depth of 8 but this resulted in reduced accuracy.

- **Early downsampling**: Max pooling layers are used after each convolution layers (either separable or normal) to half the input size and effectively reduce the computational cost, It was empirically found that

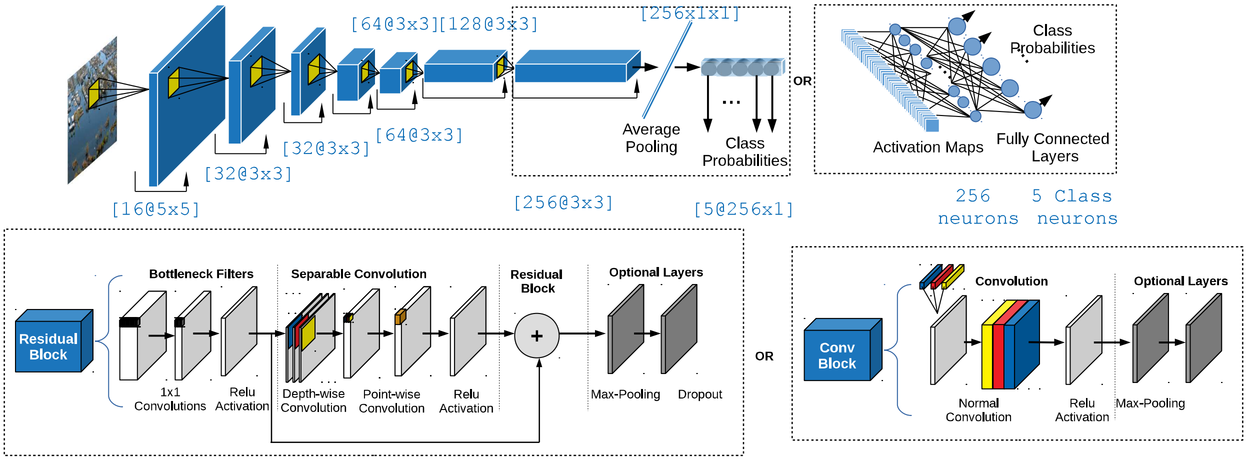

Figure 2: Different configurations for setting up a CNN model for aerial disaster classification.

downsizing the feature maps in the latter stages resulted in decreased accuracy hence, the downsampling is performed in all but the two final convolutional 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 doubled thereafter but do not increase it 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 replacing the fully connected layers with a global average pooling operation followed by convolutional layers (before feeding into the softmax activation) for reduced number of parameters.

- **Efficient 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* [3] is used after each convolutional layer and also the dropout [12] strategy is employed with a ratio of 0.5 of drop connections during training. All the convolutional layers are also regularized using an L2 regularizer.

- **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 found that a network size of 7 major processing blocks (combined layers) was sufficient to achieve comparable accuracy to the state-of-the-art, while increasing it did not result in significant accuracy improvements but incurred higher computational cost.

- **Residual Connections:** Residual learning is incorporated in the architecture. Specifically, skip connections are introduced from the input of a computation block to its output where it is merged back with an element-wise addition. As it will be shown in the experiments residual learning enables to increase the accuracy further with a slight decrease of the computation performance.

The aforementioned configurations are combined to build the *ERNet* architecture shown in Fig. 2. In addition, using the aforementioned basic principles different networks are designed featuring various combinations of the main configurations in order to compare and contrast the trade-offs. Specifically, the main differences between the networks stem from the use of specific layer types and techniques such as separable convolutions, and residual connections. First, a canonical CNN (referred to as *baseNet*) is designed composed of normal convolutional layers with fully connected layers at the end. In the second network configuration the convolutional part is replaced by a separable convolution (depth- and point- wise), while maintaining the fully-connected layer at the end (referred to as *SCNet*). For the third network configuration the fully-connected portion is replaced by convolutional layers in order to reduce the parameter size and memory demands of the network even further (referred to as *SCFCNet*). For the final configuration, residual connections are added to each layer to improve the learning performance and this constitutes the final architecture which is referred to as *ERNet*. The progressive changes in the network configurations allows us to study how each choice impacts both performance and accuracy the results of which will be shown in Section 4.

3.3. Training

All the networks are developed and tested through the same framework so as to have the same conditions and a fair comparison during the inference phase. The Keras deep learning framework [5] is used which has available all the pretrained models used for transfer learning, with Tensorflow [1] running as the backend [1]. The same image size is used for all networks where possible (except for the *mobileNet* which specifically requires 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.

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 0.6, 0.2, 0.2 ratio. As mentioned prior, the *Normal* class is the majority class and thus is overrepresented 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 we select the same number of images form each class to form a batch and this way all cases are equally represented.

All the networks where trained using a GeForce Titan Xp, on a PC with an Intel $i7 - 7700K$ processor, and 32GB of RAM. The Adam optimization method was used for training with learning rate-decay starting from a learning rate of 0.001, and multiplying it by a factor of 0.95 every 5 epochs to achieve a smoother decrease. Each network is trained for 200 epochs each comprising of 100 batch iterations, with a batch size of 64 resulting in 6400 generated training images per epoch.

4. Experimental Evaluation and Results

In this section the analysis of the trained networks is presented with results from the experimental evaluation of the approach on an actual embedded platform attached on the UAV as well as the UAV ground station. Evaluation is performed on a desktop i7 CPU that can be easily ported to a computational platform used in UAVs such as an Android platform that acts as the mobile control station or embedded devices such as Odroid XU4.

4.1. Performance Metrics

The ultimate objective of this work is to be able to run the models on board a UAV and process each image online. Hence, an important performance metric is the achievable 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. In addition, since the prior distribution over classes is signicantly nonuniform a simple accuracy measure (percentage of correctly classied examples) which is used in related works, may not be appropriate for the specific problem considered in this work since usually the normal case would have a larger number of samples in the test and training set than the other classes. To avoid this bias in our results an average accuracy ($\overline{A}$) metric [8] is employed instead that averages across the accuracies of each class rather than that of the test set as a whole.

4.2. Overall Performance, Analysis and Comparison

The results for all networks are summarized in Table 1. First, with regards to the accuracy of the pretrained models it is observed that *VGG16* outperforms all of them with a 91.9% $\overline{A}$. This is in line with what has been reported in prior works using this network achieving an accuracy between $81 - 98\%$ for different applications and scenarios however. With regards to the frame-rate it achieves 2 FPS which is not suitable for real-time use. The fastest of the pretrained networks is *MobileNet* achieving a frame-rate of 20, however it operates on smaller image resolution (224×224) and achieves a average accuracy of 88.5%. Also, all the networks require over $10MB$ which may be prohibitive for on-chip storage in embedded platforms. It is clear from this analysis that it is necessary to investigate tailored made solutions for constrained applications in order to provide an improvement on all design parameters.

Appropriately then the evaluation of the custom networks is presented next. Even starting from a *baseNet* network and following the design choices outlined in Section 3.2.2 it is noticeable that it performs close enough and in some cases outperforms some pretrained networks with regards to average accuracy. Also as a consequence of the careful design choices it manages to offer a 16 speedup over the most accurate network which is the *VGG16*. Applying additional optimizations such as employing separable convolution filters (*SCNet*) can further improve performance

[1] We plan on releasing all the models and the training data as open source

Table 1: Summary of Results for all the trained models on PC Setup

CNN Model	Type[1]	Average Accuracy (%)	Processing Time (ms) [2]	Frames-Rate[2]	Speedup[3]	Memory (MB)
ERNet	C	90.1	18.7	53	18.5	0.3
SCFCNet	C	87.7	13.1	76	26.4	0.2
SCNet	C	85.4	14.1	70	24.5	6.5
baseNet	C	88	21.2	47	16.3	7
VGG16	T	91.9	346	2	1	59.3
ResNet50	T	90.2	257	3	1.3	96.4
MobileNet	T	88.5	48.2	20	7.1	13.9

[1] C: Custom Network Trained from scratch — T: Pretrained network used for transfer learning to the new task

[2] Processing speed and Frame-Rate as measured on an Intel i7 CPU.

[3] Speedup with respect to the network with the higher accuracy which is the *VGG16*.

in terms of FPS however, it negatively impacts the accuracy as a $\sim 3\%$ drop is observed. Surprisingly, using a fully-convolutional approach (*SCFCNet*) mitigates this factor. This can be attributed to the fact that the spatial representations are preserved within the feature maps and do not collapse such as in the case of using fully dense layers. Still, however, the average accuracy is lower than that of the *baseNet* model. The very high frame-rates achievable by the existing models affords us a much larger margin to explore the design space in an attempt to further improve the accuracy. By introducing the final component of our design, the residual connections to form the final *ERNet* architecture it manages to achieve 90% average accuracy which is very close to the pretrained network approaches that have been used in prior works. Furthermore, it achieves over 50 FPS on a CPU-platform which makes the network suitable for real-time UAV applications. Also importantly, the final memory requirements for the network are $\sim 300KB$ 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.

4.3. 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. Such an analysis is particularly important to enable transparency as to why the models predict what they do and establish appropriate trust and confidence in users. In addition, such methodologies are also important during the training phase as they enable the identification of failure modes. To this end, we follow the Gradient-weighted Class Activation Mapping (Grad-CAM) approach in [19] to generate heat maps of the image regions that mostly influence a classification decision. Fig. 3 indicates some images which have been correctly classified and the produced heatmap indicating important regions in the CNN's (the *ERNet* model in particular) decision making. Notice that the CNN uses class-specific features to make a decision. For example, the demolished side of the building gin the first image and the red-orange glow of fire in the third image. The second and fourth images are more complex to analyze. In the former it seems that the network infers the flood first by the water and then by the presence of buildings that indicate a flooded area instead of a river for example. In the latter image again a combination of cues makes the network come to a decision such as the crashed train and the road segments.

4.4. Embedded Platform Results

UAV platforms can differ in their implementation depending on the use-case and deployment strategy; from very lightweight with only on-board components to requiring dedicated computing infrastructure at the ground station. The latter case has already been covered through the previous analysis. Thus in this section the focus is on the on-board processing platform. Experimental setup using a DJI Matric 100 UAV [2] and experimental results are shown in Fig. 4. For the on-board processing the ODROID-XU4 (Fig. 4) computing device is chosen which is powerful and energy-efficient and comes in a small form factor suitable for UAVs. [3]. The *ERNet* model achieves $\sim$ 9 FPS on this platform which is already far more practically applicable than other state of the art models that achieve at most ~ 3 FPS with lower accuracy. In addition, by applying quantization and bit reduction techniques it would be possible to further improve performance for CPU platforms and potentially run at even higher frame-rates.

[2] https://www.dji.com/matrice100

[3] It features a Samsung Exynos5 Octa ARM Cortex, 2Gbyte LPDDR3 RAM and uses between 10W and 20W

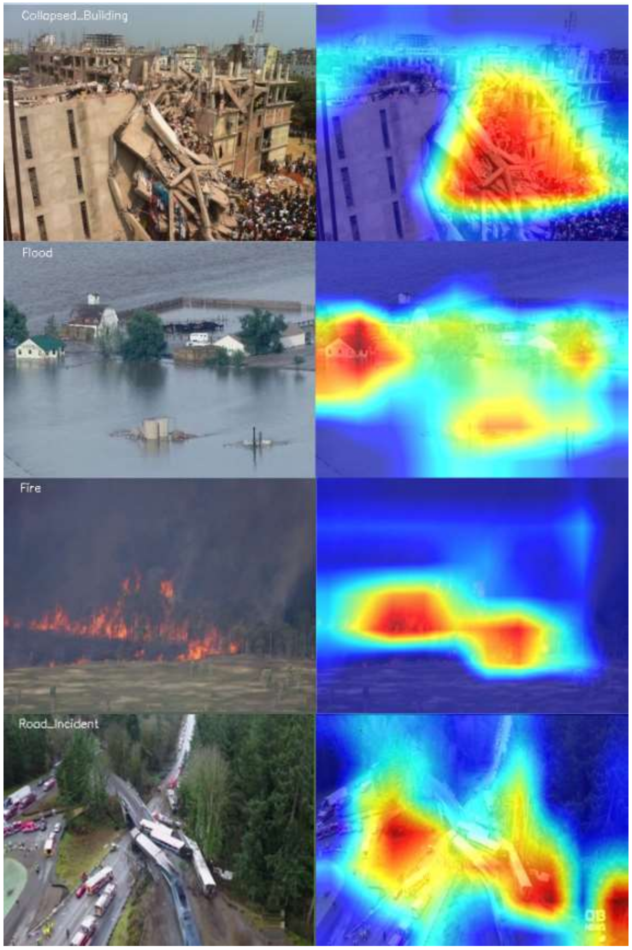

Figure 3: Images classified correctly and the corresponding class activation map. In all cases the visualization shows that the network focuses on important cues within the image to make a decision. From top to bottom: (a) an image of a collapsed building. (b) A flooded area. (c) Forest Fire. (d) Transportation Incident.

5. Conclusions and Future Work

This paper presented the first steps towards the automated classification of disaster events in real-time from onboard a UAV. It introduced a dedicated aerial image dataset for emergency response applications 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. Furthermore, a small and efficient convolutional neural network *ERNet* is developed that is up to 3× faster on an embedded platform, requires two orders of magnitude less memory and provides similar accuracy to existing models. Going forward, we are eager to investigate even further architectural design choices such as atrous-convolution layers as well as multi-frame processing to capture temporal information and further imprtove the results. It is also beneficial to study the impact of using different color spaces which may help improve the accuracy even further. Finally, the potential to combine *ERNet* with algorithms that detect people and vehicles as well as additional modalities (e.g., infrared camera) can lead to even more enhanced situational awareness that can provide valuable tool for emergency response and disaster management applications.

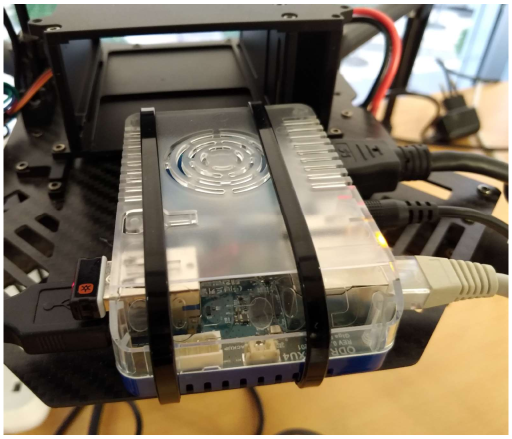

Figure 4: Experimental embedded platform: Odroid-XU4 embedded platform on-board a DJI Matrice 100 UAV

Acknowledgements

This work is supported by the European Union Civil Protection under grant agreement No 783299 (SWIFTERS) and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE) and from the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

Christos Kyrkou would like to acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

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 Martn, Fernando Garca, Arturo de la Escalera, and Jos Mara 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] 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.

[5] Franois Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] 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.

[8] Mohammad Hossin and Sulaiman M.N. A review on evaluation metrics for data classification evaluations. 5:01–11, 03 2015.

[9] 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.

[10] Andreas Kamilaris and Francesc X. Prenafeta-Bold. 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.

[11] 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.

[12] 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.

[13] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[23] Vivienne Sze, Yu-Hsin Chen, Joel S. Emer, Amr Suleiman, and Zhengdong Zhang. Hardware for machine learning: Challenges and opportunities. *CoRR*, abs/1612.07625, 2016.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[25] 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.

[26] 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.

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