CNN based Real-time Forest Fire Detection System for Low-power Embedded Devices

Jianlin Ye*, Stelios Ioannou*†, Panagiota Nikolaou*, Marios Raspopoulos*†
* University of Central Lancashire, Pyla, Larnaca, Cyprus
† INterdisciplinary Science Promotion & Innovative Research Exploration (INSPIRE)
Email: jye9@uclan.ac.uk,sioannou2@uclan.ac.uk,pnikolaou1@uclan.ac.uk,mraspopoulos@uclan.ac.uk

***Abstract*—This paper proposes a system architecture that uses deep learning image processing techniques to automatically identify forest fires in real-time using neural network models for small UAV applications. Considering the strict power and payload constraints of small UAVs, the proposed model runs on a compact, lightweight Raspberry Pi4B (RPi4B) and its performance is comparable to the state-of-the-art metrics (accuracy and real-time response) while achieving significant reduction in CPU usage and power consumption. The proposed YOLOv5 optimization approach used in this paper includes: 1) Replacing the backbone network to ShuffleNetV2, 2) Pruning the Head and Neck network following the backbone baseline, 3) Sparse training to implement the model-pruning method, 4) Fine-tuning of the pruned network to recover the detection accuracy and 5) Hardware acceleration by overclocking the RPi4B to improve the inference speed of the algorithm. Experimental results of the proposed forest fire detection system show that the proposed algorithm compared to the state-of-the-art that run on RPi single board computer, achieves 50% higher inference speed (9 FPS), reduction in CPU usage and temperature by 35% and 25% respectively and 10% reduced power consumption while the accuracy (92.5%) is only compromised by 2%. Finally, it is worth noting that the accuracy of the proposed algorithm is not affected by deviations in the bird-eye view angle.**

***Index Terms*—YOLOv5; Fire detection; UAV; Raspberry Pi**

I. Introduction

Climate change, has been responsible for the rapid increase of the size and frequency of forest/wild fires, which pose severe socioeconomic (the destruction of homes and loss of life) and environmental impact (carbon emissions, air and water quality) [1]–[5]. According to World Health Organization (WHO), 50% of recorded wildfires are from unknown origins and while the period between 1998-2017 affecting 6.2 million people and and being responsible for 35,000 fatalities [6]–[8]. Other worldwide statistics report that wildfires were responsible for Australia's 2009 'Black Saturday' destruction of 1,800 homes and burnt area of 4,500 km^2 whereas in 2017, 10,200 structures and 5,559 km^2 in California, USA, and 4,180 km^2 in Portugal [9]. The cost of wildfires damage is typically between 10 to 50 times the suppression estimated worldwide to be more than $100 billion annually [10].

Early warning methods and systems include the use of watchtowers, sensor networks, satellite remote sensing and patrolling using manned and unmanned aerial vehicles (UAVs). A comprehensive review of the aforementioned methods of early fire and smoke detection systems presented in [11] indicates that satellite systems provide a vast coverage area, but their response time depends on the location of the satellite and fire, while terrestrial sensor networks offer high accuracy and fast response times but their wide coverage is typically associated with significant increase in cost and system complexity. It seems that the use of small UAVs offers the highest future potential mostly because of their fast response time and extendable coverage area.

Technological advancements over the last 20 years, have enabled the production of an abundance of sensors and unmanned systems, affordable for commercial UAV applications. The list of such applications includes but is not limited to aerial photography [12], search and rescue [13], traffic monitoring [14], precision agriculture [15] and more. According to [16] the UAV market has surpassed $12 billion in 2021. Computer vision has gained popularity in UAV applications, however, the cumbersome processing of real-time target detection algorithms, requires additional single board computers (SBC) to avoid interference with navigation systems. Due to the strict run-time and payload availability on small and medium UAVs, the selection of onboard hardware is best addressed as a constraint optimization engineering process [17]. A comparison of the state of the art of commercially available SBC is examined by [18]. The study examined the UAV electrical power consumption due to the integrated SBC processing and its additional onboard weight. As concluded the Raspberry Pi (RPi) and Odroid platforms offer the lowest power consumption followed by Jetson.

A. Motivation of this work

The aim of this work is the development of a fire detection system for small UAV applications. Considering the current research progress in deploying target detection algorithms on embedded devices, and hardware and practical limitations, the research objectives include: The development of a computer vision target detection algorithm with reduced false alarm rate that is able to run on a compact, lightweight, and cost-effective SBC, such as RPi-4B while its performance remains comparable to the state-of-the-art metrics (accuracy and real-time response) under real-life wildfire scenarios.

This work was co-funded by the European Union under the programme of social cohesion "THALIA 2021-2027", through Research and Innovation Foundation (Project: CONCEPT/0722/0100)

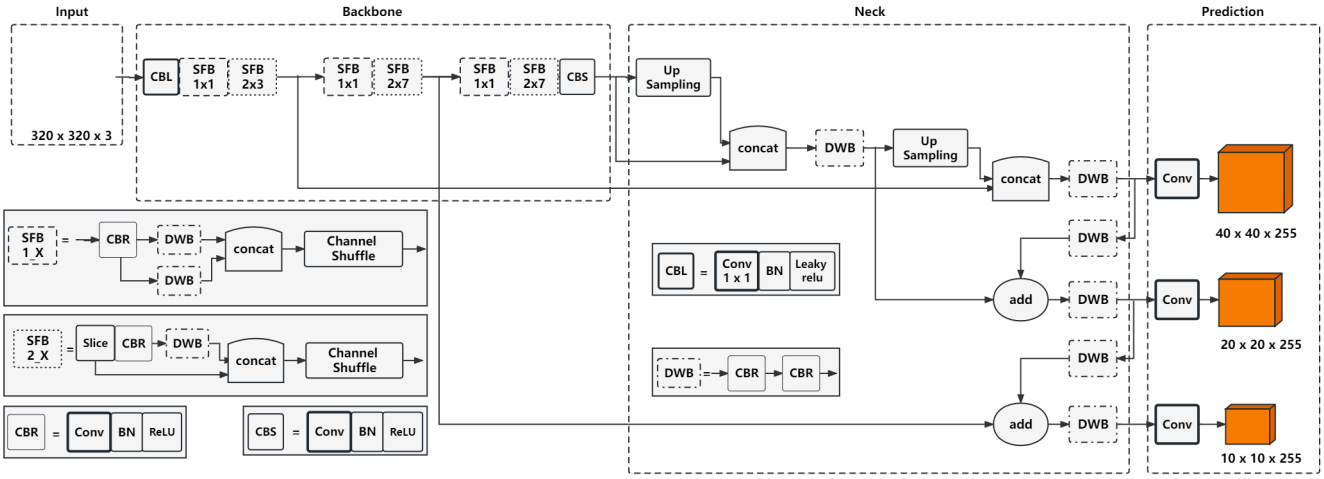

Fig. 1. The architecture of the proposed network. 1) The input is a 320 × 320 three-channel RGB image. 2) The backbone of the proposed network is ShuffleNetV2 [41], which reduces the amount of cache space occupied and increases the inference speed. SFB is the layer structure of ShuffleNetV2. 3) The Neck network part uses a FPN + PAN architecture, with channel pruning of the Head in order to optimise memory access and usage.

II. Background and Related Works

Ample of recent review papers [19]–[25] in fire detection and UAV applications, referencing more than 500 publications combined, conclude that convolutional neural networks (CNNs) which are a branch of deep learning algorithms, are currently achieving the best results in target detection and classification. Two-stage algorithms (R-CNN, Fast R-CNN, Faster R-CNN) [26]–[28] need to use the heuristic (selective search) or CNN network (RPN) to generate Region Proposal, and then do the classification and regression on Region Proposal. The other type of one-stage algorithms (YOLO, SSD) [29], [30], which uses only a CNN network to directly predict the class and location of different targets. The first method is more accurate but slower while the second algorithm is faster but less accurate. However, either networks usually require graphics processing units to operate properly because they are computationally challenging, use large amounts of memory, and are expensive to run. Therefore, deploying real-time target detection algorithms on SBC without powerful computing power is one of the current challenges of computer vision. Hence, works reported in the literature [31]–[33] examine the performance of "You Only Look Once" (YOLO) network which is becoming the most popular object detection model for SBC. As presented, the YOLO versions "tiny", offer higher inference speed (FPS) which however comes at a compromised reduced accuracy because of the reduced number of layers on the network. A successfully deployed YOLOv4 algorithm on a RPi platform achieving an improved inference speed to 2 FPS was reported in [34] while a YOLOv5 algorithm deployed on an unmanned aerial vehicle (UAV) for dynamic beehive detection and tracking achieving an inference speed of 0.5 FPS on RPi4 is reported in [35]. Nevertheless, this speed is still not fast enough for UAV-applications, hence researchers began to experiment with deploying target detection algorithms on the NVIDIA Jetson Nano, at the expense of higher power consumption. An Improved YOLOv5 based Real-time Spontaneous Combustion Point Detection Method was proposed using "FPN + PAN" in the Neck layer of YOLOv5 algorithm to enhance the detecting the smoke or flame in early stage and achieved 7 FPS speed in real-time video stream on NVIDIA Jetson Nano [36]. Finally, the authors of [37] used a single-shot object detector based on deep convolutional neural networks (CNNs) deployed on Odroid-XU4 achieving 8-10 FPS with a accuracy around 95% and 5-6 FPS at 95% on a RPi3B. However, in both cases the CPU usage reached 100%.

III. Proposed Optimization

This work proposes a modified YOLOv5 improving its inference speed by replacing its backbone network and by applying pruning to reduce the computational cost. A fine-tuning technique is implemented on the obtained pruned model to restore the detection accuracy of the model while improving the mapping effect. Additionally, the RPi was overclocked at the hardware level, to achieve better CPU/GPU performance. Both the ordinary YOLOv5 and the proposed optimized one, were re-trained using the same datasets, and the achieved accuracy and inference speed were obtained experimentally.

The proposed network structure as shown in Fig. 1 consists of four main parts: Input, Backbone, Neck, and Prediction. The input is responsible to pre-process the data with mosaic data enhancement, adaptive initial anchor box calculation, and adaptive image scaling. The backbone uses the lightweight ShuffleNetV2 algorithm to improve the YOLOv5s model, which reduces the size of the model and the number of parameters, effectively saving computational resources. The proposed network performs channel pruning at the head and neck according to the backbone network to ensure proper operation of the network.

The pipline of the optimization process is presented in Fig. 2 and starts by replacing the backbone network of the

ordinary YOLOv5 network and modifying the number of head and neck channels to ensure proper operation of the network. Then an iterative process includes sparse training to identify the layers that have a minor effect on the detection results followed by pruning of the small scale factors of the channels and then fine-tuning to recover the detection accuracy of the pruned model. The achieved and desired performances are compared on each iteration and when the desired goal is reached the compact and high performance model is obtained.

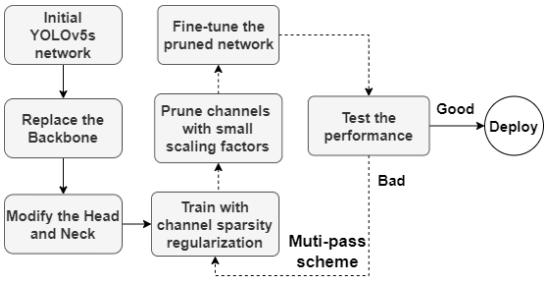

Fig. 2. The pipline of the proposed optimization. Dashed lines denote the iterative process.

A. Replacement of the Backbone Network

Computational complexity and parameter storage have a negative impact on the speed of CNN networks which can be extremely slow when running on computational- and power-constrained devices. Mobile devices need compact models to achieve low latency which leads to accurate and fast detection. To meet these requirements, some lightweight CNN networks such as MobileNet [39] and ShuffleNet [40] have been proposed, which achieve a good balance between speed and accuracy.

The current state-of-the-art lightweight network ShuffleNetV2 [41] is an improved version based on ShuffleNet. [49] ShuffleNetV2 is faster and more accurate than most other networks for the same complexity of inference. Hence, it is ideal for replacing the ordinary YOLOv5 backbone network.

The proposed work achieves network optimization following the four guidelines of the ShuffleNetV2 design: 1) Balance between the input and output feature channel and output feature channels of the convolution layer. 2) Avoidance of use group convolution. 3) Using less number of network branches and avoidance of parallel structures. 4) Keep minimal Element-Wise operations.

B. Network Pruning

One way to reduce the parameter storage and computational cost without compromising accuracy is to Prune the CNN network. Pruning is a technique in deep learning to develop compact and more efficient neural networks. It is a model optimisation technique that involves removing large amounts of redundant neurons and weights in neural network models. The compressed neural network runs faster while it reduces the computational cost of training the network which is particularly critical when deploying models on mobile phones or other edge devices. Li et al [38] proposed an acceleration method for CNNs that directly removes convolutional kernels that have little impact on the accuracy of the CNN, which can greatly reduce the computational cost without compromising accuracy.

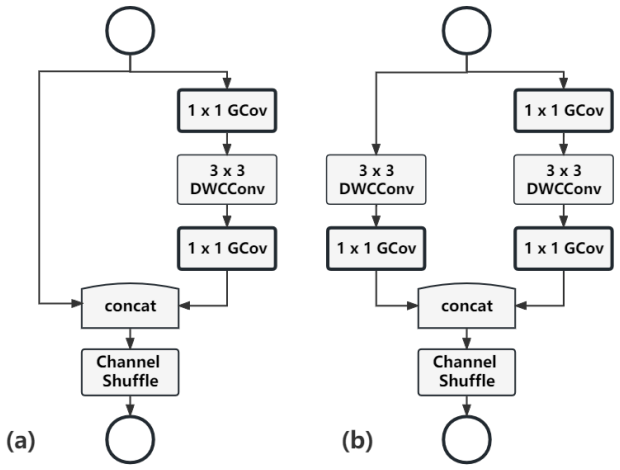

Fig. 3. The layer structure diagram for ShuffleNetV2 [41], which is also the main structure in proposed YOLOv5, corresponds to SFB1_X and SFB2_X in the Fig. 1 respectively.

C. Sparse training and pruning preparation

The channel dimension is pruned after sparse training according to the method of network slimming proposed by Liu et al. [47]. The principle includes two parameters γ, β in the BN layer, and the input is normalised through BN to obtain a normal distribution. When γ, β tends to 0, the input is multiplied by 0, and the convolution on that channel will only output 0. To improve the speed of the network, redundant channels which have minor impact on the detection results of the model are eliminated. When a network is trained without any modification, then the γ is generally distributed around 1 due to initialisation. To converge γ to 0, the coefficients are constrained by the addition of the L1 rule which makes the coefficients sparse also known as sparse training. After sparse training, the filters with small effect to the results can be pruned and the trained model can be further reduced by this process.

The BN layer performs the following transformation :

$$\hat{z} = \frac{z_{in} - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}}; z_{out} = \gamma\hat{z} + \beta \tag{1}$$

The activation size z_{out} per channel is positively correlated with the coefficient γ (which in the pytorch framework corresponds to the weights of the BN layer and β to the bias).If γ is close to 0 then it implies that the activation value is also very small with minor impact on the final result.

After normal network training, the coefficients of the BN layers are similar to a normal distribution. This makes the identification of the importance of each layer very difficult. Whereas, sparse training using the L1 rule distinguishes

the importance of each layer thus enabling the pruning of unimportant layers which leads to higher inference speed.

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma) \tag{2}$$

The first term of equation 2 is the normal training loss function, whereas the second term is the constraint, where $g(s) = |s|$ and λ is the canonical coefficient, adjusted according to the dataset. The back propagation technique is used to update the weight values which minimise the loss function.

$$\begin{aligned} L' &= \sum l' + \lambda \sum g'(\gamma) = \sum l' + \lambda \sum |\gamma|' \\ &= \sum l' + \lambda \sum \gamma * sign(\gamma) \end{aligned} \tag{3}$$

The following points are worth mentioning for the pruning implementation on the YOLOv5 model: 1) YOLOv5 uses automatic mixed precision training, therefore, the FP32 mode is required during training. 2) Not all BN layers are selected for pruning. The implementation removes the Bottleneck layers with shortcut, mainly to ensure that the shortcut is as addable as the residual layer channel. 3) After sparse training, the convolution kernels of the corresponding channels in the convolution layer before the BN layer and the corresponding feature maps after the BN layer are pruned off.

D. Pruning process and Fine-tuning of the pruned model

After obtaining the sparse trained model, the next step is to prune out the channels with γ going to 0. First, the γ of all BN layers are counted and aligned and sorted to find the threshold corresponding to the pruning rate. Then, the mask of each BN layer is obtained according to the threshold, ensuring that the pruned channel is guaranteed to be a multiple of 4, which is required to compound the front-end acceleration. Since, the pruned network does not align with the original network channel, the network needs to be redefined and parsed. The reconstructed network structure needs to be redefined as more parameters need to be imported. After reconstructing and parsing the network, we also need to fill in the parameters of the parsed network, find the parameters of each layer of the parsed network corresponding to the ordinary network, and clone them to the reconstructed network. This completes all the steps of the pruning process.

However, as the network structure of the pruned model changes compared to the original model, while the neural network parameters learned from the original network structure remain unchanged, the target detection capability of the pruned model decreases and the mapping capability is low. The reduction in the accuracy after pruning can be fine-tuned to recover. Since the pruned model is able to relearn the neural network parameters based on the current network structure, it is the finetuning for training and this restores the detection accuracy of the model and improves the mapping effect.

E. Hardware Acceleration

In order to get the best performance out of the algorithms the RPi4B CPU was overclocked to 2.0 GHz (maximum of 2140 MHz). Experimental work (Table III) clearly shows that the faster the clock runs, the more energy is transferred, and the more heat is generated by the chip. As the RPi is normally used with a CPU with its NEON-ARM instructions, the GPU was not overclocked for this study and its default frequency, 500 MHz was used (maximum of 650 MHz). To avoid overheating of the RPi platform, automatic over-voltage adjustment and dynamic clock frequency were used.

IV. Dataset Generation and Training Procedure

A. Dataset

The dataset was generated using a python crawler tool. However, unrelated images were manually removed. Manual was also the process of labelling of the flames and smoke areas on the images. The process of filtering and labeling images was time consuming. The dataset was randomly divided into three independent and equally distributed sets: (i) the Training set, containing 83% of the images (20,255); (ii) the Validation set, containing 13% of the images (3,148); and (iii) the Test set, containing 4% of the images (977). Data augmentation was not a straight forward process because the fire and smoke characteristics (i.e. color disperse, concentration, etc) are significantly affected by the fire intensity, air temperature, the type of flammable material as well as wind speed and direction. Hence, the training set had undergone through randomly selected data augmentation techniques. Given these premises and the inherent difficulties of detection, this learning adds some additional image data to this dataset using data augmentation. Furthermore, images such as sunsets, flashing lights, etc were added to the dataset to reduce the model's false detection of fire-like objects that could trigger false alarms, as shown in Fig. 4. As a result, the dataset becomes robust and accurate under abstract conditions and patterns including camera angles.

Fig. 4. Some examples of the dataset: (a) and (b) show image based data augmentation techniques to expand the dataset. (c) and (d) show respective ground-truth bounding boxes. (e) and (f) represent images that are prone to false detection. (g) is a ground-based image of the forest fire and (h) is an aerial view of the fire.

B. Training procedure

Initially, the YOLOv5 network for the proposed fire detection system needed to undergo through training. Following the recommendations of [42] and [43] the training process included the MGD-algorithm (Mini-batch Gradient Descent) which required approximately 30,000 iterations. The proposed size and momentum were 64 and 0.937 respectively. Worth noting that the GPU simultaneous processing was limited to a maximum of 64 images for as long as the mini-batch size of 64 had undergone through additional subdivision to a total of 317 partitions. Over-fitting was avoided by stopping the training process when no-improvement was achieved for 5,000 iteration. Warm-up technique [44] with a warm-up momentum 0.8 and warm-up epochs 3.0 were used to improve the training. A warm-up bias learning rate of 0.1 has been used during the warm-up process. As proposed in [45], the image size (N) was selected to 320 which lead to a grid cell size (S=N/32) of 10 which then lead to 3 bounding boxes per grid cell (B).

The ordinary YOLOv5s model detects two classes (Fire and Smoke). The forward propagation pass when processing a single RGB image of size 320×320 pixels, occupies the use of 15.8 GFLOPs. However, the popular YOLOv3 [46] requires 154.6 GFLOPs for the same process. To achieve a clear comparison, the Ordinary YOLOv5 network and the optimized YOLOv5 network were trained with the same hyper-parameters mentioned above. In addition, the weights of the layers were initialized through the transfer learning in the COCO (Microsoft Common Objects in Context) dataset. The proposed model uses input image size of 320 and the weight of each category is related to the number of labels.

The Adam optimization method [50] 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 300 epochs each comprising of 317 batch iterations, with a batch size of 64.

V. Evaluation and Experimental Results

A full quantitative evaluation of the fire and smoke detectors for YOLOv5 and its optimised version is performed and presented in this section. The training and inference phases were conducted on a machine with 16 GB of RAM using a single GPU NVIDIA RTX3050. The system was running Ubuntu 20.04 LTS operating system and CUDA compiler tools, version 11.7. The training and optimization procedures were implemented in Python using the PyTorch [31] framework. The platform used to test and deploy the algorithms is the RPi4B with Broadcom BCM2711, quad-core Cortex-A72 (ARMv8) 64-bit SoC@1.5GHz, 4GB memory, OpenGLES 3.1, Vulkan 1.0 and a 64GB Micro-SD card for both operating system installation and data storage. To evaluate the performance of the proposed network model, the most widely used object detection evaluation metrics were used in the experiments, which include Intersection over Union (IoU), precision rate, recall rate, average mean precision (mAP), F_1-score and detection speed (inference time per image).

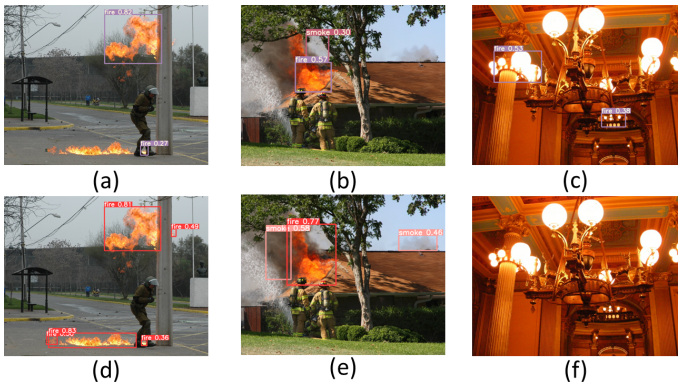

Fig. 5. Some predictions results from the BoWFire dataset [48], (a)-(c) are the results for the original YOLOv5 and (d)-(f) are the results for the proposed YOLOv5. The detection results show that the proposed YOLOv5 can mark out more areas where flames and smoke are present in the pictures and there are no false detections in pictures with flame-like objects.

TABLE I
Performance metrics on the test set

Network	mAP@0.50 (%)	AP@0.50 smoke (%)	AP@0.50 fire (%)	F_1 score	avg IoU (%)
YOLOv5	82.93	91.17	74.86	0.83	73.51
Proposed	92.54	96.35	88.71	0.85	68.34

A. Comparison of models' performances on Prediction

Once the original and the proposed YoloV5 networks have been trained (see Sect. IV-B), their performance was evaluated through a fivefold cross-validation procedure. Table I, shows that the optimised YOLOv5 outperforms the original YOLOv5 across most of the evaluation metrics considered and the optimised one has more true positives (real fire identification) and fewer false positives than the original YOLOv5 network when detecting images that could be easily confused as flames as shown in Fig. 5. However, the location of objects detected by the original YOLOv5 network is on average more accurate in terms of IoU experimental values. This is because the original YOLOv5 retains more convolutional layers than the optimised network, although the inference speed is much slower, 1 FPS compared to 9 FPS as shown on Table III.

Fig. 6. Evaluation on the Raspberry Pi platform: the detection result of the image obtained in real time from the web camera, with the live detection frame rate in the top left corner of the detection screen.

TABLE II
PARAMETER COMPARISON OF THE PROPOSED DETECTION MODEL UNDER DIFFERENT PRUNING RATES.

Pruning Rate(%)	mAP@0.5(%)	Parameters/10^6	Model Size(MB)
Baseline	93.7	7.02	14.1
80	82.3	0.89	1.95
70	89.7	1.54	3.34
60	91.2	2.30	4.64
50	93.2	2.85	5.66

B. Analysis of the model pruning results

Once determining a scale factor γ = 0.005 for sparse training, then the effectiveness of model pruning was verified using pruning rates of 0.5, 0.6, 0.7 and 0.8 to the sparsely trained model. Parameter comparison of the proposed detection models for different channel pruning rates are tabulated on Tab. II, According to Tab. II, all 4 evaluation metrics were reduced for different channel pruning rates. After performing fine-tuning training, the mAP recovered to 84.87%, 92.5%, 92.65% and 93.41% respectively. Fine-tuning revealed that the channel pruning rate of 70% achieved the best balance between accuracy and inference speed, which resulted in a better model compression with less loss of average accuracy.

TABLE III
PERFORMANCE METRICS ON THE COMPARISON EXPERIMENT

Network	Clock Speed (GHz)	Power (W)	CPU Usage (%)	CPU Temp ($^\circ C$)	FPS
YOLOv5	0.6	5.70	93	47	0.41
YOLOv5	1.8	7.70	95	58	1.06
YOLOv5	2.0	8.10	97	67	1.16
Proposed	0.6	5.20	60	43	3.02
Proposed	1.8	7.02	58	47	7.23
Proposed	2.0	7.28	64	50	8.57

Comparison results on the performance metrics are tabulated on Tab. III.As presented the proposed algorithm increases the inference speed by a factor of 7.4 at all clock speeds, reaching a highest value of 8.57 FPS at 2GHz CPU clock speed. In addition, the proposed network achieves a reduction of the CPU usage in the range of 35% (from high 90s to low 60s) which also translates in a 25% reduction of CPU temperatures. Compared to the state-of-the-art literature presented in previous sections, this work increases inference speed by 50% and reduces the CPU usage by 35%. Finally, the proposed work achieves a 10% reduction in electrical power consumption which prolongs operation run-time and range of applications involving small UAVs. As research shows [51] the battery run-time is not linear but exponential.

C. Optimized YOLOv5 on Raspberry Pi Platform

In order to map the proposed model on the RPi-4B platform, the onnx-runtime framework was used to implement the proposed network structure. The runtime environment was installed on the RPi-4B, and the PyTorch weights were converted to the onnx format. A web camera was used to acquire live video at a size of 320×320 pixels and pass it frame by frame to the processing board where the flames and smoke were detected. The model detection results are shown in Fig. 6. The performance of the RPi was approximately 7-9 FPS and the accuracy remained at approximately 92.5%. It is worth noting that the optimised YOLOv5 model on the RPi platform is over 7 times faster and uses less CPU usage and power consumption than the original YOLOv5, as shown in Tab. III. In the case of deploying a RPi as an on-board computer on a small UAV, the optimised algorithm can use the remaining CPU usage for other purposes of computing. Furthermore, less power consumption will increase battery runtime and range which are of utmost importance for small UAV patrol missions. Finally, it is worth noting that the accuracy of the proposed algorithm is not affected by deviations in the bird eye view angle.

VI. CONCLUSION

The work reported in this paper improves the performance of the YOLO model by replacing the lightweight backbone network, sparsely training the model, pruning and fine tuning. The optimised YOLOv5 produced the highest mAP of 92.5% compared to the ordinary YOLOv5s model and could detect at 7-9 FPS on the RPi-4B, indicating that the optimised model can detect potential fires excellently on a low computing power embedded platform. Furthermore, the optimised model use 35% less CPU usage than the original YOLOv5 because of the reduction in Flops, memory usage, and parameters during the optimisation process. The reduced CPU usage also translated to 25% reduction in CPU temperature. The approach uses both software and hardware level acceleration and the experimental results show that the proposed forest fire detection system is more suitable for small UAV-based fire detection tasks in terms of power consumption, size and weight compared with the NVIDIA Jetson Nano. Overall, the optimisation of the YOLOv5 model improves the detection accuracy and inference speed. The deployment approach in this study reduces the difficulty of deploying the deep-learning fire detection model on edge devices.

REFERENCES

[1] H. Heidari, M. Arabi and T. Warziniack, "Effects of Climate Change on Natural-Caused Fire Activity in Western U.S. National Forests", Atmosphere, 981, 1-12, 2021.

[2] J.E. Halofsky, D.L. Peterson and B.J. Harvey, " Changing wildfire, changing forests: The effects of climate change on fire regimes and vegetation in the Pacific Northwest, USA", Fire Ecol. 16, 1–26, 2020.

[3] P. Gao, A.J. Terando, J.A. Kupfer, J, M Varner, M.C. Stambaugh, T.L. Lei and J. K. Hiers, "Robust projections of future fire probability for the conterminous United States", Sci. Total Environ. 789, 2021.

[4] H. Heidari, T. Warziniack, T.C. Brown and M. Arabi, "Impacts of climate change on hydroclimatic conditions of US national forests and grasslands", Forests 139, 1-12, 2021.

[5] W.S. Keeton, P.W. Mote and J.F. Franklin, "Chapter 13—Climate variability, climate change, and western wildfire with implications for the urban-wildland interface", Adv. Econ. Environ. Resour. 6, 225–253, 2007.

[6] A. Vaughan, "Wildfire pollution linked to at least 33,000 deaths worldwide", Health, Newscientist, Online Posting: https://www.newscientist.com/article/2289547-wildfire-pollution-linked-to-at-least-33000-deaths-worldwide/ [Accessed 27/10/22].

[7] Statista 2022, “Most severe wildfires by number of fatalities worldwide between 1900 and 2021”, Online Posting: https://www.statista.com/statistics/267801/death-toll-due-to-wildfires/ [Accessed 27/10/22].

[8] IQ Fire Watch, “Global Impacts of Wildfires – Damages, Losses, Costs and Effects”, Online Posting: https://www.iq-firewatch.com/risk [Accessed 27/10/22].

[9] World Wild Life, “Fire Management”, Online Posting: https://www.worldwildlife.org/stories/fire-management [Accessed 27/10/22].

[10] P. Howard, “The Cost of Carbon Project - Flammable Planet: Wildfires and the Social Cost of Carbon”, Institute of Policy Integrity, NY University, School of Law, 2014.

[11] Panagiotis Barmpoutis, Periklis Papaioannou, “A Review on Early Forest Fire Detection Systems Using Optical Remote Sensing,” 11 November 2020. Online Posting: https://doi.org/10.3390/s20226442. [Accessed 16 November 2022].

[12] H. Shakhatreh et al., ”Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges,” in IEEE Access, vol. 7, pp. 48572-48634, 2019, doi: 10.1109/ACCESS.2019.2909530.

[13] P. Petrides, C. Kyrkou, P. Kolios, T. Theocharides, and C. Panayiotou. 2017. Towards a holistic performance evaluation framework for drone-based object detection. In 2017 International Conference on Unmanned Aircraft Systems (ICUAS). 1785–1793. https://doi.org/10.1109/ICUAS.2017.7991444

[14] C. Kyrkou, S. Timotheou, P. Kolios, T. Theocharides, and C. G. Panayiotou, “Optimized vision-directed deployment of UAVs for rapid traffc monitoring” IEEE International Conference on Consumer Electronics, ICCE 2018, Las Vegas, NV, USA, January 12-14, 2018. 1–6. https: //doi.org/10.1109/ICCE.2018.8326145

[15] J. del Cerro et al., “Unmanned Aerial Vehicles in Agriculture: A Survey”, Agronomy, 11, 203, 2021. https://doi.org/10.3390/agronomy11020203.

[16] A. Meola, “Precision agriculture in 2021: The future of farming is using drones and sensors for efficient mapping and spraying”, (Feb 8, 2021), Online Posting: https://www.businessinsider.com/agricultural-drones-precision-mapping-spraying [Accessed 3/1/23].

[17] S. Ioannou, “Discrete Linear Constrained Multivariable Optimization for Power Sources of Mobile Systems”, PhD Dissertation, Electrical Engineering, University of South Florida, November 2008.

[18] D. Hulens, J. Verbeke and T. Goedemé, “How to Choose the Best Embedded Processing Platform for on-Board UAV Image Processing ?”, 10th International Conference on Computer Vision Theory and Applications (VISAPP-2015), pages 377-386, 2015.

[19] Keiron O’Shea, Ryan Nash, “An Introduction to Convolutional Neural Networks,” 26 November 2015. Online Posting: https://arxiv.org/abs/1511.08458. [Accessed 18 September 2022].

[20] F.M. Anim Hossain, Y. M. Zhang and M. A. Tonima, “Forest fire flame and smoke detection from UAV-captured images using fire-specific color features and multi-color space local binary pattern”, Journal of Unmanned Vehicle Systems, 30 June 2020, https://doi.org/10.1139/juvs-2020-0009

[21] Y. Zhao, J. Ma, X. Li and J. Zhang, “Saliency Detection and Deep Learning-Based Wildfire Identification in UAV Imagery”, Sensors, 18(3), 712, 2018. https://doi.org/10.3390/s18030712

[22] A. Ramachandrana and A. K. Sangaiah, “A review on object detection in unmanned aerial vehicle surveillance”, International Journal of Cognitive Computing in Engineering, Volume 2, pp 215-228, 2021.

[23] F. Ahmed and M. Jenihhin, “A Survey on UAV Computing Platforms: A Hardware Reliability Perspective”, Sensors, 22, 6286, 2022. https://doi.org/10.3390/s22166286

[24] M. Iqbal, C. Setianingsih and B. Irawan, ”Deep Learning Algorithm for Fire Detection,” 2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), 2020, pp. 237-242, doi: 10.1109/EECCIS49483.2020.9263456.

[25] K. Manoj et al. “Smoke and Fire Detection using Deep Learning: A Review”, International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Vol. 2, Issue 1, November 2022.

[26] Girshick, Ross, et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” 2014 IEEE Conference on Computer Vision and Pattern Recognition, June 2014. Crossref, https://doi.org/10.1109/cvpr.2014.81.

[27] Girshick, Ross. ”Fast r-cnn.” In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.

[28] Ren, Shaoqing, et al. ”Faster r-cnn: Towards real-time object detection with region proposal networks.” Advances in neural information processing systems 28 (2015).

[29] Redmon, Joseph, et al. ”You only look once: Unified, real-time object detection.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[30] Liu, Wei, et al. ”Ssd: Single shot multibox detector.” European conference on computer vision. Springer, Cham, 2016.

[31] H. Feng, et al., “Benchmark Analysis of YOLO Performance on Edge Intelligence Devices”, Cryptography 2022, 6, 16. https://doi.org/10.3390/cryptography6020016

[32] U. Nepal and H. Eslamiat, “Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs”, Sensors 2022, 22, 464. https://doi.org/10.3390/s22020464

[33] M. E. Atik, Z. Duran and R. Ojgunluk, “Comparison of YOLO Versions for Object Detection from Aerial Images”, International Journal of Environment and Geoinformatics (IJEGEO), Vol. 9, Issue 2, 2022.

[34] Song Han, Huizi Mao, William J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, 15 February 2016. Online Posting: https://arxiv.org/abs/1510.00149. [Accessed 16 November 2022].

[35] Wahyutama, A.B.; Hwang, M. YOLO-Based Object Detection for Separate Collection of Recyclables and Capacity Monitoring of Trash Bins. Electronics 2022, 11, 1323. [Accessed 16 November 2022].

[36] Gao, P., Lee, K., Kuswidiyanto, L.W. et al. Dynamic Beehive Detection and Tracking System Based on YOLO V5 and Unmanned Aerial Vehicle. J. Biosyst. Eng. (2022). https://doi.org/10.1007/s42853-022-00166-6

[37] C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris and C. S. Bouganis, “DroNet: Efficient convolutional neural network detector for real-time UAV applications”, 2018 Design, Automation & Test in Europe Conference & Exhibition, 2018. doi:10.23919/date.2018.8342149

[38] Li, Hao, et al. ”Pruning filters for efficient convnets.” arXiv preprint arXiv:1608.08710 (2016).

[39] Howard, Andrew G., et al. ”Mobilenets: Efficient convolutional neural networks for mobile vision applications.” arXiv preprint arXiv:1704.04861 (2017).

[40] Zhang, Xiangyu, et al. ”Shufflenet: An extremely efficient convolutional neural network for mobile devices.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[41] Ma, Ningning, et al. ”Shufflenet v2: Practical guidelines for efficient cnn architecture design.” Proceedings of the European conference on computer vision (ECCV). 2018.

[42] Bottou L (1998) Online learning and stochastic approximations. Online Learn Neural Netw 17(9):142

[43] Polyak BT (1964) Some methods of speeding up the convergence of iteration methods. Ussr Comput Math Math Phys 4(5):1–17

[44] Redmon J, Divvala S, Girshick R, Farhadi A (2016) You Only Look Once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788

[45] Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271

[46] Redmon, Joseph, and Ali Farhadi. ”Yolov3: An incremental improvement.” arXiv preprint arXiv:1804.02767 (2018).

[47] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[48] Chino DY, Avalhais LP, Rodrigues JF, Traina AJ (2015) BoWFire: detection of fire in still images by integrating pixel color and texture analysis. In: 2015 28th SIBGRAPI conference on graphics, patterns and images. IEEE, pp 95–102

[49] Zhang, Xiangyu, et al. ”Shufflenet: An extremely efficient convolutional neural network for mobile devices.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[50] Kingma, Diederik P., and Jimmy Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).

[51] S. Ioannou, K. Dalamagkidis, E. K. Stefanakos, K. P. Valavanis and P. H. Wiley, ”Runtime, capacity and discharge current relationship for lead acid and lithium batteries,” 2016 24th Mediterranean Conference on Control and Automation (MED), Athens, Greece, 2016, pp. 46-53, doi: 10.1109/MED.2016.7535940.