Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

A real-time detection model for smoke in grain bins with edge devices

Hang Yin [a,b,*], Mingxuan Chen [b], Yinqi Lin [b], Shixuan Luo [b], Yalin Chen [b], Song Yang [c], Lijun Gao [d,**]

[a] *College of Big Data and Internet, Shenzhen Technology University, Shenzhen, 518118, China*
[b] *College of Information Science and Technology, Zhongkai University of Agriculture and Engineering, Guangzhou, 510225, China*
[c] *College of Software, Dalian University of Foreign Languages, Dalian, 116044, China*
[d] *College of Computer Science, Shenyang Aerospace University, Shenyang, 110136, China*

A R T I C L E I N F O

Keywords:
Grain bins
Real-time smoke detection
Sparse training
Channel pruning
Edge devices

A B S T R A C T

The global food crisis is becoming increasingly severe, and frequent grain bins fires can also lead to significant food losses at the same time. Accordingly, this paper proposes a model-compressed technique for promptly detecting small and thin smoke at the early stages of fire in grain bins. The proposed technique involves three key stages: (1) conducting smoke experiments in a back-up bin to acquire a dataset; (2) proposing a real-time detection model based on YOLO v5s with sparse training, channel pruning and model fine-tuning, and (3) the proposed model is subsequently deployed on different current edge devices. The experimental results indicate the proposed model can detect the smoke in grain bins effectively, with mAP and detection speed are 94.90% and 109.89 FPS respectively, and model size reduced by 5.11 MB. Furthermore, the proposed model is deployed on the edge device and achieved the detection speed of 49.26 FPS, thus allowing for real-time detection.

1. Introduction

Currently, various information technologies are employed in agriculture, ranging from cultivation and storage to transportation and market [1]. In grain storge, some situations are needed to take attention, such as plague of insects [2] and fire [3]. There are four types of grain bins shown in Fig. 1, including transit warehouse (a), storge warehouse (b), repository (c) and comprehensive warehouse (d), which we obtained by the document of China government from Intent [4,5]. The process of storing grain can lead to fires in grain bins due to the microbial fermentation of the grain itself, resulting in significant losses of life and property [6]. If the smoke emanating from the early stages of fire can be detected within the bins, an early warning system can be triggered, and the emergency can be promptly resolved. However, during earlier times, fire and smoke warnings for grain bins were primarily based on various sensors. Despite the sensors' ability to provide some early warning [7], they required high thresholds for smoke concentration, size, indoor temperature, and humidity detection, and cannot achieve real-time detection. Therefore, it is impractical to detect smoke of grain bins with sensors.

The paper is expounded upon in subsequent passages. section 2 elucidates the Related Work. In Section 3, Materials and Methods is

* Corresponding author. College of Big Data and Internet, Shenzhen Technology University, Shenzhen, 518118, China.
** Corresponding author.
E-mail addresses: yinhang@sztu.edu.cn (H. Yin), gaolijun@sau.edu.cn (L. Gao).

https://doi.org/10.1016/j.heliyon.2023.e18606
Received 1 May 2023; Received in revised form 21 July 2023; Accepted 24 July 2023
Available online 31 July 2023

introduced. Subsequently, in Section 4, the Results is presented. Lastly, Sections 5 and 6 respectively shed light on the Discussion and Conclusion.

2. Related work

As artificial intelligence continues to advance, computer vision techniques have been employed to detect smoke. These techniques can be broadly classified into two categories, which are based on machine learning and deep learning convolutional neural networks. Smoke was detected by identifying the irregular motion of objects, which was a common feature of both [8]. It was achieved using motion estimation with the optical flow method, followed by feature extraction using wavelet analysis, Weber contrast, and color, which allowed for the classification of moving spots corresponding to smoke. Features, such as color, wavelet coefficients, motion direction, and directional gradient histograms were used to extract information for smoke detection [9]. The extracted features were then classified using decision trees and random forests. Although the machine learning-based smoke detection approach yields good results, the methods may result in lower detection accuracy compared to the convolutional neural network-based method. This is because its feature extraction process is more subjective and can overlook characteristics that humans are unable to observe.

With the impressive advances in GPU computing power [10] and the limitations of machine learning methods to accurately detect smoke, deep learning convolutional neural networks have become the focus of smoke detection research. Among the classic convolutional neural networks that have been developed we can mention Alexnet [10], Vggnet [11], among others. To enhance the accuracy of smoke detection, attention mechanisms have gained popularity in recent years, as demonstrated by Jiang and Yin [12,13], all of which have yielded desirable results [14]. Fire smoke detection was performed in forest fire scenarios [15–18]. As agricultural straw burning could contribute to air pollution, Jiang [12] used the Efficient-det2 object detection model based on the Resnet feature extraction network. They employed the self-attentive mechanism GA-BLOCK to solve false negatives caused by insufficient consideration of realistic environments. Meanwhile, Yin [13] targeted small smoke detection in urban scenarios. The YOLO v5 lightweight target detection model was enhanced by adding a double attention mechanism to improve the weighting of small smoke features during training. In Ref. [15], forest fire smoke detection was performed using deeply separable convolution for the feature extraction network, which had model size one-fifth that of Alexnet. Finally, it led to good classification results in forest smoke detection. Zhan [16] also focused on forest fire smoke detection, but with the inclusion of a more robust dataset from Unmanned Aerial Vehicles. They improved upon the PP-YOLO object detection network by implementing a two-layer feature extraction network, with one layer responsible for primary feature extraction for smoke and the other for secondary extraction. Additionally, the detection frame recognition algorithm was improved to propose GO-NMS, which yielded satisfactory results. Despite achieving a high level of accuracy, smoke detection based on deep learning convolutional neural networks is prone to latency issues caused by network transmission. In order to detect forest fires accurately [17], proposed a state-of-the-art network architecture named MVMNet. This innovative network introduced a multi-directional detection frame method to detect the source of smoke and the proposed network exhibited a noteworthy improvement in detection precision and speed. In a separate study conducted by Ref. [18], drones were employed to address the issue of early-stage forest fire smoke detection. Additionally, their proposed method demonstrated superior resulted when compared to traditional networks. However, it should be noted that there are some differences when it comes to detecting smoke from grain bins as compared to those from forests and urban environments. It should be noted that smoke characteristics within indoor grain bin environments are typically smaller and thinner than those of forest and urban fire. Therefore, leveraging datasets from other smoke sources may not necessarily enhance the generalizability of smoke detection models for grain bins. In addition, this presents a significant challenge to achieving real-time smoke detection if network transmission conditions are taken into account.

In agriculture, deploying algorithms on edge computing devices is an effective method to avoid network latency and enable automated control. An innovative lightweight network was introduced in Ref. [19] for smoke and fire detection in forests, using close to 60,000 datasets in total. One of their innovation lies in the deployment of smoke detection on the cloud and an unmanned aerial vehicle, which yielded promising results. However, the application scenarios in Ref. [19] were primarily focused on forests and may not be suitable for grain bins deployment. Conversely, Zhang, Jiao and Mazzia [20–22] employ edge computing to facilitate fruit ripeness determination and planting health management for fruit farmers, specifically in the context of orchard robots. To address the challenge of visually inspecting strawberry ripeness, Zhang [20] utilized an unmanned aerial vehicle to collect data and reduced the YOLOv4 tiny [23] model's weight by adjusting the number of model layers, which resulted in successful strawberry detection on edge devices. To improve the efficiency of automated litchi picking in complex environments, Jiao [21] developed an edge computing-based litchi detection system that employed the YOLO v3 model and a novel lightweighting algorithm. In Ref. [22], YOLOv3 was employed in

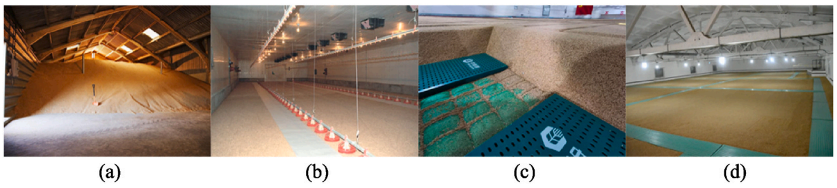

Fig. 1. Internal environment and storage methods of grain bins: (a) Transit Warehouse; (b) Storge warehouse; (c) Repository; (d) Comprehensive Warehouse. The function of (a) is of a temporary nature. (b) Has a limited storage capacity, in contrast to (c), which is utilized for large-scale storage. (d) Encompasses all of the aforementioned functionalities.

an edge device to enhance the efficiency of automated apple picking. The detection frame was narrowed to achieve small target apple detection at a distance, and different edge devices, such as Raspberry Pi, Jeston Nano, and Jeston AGX Xavier, were compared. Ultimately, Jetson Nano was proved to be the ideal device for experimentation.

Despite the fact that convolutional neural networks and edge devices have already been applied in agriculture, there has been a paucity of research into their use in grain bins. In grain bins settings, smoke detection is of the utmost importance in the event of fire, necessitating real-time detection as one of top priority. Therefore, deploying models that detect smoke in grain bins on edge devices is imperative. Due to their limited storage capacity and computational constraints, edge devices require models with reduced parameters without much compromising on smoke detection accuracy. This presents two significant challenges: first, lack of online smoke datasets specific to grain bins settings; and second, smoke detection in grain bins needs to be both real-time and accurate in the situation of network latency, making edge devices the ideal choice. However, existing models with high smoke detection accuracy are not conducive to be deployed on edge devices due to their large model size. Thus, in this paper, several innovations are introduced: (1) To address the lack of smoke photos datasets specifically for grain bins applications, collecting data in a back-up grain bin is an ideal method. Experiments of different types of smoke sources also adds to the diversity of the dataset, allowing for more comprehensive training of the smoke detection model. (2) To achieve real-time smoke detection in grain bins, the proposed model is designed based on lightweight object detection model YOLO v5s. Subsequently, the proposed model is trained with sparse constrains to lay the foundation for pruning channels, which greatly reduce the size of the model. Finally, the model is fine-tuned to restore detection accuracy. (3) To avoid the influence of network latency, the smoke detection model that has been trained is deployed on different edge devices.

3. Material and methods

3.1. System setup

An effective method for real-time smoke detection in grain bins during the early stages of fire is proposed. Fig. 2 illustrates the flow chart of conventional smoke detection in a grain reserve with our proposed method and with traditional sensors respectively.

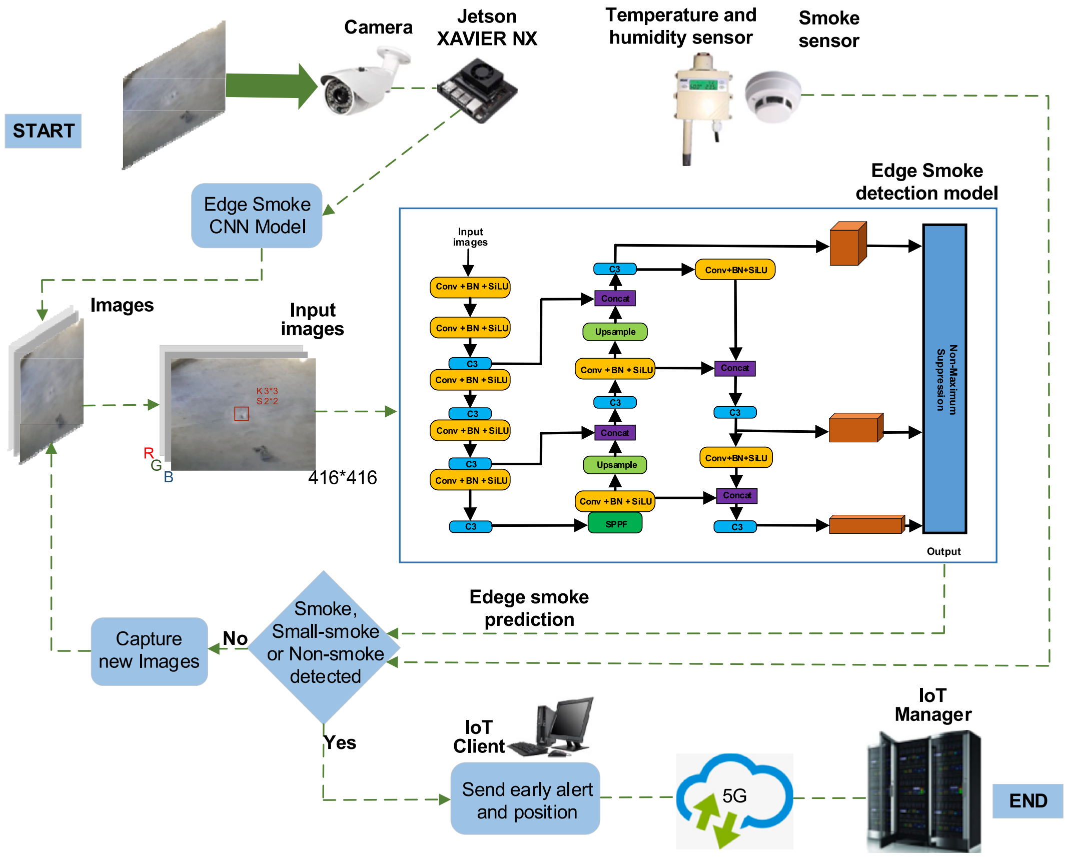

Fig. 2. General flow chart.

3.2. Data collection and processing

In order to address lack of the public datasets of grain bins, our team obtained access to a back-up bin of grain reserves that is similar to the scenes of Fig. 1 for conducting smoke experiments. In Ref. [24], 3 types of smoke are recommended, including smoke from the smoke emitter, smoke from the smoke emitter sheet, smoke from the cotton strings. As shown in Fig. 3, it is self-created grain bins smoke dataset. The smoke emitter allows for adjustable smoke size and is manually operated, as shown in Fig. 3 (a). Smoke emitted from the burning of the smoke emitter sheet and cotton string, as illustrated in (b) and (c) of Fig. 3, respectively. They more closely resemble the reality of grain combustion due to the similar composition of these emitters and that of grain.

To improve the model's robustness and establish a foundation for future edge deployment, we apply mosaic data augmentation to the dataset. Fig. 4 demonstrates random crops of the smoke dataset

Being taken, followed by horizontal/vertical flipping, 90-degree rotation, shrinking. Finally, they are stitched together into an image. This method of data augmentation not only offers a more various and comprehensive context for the grain bin smoke dataset, but also accelerates the training speed and improves model convergence by processing 4 images.

After undergoing data augmentation, the self-created smoke dataset for grain bins comprises a total of 9350 images. The dataset is categorized into Smoke, Small Smoke. The classification between smoke and small smoke is whether smaller than 100 pixels of our dataset, pictures with 1920 × 1080. The dataset was split into 90% for the training and validation sets, and 10% for the test set. In addition, the distribution of images between the training set and validation set follows a ratio of 9:1. As depicted in (Table 1), our experimental training set comprises 7573 images, the validation set has 842 images, and the test set contains 935 images.

3.3. YOLO v5s objection detection model

Convolutional neural network-based object detection models can be classified into two types: two stage and one stage. Although the two-stage object detection model has higher detection accuracy than the single-stage model, the one-stage object detection model has fewer parameters and is faster [25]. Since real-time smoke detection is crucial for grain bins, we choose a one-stage object detection model for our experiments. The YOLO series of models is representative of the one-stage target detection models.

Since the Joseph Redmon released the first YOLO model in 2016, it has gone through YOLO v2 [26], YOLOv3 [27], YOLO v4 [23], and now YOLO v5, which is still undergoing updates. The YOLO v5 series comprises five different versions, including n, s, m, l, x, depending on the size of their parameters. After experimental validation, the YOLO v5s is selected as our object detection model of choice. The YOLO v5s model employs three distinct loss functions: classes loss, abjectness loss, and location loss.

The architecture of the YOLO v5s object detection model consists of backbone network, neck network, and detection network, each with distinct functions, as illustrated in Fig. 5. (1) The core of the entire model is the backbone network, a convolutional neural network that extracts features from the image data. YOLO v5s employs the improved CSP-Darknet53 as its feature extraction network, which is compared to CSP-Darknet19 used in previous YOLO. (2) The neck network is the network in which the extracted features are fused. The authors here draw on other excellent network structures, Path Aggregation Network (PANet) [28] and Spatial Pyramid Pooling (SPP) [29]. (3) The detection network's essential component is the classification and regression network. The network classifies and divides candidate boxes into original feature maps according to the detect size. Each candidate box consists of 3 bounding box sizes and 6 pieces of information. Using features extracted by the neck network, the detection network locates the Bounding Boxes and eliminates overlapping boxes through non-maximum suppression.

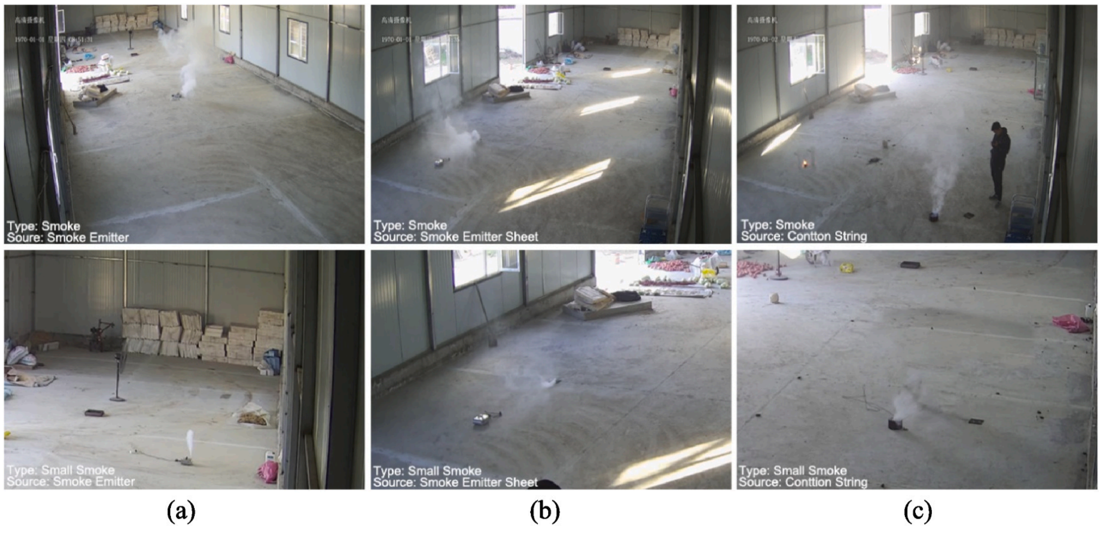

Fig. 3. Self-created grain bins smoke dataset: (a) Smoke Emitter; (b) Smoke Emitter Sheet; (c) Cotton String. The photos are the dataset of smoke in the first row and small smoke in the second row.

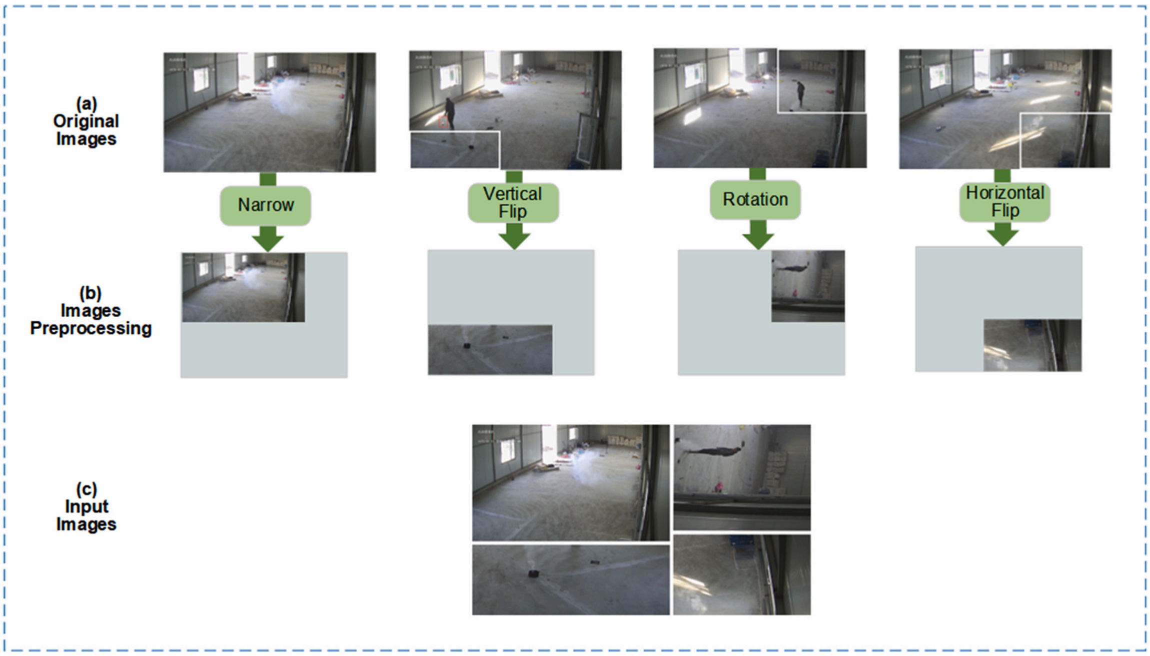

Fig. 4. Mosaic data enhancement.

Table 1
Dataset division.

Type	Train	Validation	Test	Total
Smoke	3361	374	415	4150
Small-smoke	4212	468	520	5200
Total	7573	842	935	9350

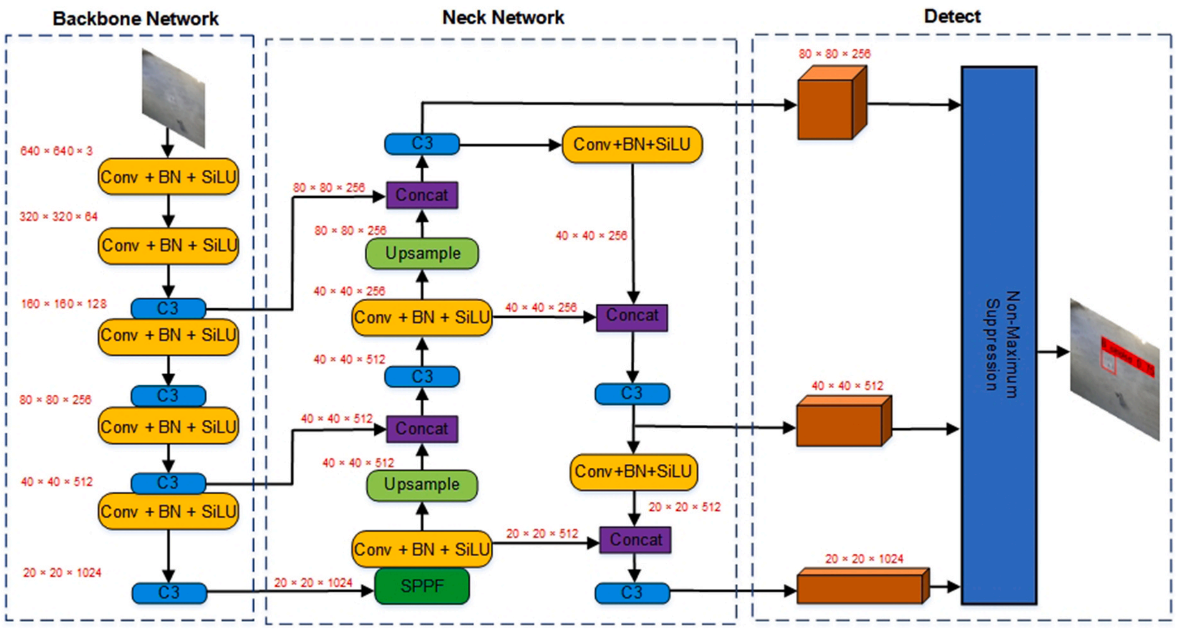

Fig. 5. YOLO v5s network.

3.4. Model compression

Although the YOLO v5s model is effective in our grain bins smoke detection application, we recognize the need for further improvements in detection speed in real grain bins scenarios where time is of the essence during fire rescue operations. Therefore, we have implemented a series of model compression techniques to optimize the YOLO v5s model. Since our dataset contains only two classification categories, smoke and small smoke, the impact of certain feature extraction techniques on detection accuracy is expected to be minor [30]. Thus, model compression will not significantly impact detection accuracy on small-classification datasets. As shown in Fig. 6, we utilize. an iterative approach that involves sparse training, channel pruning, model fine-tuning and 5% pruning per iteration. The model is initially trained with L1 regularization constraints on its channels. Batch Normalization (BN) [31] operation is then employed to determine whether to perform channel pruning to remove channels based on the uncompressed model. The pruned model is then fine-tuned on our dataset, and accuracy is restored through migration until the desired pruning rate is achieved.

3.4.1. Sparse training

In order to determine the importance of each channel in the YOLO v5s model, we use L1 regularization sparse training without altering the model structure. This is because the weight factor within each BN module corresponds to a specific channel. By evaluating the size of the BN weight factor, the significance of each channel in the model will be judged.

Equations (1) and (2) represent the key components of the BN operation. Z_{in} and Z_{out} are the input and output channel numbers, respectively. μ_B and σ_B^2 represent the mean and variance of the activation in BN. And c is the bias of the equation. While γ and β are the weight and shifting e parameters for the BN linear transformation. After the operation of equation (2), the distribution of data is from zero to one. Therefore, That the γ and β in equation (1) are adaptive can improves the robustness of the model. Equation (3) presents the training loss that includes the normal convolutional neural network and the L1 regularization function g within Γ in training time. Characters, x and y refer to the input and target values during training. W is the weight during training. Meanwhile, f (x, W) refers that input processed through the weights. In addition, the sparsity factor λ is also included in the formula. Equation (4) is the L1 regularization function, which is the sum of the absolute values of each weight.

$$Z_{\text{out}} = \gamma \hat{z} + \beta \tag{1}$$

$$\hat{z} = \frac{z_{\text{in}} - \mu_B}{\sqrt{\sigma_B{}^2 + c}} \tag{2}$$

$$L = \sum_{x,y} l(x, y) + \sum_{\gamma \in r} g(\gamma) \tag{3}$$

$$\|\gamma\|_1 = |\gamma_1| + |\gamma_2| + \cdots + |\gamma_n| \tag{4}$$

3.4.2. Channel pruning with BN weight factors

To optimize network compression and deployment, techniques such as parameter decomposition, model quantization, and model pruning are utilized [32,33]. Parameter decomposition involves breaking down a high-rank tensor into a low-rank tensor to simplify memory inversion and compress model size, but it doesn't significantly speed up networks [32]. Model quantization saves storage space by reducing the data storage set type (from FP32 to INT8), but typically leads to reductions in detection accuracy [32]. There are two main types of pruning: unstructured pruning and structured pruning [30]. Unstructured pruning can significantly reduce the number of model parameters, which make it difficult to deploy on edge devices. Despite structured pruning resulting in a lesser reduction of parameters compared to unstructured pruning, it is deemed highly compatible for deployment with edge devices. For the YOLO v5s model, each BN algorithm corresponds to a specific channel, and the weight factor on the BN algorithm is used as a factor to determine whether to prune that channel. As Algorithm 1 shown, it is the pseudocode of model pruning based on Batch Normalization.

Algorithm 1: model pruning based on Batch Normalization.

Input: model of YOLO v5s

```
1   for layer in model of YOLO v5s do
2   if m is Batch Normalization do
3   the weight of m is appended in list
```

(continued on next page)

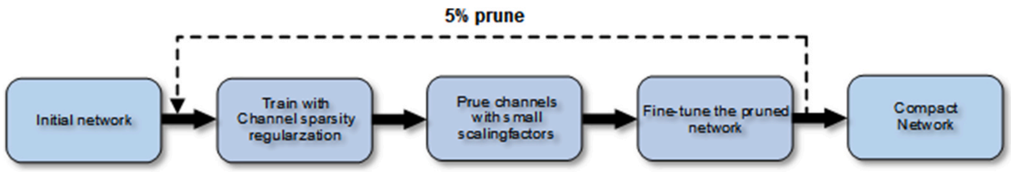
5% prune
Initial network
Train with Channel sparsity regularzation
Prue channels with small scalingfactors
Fine-tune the pruned network
Compact Network

Fig. 6. Model compression flow chart.

(*continued*)

```
Algorithm 1: model pruning based on Batch Normalization.
Input: model of YOLO v5s
4            end if
5            sort (list)
6            if the weight of the channel < sort (list) * pruned_radio do
7            prune the channels
8            end if
9            end for
```

Our channel pruning flowchart is shown in Fig. 7. The image on the left represents the unpruned network, while the right is the compact network. The blue channels are those with the smaller BN weight factors, while the yellow channels represent the channel with the larger BN weight factors. After sorting the channels according to the BN weight factors from smallest to largest, the channels with smaller weight values will be removed by a certain ratio, thus completing the channel pruning. As Algorithm 1 shown, it is Channel pruning with BN weight factors.

3.4.3. *Model fine-tuning*

After the channel pruning with BN weight factors, the accuracy of the model initially drops compared to the original model. To recover from this drop in accuracy, the model is fine-tuned, which improves the smoke detection accuracy. While the proposed model's accuracy is lower than the original model on our dataset, the improved model has significantly fewer parameters and faster smoke detection speed, making it more convenient for detecting smoke in grain bins.

3.5. *Model deployment*

One obstacle to implement artificial intelligence in agriculture is managing the considerable overhead in relation to the real-time functionality. Many large AI Companies have their servers stationed in the cloud and necessitate network transfer to function. To mitigate the network delays caused by the remote nature of the grain bins, the model for detecting early-stage fires is deployed in the grain bins with edge devices. Fig. 8 illustrates the 4 types of edge devices, (a) Raspberry Pi 3 B+, (b) Firefly, (c) Jetson Xavier NX, (d) Jetson Nano, used in our study for comparisons. In the following, Jetson Xavier NX is treated as example for introduction. The model was trained and fine-tuned, resulting in the generation of the PyTorch Model File (PTH) format. However, to further enhance the detection speed of our smoke model, we employed the Nvidia Tensorrt (TRT) format, which quantizes the original precision of FP32 to FP16 type, a unique feature of NVIDIA. This approach allowed for the accelerated deployment of edge computing, leading to improved detection speed.

4. Result

4.1. *Experimental environment*

The experimental preparation is established upon the Ubuntu 18.04 Linux operating system and harnesses the processing power of GPUs, with TITAN RTX serving as the Training equipment. In the course of the base and sparse training, the Adam optimization function is utilized, with a Batch Size of 32 across 100 epochs. For the purpose of fine-tuning the model, the number of training sessions is increased to 150 epochs.

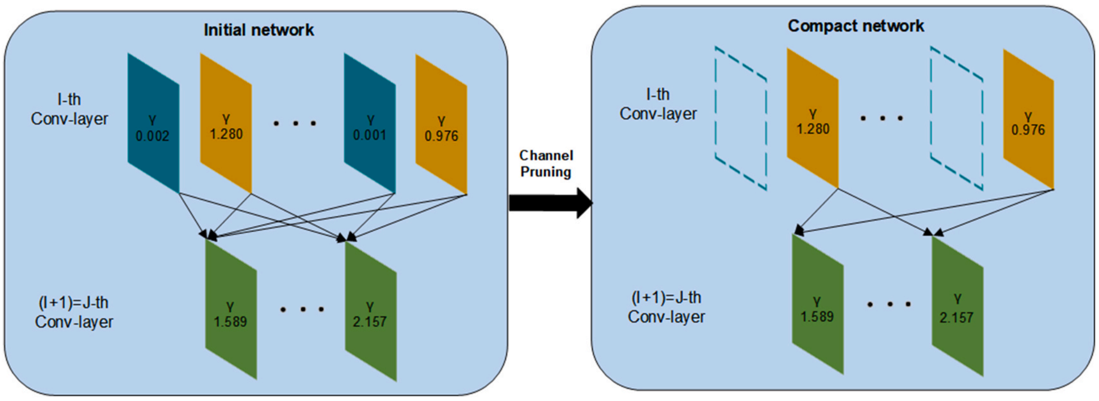

Fig. 7. Channel pruning based on BN diagram.

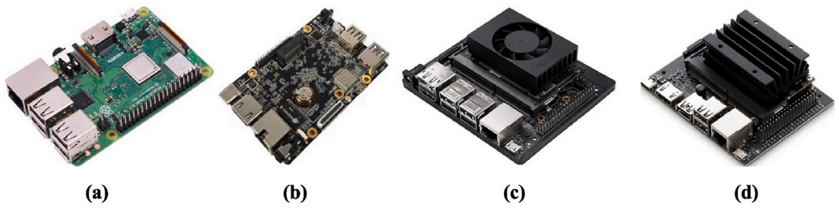

Fig. 8. Edge Computing devices: (a) Raspberry Pi 3 B+; (b) Firefly; (c) Jetson Xavier NX; (d) Jetson Nano.

4.2. Evaluation indicators

Our experimental results are evaluated using several metrics including the mAP, F1-score, model size, and detection speed, in order to assess the performance of the model and compare it with classical models. Precision shown in equations (5) refers to the ratio of correctly predicted positive samples to the total number of samples predicted to be positive. Recall shown in (6) is the ratio of correctly predicted positive samples to the total number of positive samples. The F1-score as shown in equation (7) is the harmonic mean of Precision and Recall. As shown in Equation (8), the average precision (AP) is calculated using integration to find the area enclosed by the PR curve and the coordinate axis, and is typically calculated for a single category. The mAP is the average of AP across 2 categories in our experiments, shown in (9).

$$Precision = \frac{1}{2}\sum_{i}^{2}\frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \tag{5}$$

$$\mathrm{Recall} = \frac{1}{2}\sum_{i}^{2}\frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i} \tag{6}$$

$$\mathrm{F1-Score} = \frac{1}{2}\sum_{i}^{2}\frac{2\times \mathrm{P}\times \mathrm{R}}{P+\mathrm{R}} \tag{7}$$

$$\mathrm{AP} = \int_0^1 P(r)dr \tag{8}$$

$$\mathrm{mAP}_{(0.5)} = \frac{\sum_{i}^{2} AP}{2} \tag{9}$$

In order to assess the efficacy of our novel smoke detection approach, we conducted a comparative analysis against several state-of-the-art target detection networks that utilize convolutional neural networks. Based on previous researches [20,21], the networks we compared include models of SSD, YOLO v4, YOLO v4tiny, YOLO v5n, and YOLO v5s.

4.3. Results of smoke detection

Table 2 presents a comparison of our proposed model with other target detection networks based on metrics such as mAP, F1 score, model size, and detection speed. The YOLO v5s model has the highest mAP and F1 in our dataset. Although the YOLO v5s model is larger and has a lower FPS than the YOLO v5n model, the mAP and F1 are 4.4% and 2.94% higher. Therefore, compression of the YOLO v5s model is proposed, resulting in a 5.11 M smaller model size without a significant reduction in mAP and F1, and a 15.59 FPS increase in the proposed model's detection speed compared to YOLO v5s and 7.85 FPS higher than YOLO v5n. The mAP and F1-Score of the proposed model are 94.9% and 92.85 respectively, keeping the accuracy almost unchanged compared to YOLO v5s.

In Fig. 9, a depiction of our proposed model alongside other models after being trained on our data is presented. SSD (figure (a)) and YOLO v4 (figure (b)) can not detect the picture of small smoke. Although figure (f) of Fig. 9 doesn't clearly show our improved model as

Table 2
Smoke detection of different models.

Models	mAP (%)	F1-score (%)	Model size (MB)	Detection speed (FPS)
SSD	88.66	87.37	91.64	109.31
Yolo v4	86.54	86.05	244.42	31.358
Yolo v4tiny	89.87	88.67	22.54	129.33
Yolo v5n	92.10	91.74	3.76	102.04
Yolo v5s	96.50	94.68	13.73	94.30
Proposed	94.90	92.85	8.62	109.89

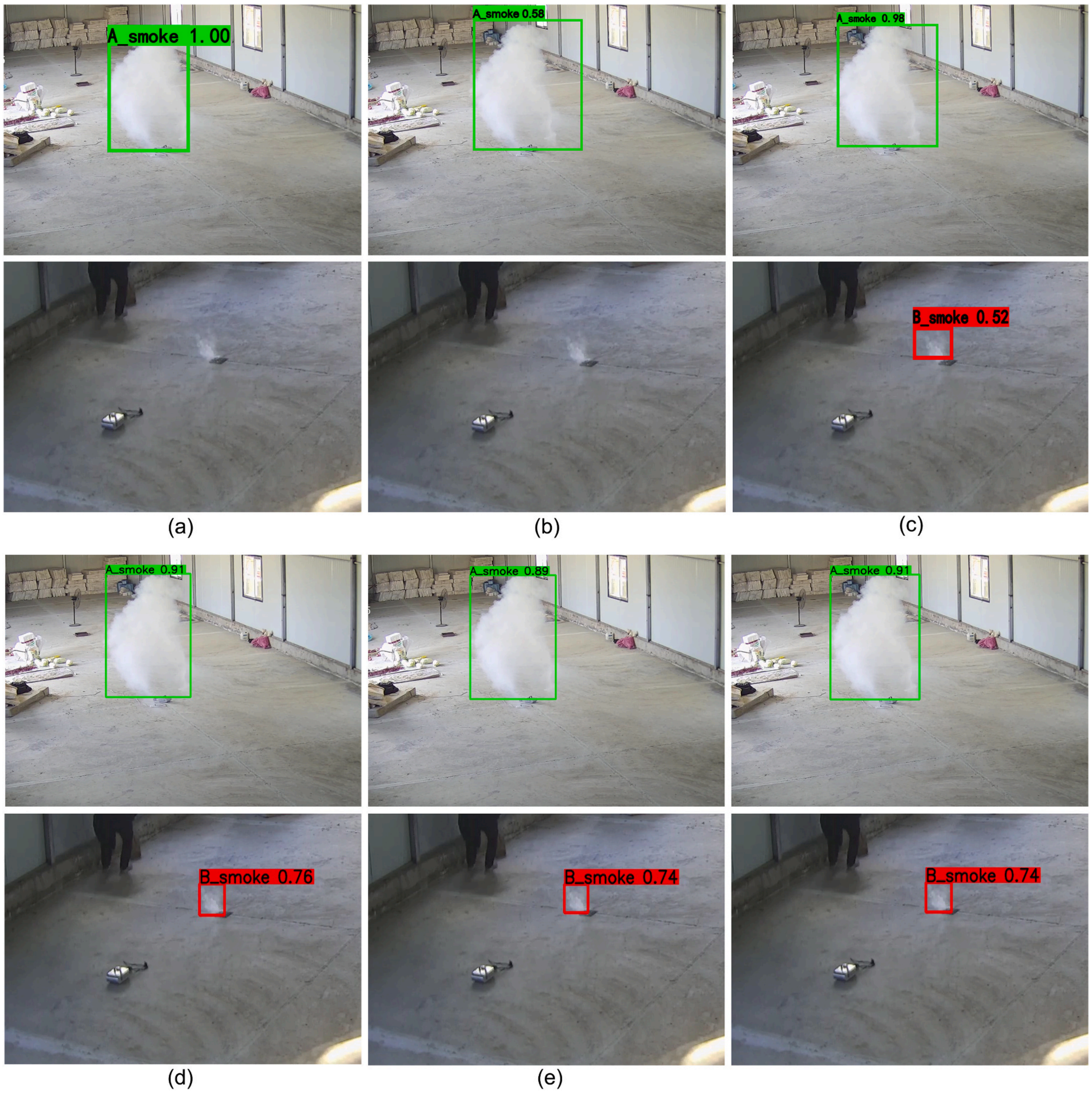

Fig. 9. Overall effect: smoke detection detected by (a) SSD;

better than YOLO v4 (figure (c)), YOLO v5n (figure (d)), and YOLO v5s (figure (e)), we have evaluated our model as performing better overall. The mAP, F1-score, model size and detection speed of our model are all better than YOLO v4. Our model exhibits superior mAP, F1-score, and detection speed in comparison to YOLO v5n. In addition, our model is 37.23% smaller and 16.53% faster than the initial YOLO v5s model, despite a reduction in accuracy. Overall, our improved model is useful and has practical applications.

4.4. Results of sparse training

Following our identification of model compression improvements for YOLO v5s, the model is trained on grain bins dataset using sparse L1-based regularization for BN weight factors. Fig. 10 presents a plot of the BN weight factors' change following training using six different sparse factors, with the horizontal axis representing the value of BN weight factors and the vertical axis the number of sparse training iterations. The sparsity factors choices of 10^{-5}, 10^{-4}, 10^{-3}, 3×10^{-3}, 2×10^{-3} and 2.5×10^{-3}, as referenced in Ref. [30], were considered. As shown in (a), (b), (c) and (d) the BN weight factors distribution is approximately Gaussian, which is not ideal for pruning filters. To counter this, we slightly increased the sparsity factor in line with the trend. In (e), a coefficient factor of 0.003 was used, resulting in all BN weights factors approaching zero, which lead to a decline in accuracy.

Consequently, we adjusted the sparsity factor size to 2.5×10^{-3}, as demonstrated in (f). The BN weight factors appear less Gaussian

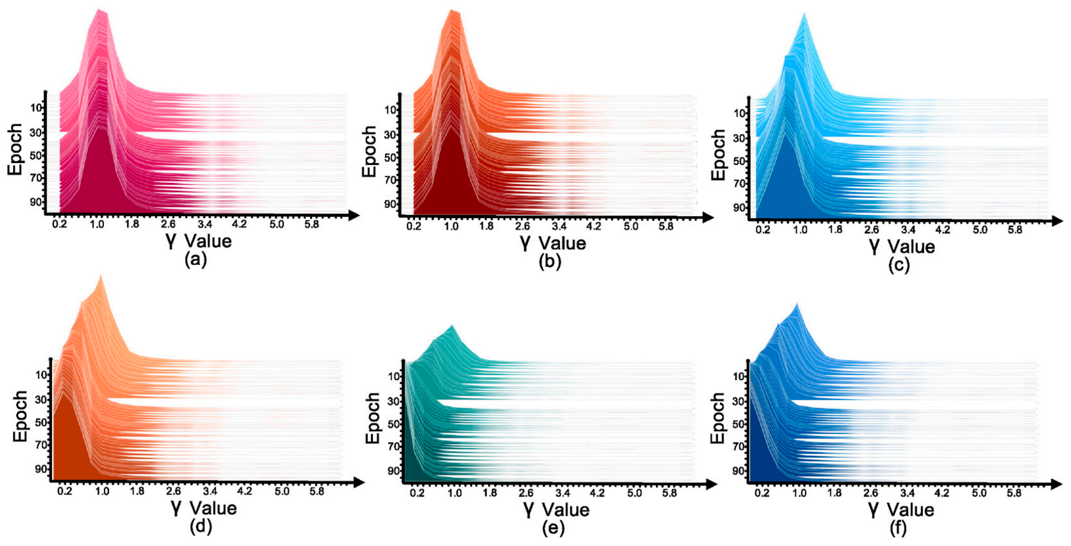

Fig. 10. γ under different sparse factors: (a) sparse factor 0.00001; (b) sparse factor 0.0001; (c) sparse 0.001; (d) sparse factor 0.002; (e) sparse factor 0.003; (f) sparse factor 0.0025.

and do not approach zero as the number of iterations increase.

4.5. Results of model pruning

After sparse training, we proceed with channel pruning and model compression based on the BN weights factor size. Different pruning rates were tested, and the most suitable pruning rate was selected during subsequent model fine-tuned. As depicted in Fig. 11, red color indicates total channels, and blue represents the remaining channels. It can be observed that in the early stages of the model, up to approximately 30 layers, only a small number of channels are pruned. As this initial stage of feature extraction plays a crucial role in subsequent detection [23], it is essential to preserve as much information as possible. However, around layer 40, there is a layer where only a few channels are left, and not all of them are completely pruned. This is because too many layers being pruned tend to cause a mismatch in subsequent training. Hence, as depicted in Fig. 11, the pruning rate is higher after 30 layers while ensuring that at least that number of layers is retained.

4.6. Results of fine-tuning

After model pruning improvements based on the BN weight factors, the accuracy of the model decreases. Therefore, we fine-tune our pruned model. Table 3 shown the results, these are the results of our experiments to fine-tune the model according to different pruning rates. After pruning with 30%, 20%, and 10% pruning rate, the model size is reduced by 5.1, 3.4, and 1.6 MB respectively. After 30% pruning, there is a relatively large reduction in the fine-tuned model results. Therefore, we chose the 30% pruned and fine-tuned model as our choice for deployment on edge devices.

4.7. Results of the test on the different edge devices

After fine-tuning, the model with the suitable channel pruning rate is obtained. To assess the compressed model more comprehensively, detection speed on diverse edge devices and parameters of various edge devices are compared [34], including Raspberry Pi 3 B+, Firefly, Jetson Nano and Jetson Xavier NX. As depicted in (Table 4), a comparative analysis of our model's detection format, the Jetson Xavier NX showcases the highest speed of detection, 49.26 FPS, for the self-created dataset. However, the speeds of Raspberry Pi 3 B+ and Firefly are only 0.54 FPS and 0.93 FPS respectively when detected without GPU. Particularly noteworthy is the advent of Firefly, a novel edge device that features with NPU that is similar GPU. It can speed up computing through the Rockchip Neural Network (RKNN) format and obtain the speed of 22.35 FPS.

Table 5 shows a comprehensive comparison of the edge devices used for detection speed analysis. The analysis includes chip cores, memory, language support, development feasibility, cost, and significant AI performance. With a staggering 6000 GFLOPs of AI computing power, Jetson Xavier NX outshines the other edge devices, which promises smoke detection in real time Hence, Jetson Xavier NX has been opted as the edge device for deploying the proposed model.

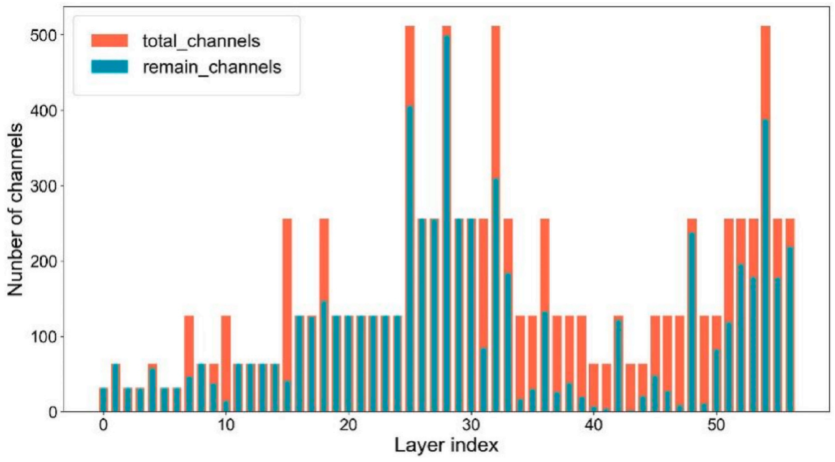

Fig. 11. Numbers of pruned channels for different layers: the red histogram illustrates total channels. And the blue histogram illustrates the remaining channel from pruning.

Table 3
Model performance after model fine-tuning.

Models	mAP (%)	F1-score (%)	Model size (MB)	Detection speed (FPS)
Yolo v5s	96.60	94.68	13.7	94.3
10%-prune	95.2	93.049	12	102.76
20%-prune	95.3	92.947	10.2	106.42
30%-prune (Proposed)	94.9	92.85	8.6	109.89
40%-prune	93.2	91.03	7.2	113.63

Table 4
Performance comparison on different edge devices.

Edge devices	Speed-PTH(FPS)	Speed-TRT (FPS)	Speed-RKNN(FPS)
Raspberry Pi 3B+	0.54	/	/
Firefly	0.93	/	22.35
Jetson Nano	9.55	17.48	/
Jetson Xavier NX	30.96	49.26	/

Table 5
Comparison of edge computing devices.

Brand	Chip	RAM	Language support	AI performance (GFLOPs)	GPIO	Size	Weight	Cost
Raspberry Pi 3B+	BCM2835 Processor	1 GB	Java, Python, C, C++	~21.4	40	85 × 54*17	46g	~$48
Firefly	RK3566	8G	Java, Pthon, C, C++	~800	30	93 × 60*13	55g	~ $129
Jetson Nano	1.4 Ghz Quad Core Cortex A57	4G	Java, Python, C, C++, CUDA	~500	40	100 × 80*30	60g	~ $110
Jetson Xavier NX	NVIDIA Carmel ARM v8.2 6	8G	Java, Python, C, C++, CUDA	~6000	40	102 × 90*30	76g	~ $459

5. Discussion

This paper proposes a model-compressed technique for promptly detecting smoke in grain bins with edge devices. Compared to other papers on computational agriculture, we are proud to introduce our pioneering efforts in generating a dataset dedicated to grain bin smoke detection. We have implemented a parameter compression strategy on the stable detection model, which led to a significantly reduced model size and greatly improved detection speed. Furthermore, we conducted experimental deployment on four edge devices, ultimately attaining the most optimal deployment device. However, the verification of our dataset's generalizability is a necessary step, and it has been observed that the accuracy of the compressed model is comparatively lower than that of the original model. During model comparison experiments, the latest detection model was not employed for comparison purposes, thus indicating the need for future endeavors towards conducting additional model comparison experiments.

In the experiments on model pruning, we decided not to perform channel pruning on the backbone network, as it is the initial feature extraction network and its features are critical for smoke detection. However, further experimentation is needed to determine whether this part of pruning has any effect on the final accuracy of the model. If the experimental process demands heightened cost control, deploying our model on edge devices like Firefly and Jetson Nano would serve as commendable choices. Additionally, future work will pay further attention to detection speed to better accommodate the computing constraints of edge devices in grain bins while keeping accuracy. We also aim to expand our detection capabilities beyond the complex scenarios of grain bins to other agricultural settings.

6. Conclusion

This paper proposes a real-time smoke detection model for grain bins based on YOLO v5s with edge devices, aimed at addressing the issues of smoke detection in grain bins and network latency. Compared to the original model, the proposed model reduces 5.11 MB in size while nearly maintaining the detection accuracy. The experimental results demonstrate the proposed model achieves mAP of 94.90% and an F1-score of 92.85%. Additionally, the model can detect smoke **a**t a server detection speed of 109.89 FPS and deployed detection speed of 49.26 FPS on the Jetson Xavier NX.

Author contribution statement

Hang Yin: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data. Mingxuan Chen: Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper. Yinqi Lin: Performed the experiments; Analyzed and interpreted the data; Wrote the paper. Shixuan Luo: Yalin Chen: Performed the experiments; Contributed reagents, materials, analysis tools or data. Song Yang: Analyzed and interpreted the data. Lijun Gao: Contributed reagents, materials, analysis tools or data.

Data availability statement

Data will be made available on request.

Declaration of interest's statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D.I. Patrícioa, R. Riederb, Computer vision and artificial intelligence in precision agriculture for grain crops: a systematic review, Comput. Electron. Agric. 153 (2018) 69–81.
[2] Q. Liu, Y. Qu, J. Liu, S. Wang, Effects of radio frequency heating on mortality of lesser grain borer, quality and storage stability of packaged milled rice, LWT 140 (2021).
[3] W.F. Wu, H.W. Cui, F. Han, et al., Zhang Q. Digital monitoring of grain conditions in large-scale bulk storage facilities based on spatiotemporal distributions of grain temperature, Biosyst. Eng. 210 (2021) 247–260.
[4] Nation Food, Strategic Reserves Administration, Standard for Grain Bins Construction, 2016. https://www.doc88.com/p-0784845086053.html. Accessed April 19, 2019.
[5] Y. Xu, W. Wu, X. Meng, et al., Code for layer and installation of multi-parameter storged-grain condition sensor network in grain bins (DB22/T 3114-2020). https://www.doc88.com/p-28339703219805.html, 2020. Accessed June 5, 2020.
[6] C. Jia, D. Sun, C. Cao, Mathematical Simulation of Temperature ®elds in a Stored Grain Bin Due to Internal Heat Generation, 2000. www.elsevier.com/locate/jfoodeng.
[7] J. Fonollosa, A. Solórzano, S. Marco, Chemical sensor systems and associated algorithms for fire detection: a review, Sensors 18 (2) (2018).
[8] S. Ye, Z. Bai, H. Chen, R. Bohush, S. Ablameko, An effective algorithm to detect both smoke and flame using color and wavelet analysis, Pattern Recogn. Image Anal. 27 (1) (2017) 131–138.
[9] B. Ko, Wildfire smoke detection using temporospatial features and random forest classifiers, Opt. Eng. 51 (1) (2012) 7208.
[10] A. Krizhevsky, I. Sutskever, G. Hinton, ImagesNet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2) (2012, December) (Nevada, NV).
[11] K. Simonyan, A. Zisserman, May). Very deep convolutional networks for large-scale image recognition, in: 2015 International Conference on Learning Representations, ICLR, San Diego, SD, 2015.
[12] M. Jiang, Y. Zhao, F. Yu, C. Zhou, T. Peng, A self-attention network for smoke detection, Fire Saf. J. (May) (2022) 129.
[13] H. Yin, M. Chen, W. Fan, Y. Jin, S. Hassan, S. Liu, Efficient smoke detection based on yolo v5s, Mathematics 10 (19) (2022).
[14] M. Guo, T. Xu, J. Liu, et al., Attention mechanisms in computer vision: a survey, Comp. Visual Media 8 (2022) 331–368.
[15] Y. Peng, Y. Wang, Real-time forest smoke detection using hand-designed features and deep learning, Comput. Electron. Agric. 167 (2019), 105029.

[16] J. Zhan, Y. Hu, G. Zhou, Y. Wang, W. Cai, L. Li, A high-precision forest fire smoke detection approach based on ARGNet, Comput. Electron. Agric. 196 (2022).
[17] Hu, Y., Zhan, J., Zhou, G., et al. (2022). Fast forest fire smoke detection using MVMNet. Knowl. Base Syst., 241.
[18] J. Zhan, Y. Hu, W. Cai, G. Zhou, et al., PDAM–STPNNet: a small target detection approach for wildland fire smoke through, Remote Sensing Images 13 (12) (2022) 2260.
[19] J. Almeida, C. Huang, F. Nogueira, S. Bhatia, A. De, Edge Firesmoke: a novel lightweight CNN model for real-time video fire-smoke detection, IEEE Trans. Ind. Inf. 18 (11) (2022) 7889–7898.
[20] Y. Zhang, J. Yu, Y. Chen, W. Yang, W. Zhang, Y. He, Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): an edge AI application, Comput. Electron. Agric. 192 (2022).
[21] Z. Jiao, K. Huang, G. Jia, H. Lei, Y. Cai, Z. Zhong, An effective litchi detection method based on edge devices in a complex scene, Biosyst. Eng. 222 (2022) 15–28.
[22] V. Mazzia, A. Khaliq, F. Salcetti, M. Chiaberge, Real-time apple detection system using embedded systems with hardware accelerators: an edge AI application, IEEE 2020 (2020).
[23] A. Bochkovskiy, C.Y. Wang, H. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020.
[24] Ding, H., Qu, L., Dou, B., et al, (2008). Special Fire Detectors (GB15631-2008). https://std.samr.gov.cn/gb/search/gbDetailed?id=71F772D7FFCED3A7E05397BE0A0AB82A. Accessed September 1, 2008.
[25] K. Tong, Y. Wu, Deep learning-based detection from the perspective of small or tiny objects: a survey, Image Vis Comput. 123 (2022).
[26] J. Redmon, A. Farhadi, Yolo9000: Better, Faster, Stronger, IEEE, 2017.
[27] J. Redmon, A. Farhadi, YOLOv3: an Incremental Improvement, 2018 (arXiv-preprint).
[28] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path Aggregation Network for Instance Segmentation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pub, Salt Lake City, UT, USA, 2018, December, pp. 8759–8768.
[29] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongle, Feature Pyramid networks for object detection. 2017, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pub, Honolulu, HI, USA, 2017, July, pp. 936–944.
[30] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming. 2017, in: IEEE International Conference on Computer Vision (ICCV), pub, Venice, Italy, 2017, December, pp. 2755–2763.
[31] S. Loffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: http://proceedings.mlr.press/v37/ioffe15.html, 2015.
[32] T. Liang, J. Glossner, L. Wang, S. Shi, X. Zhang, Pruning and quantization for deep neural network acceleration, A survey. Neurocomputing 461 (2021) 370–403.
[33] C. Yang, H. Liu, Channel pruning based on convolutional neural network sensitivity, Neurocomputing 507 (2022) 97–106.
[34] M. Nasir, K. Muhammad, A. Ullah, J. Ahmad, B. Wook, M. Sajjad, Enabling automation and edge intelligence over resource constraint IoT devices for smart home, Neurocomputing 491 (2022) 494–506.