IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 February 2023, accepted 1 March 2023, date of publication 3 March 2023, date of current version 13 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3252499

RESEARCH ARTICLE

An Accurate and Fast Animal Species Detection System for Embedded Devices

MAI IBRAHEAM, KIN FUN LI, (Senior Member, IEEE), AND FAYEZ GEBALI, (Life Senior Member, IEEE)
Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada
Corresponding author: Mai Ibraheam (maieelgendy@uvic.ca)

ABSTRACT Encounters between humans and wildlife often lead to injuries, especially in remote wilderness regions, and highways. Therefore, animal detection is a vital safety and wildlife conservation component that can mitigate the negative impacts of these encounters. Deep learning techniques have achieved the best results compared to other object detection techniques; however, they require many computations and parameters. A lightweight animal species detection model based on YOLOv2 was proposed. It was designed as a proof of concept of and as a first step to build a real-time mitigation system with embedded devices. Multi-level features merging is employed by adding a new pass-through layer to improve the feature extraction ability and accuracy of YOLOv2. Moreover, the two repeated 3×3 convolutional layers in the seventh block of the YOLOv2 architecture are removed to reduce computational complexity, and thus increase detection speed without reducing accuracy. Animal species detection methods based on regular Convolutional Neural Networks (CNNs) have been widely applied; however, these methods are difficult to adapt to geometric variations of animals in images. Thus, a modified YOLOv2 with the addition of deformable convolutional layers (DCLs) was proposed to resolve this issue. Our experimental results show that the proposed model outperforms the original YOLOv2 by 5.0% in accuracy and 12.0% in speed. Furthermore, our analysis shows that the modified YOLOv2 model is more suitable for deployment than YOLOv3 and YOLOv4 on embedded devices.

INDEX TERMS Convolutional neural networks, deformable convolutional layers, YOLO, embedded devices, animal detection.

I. INTRODUCTION

Wildlife-human and wildlife-vehicle encounters can cause serious problems for both humans and animals. These interactions have a wide range of effects, ranging from financial to physical to social. Wildlife-vehicle collision (WVC) and Wildlife-human conflict (WHC) have increased in North America over the last few decades [1], [2]. North American countries have large wildlife areas and many highways; hence, the probabilities of WVC are higher than in other areas in the world. Some of the reasons that led to WHC are: (1) the expansion of agriculture; (2) the expansion of urban areas towards wildlife habitats; (3) climate change, for example, the increase in heat waves and wildfires that pushes wildlife towards urban areas for survival [3]; (4) the COVID-19 pandemic, with the number of sightings of wildlife in urban areas increases and wildlife becomes urban dwellers, as human activities decrease due to lockdowns [4], [5].

The objective of this research was to alleviate the negative impacts of these conflicts on humans and wildlife by innovating and installing WVC and WHC mitigation systems on highways and trails, and in urban areas. The WVC and WHC mitigation consist of two subsystems: (i) an animal detection subsystem running on an embedded device [6] to detect moving animals using computer vision techniques on images from motion triggered camera [7], and (ii) a warning subsystem to warn drivers, hikers, and residents by activating visual or sound alerts once an animal is detected. Here, we focused on the animal detection subsystem with the additional goal of identifying the species.

The associate editor coordinating the review of this manuscript and approving it for publication was Bo Pu.

There are two major challenges in building an animal species detection system on embedded devices as a first step of the WVC and WHC mitigation systems: (i) enhancing detection accuracy while avoiding false alarms in unfavorable environments (snow, shade, bad illumination, etc.), and (ii) improving animal species detection speed so that a detection system can be deployed in near real time. The design of the animal species detection model architecture plays a vital role in improving detection accuracy. Any changes in the detection model's structure may lead to a different detection performance. Moreover, the trade-offs between detection accuracy and speed are crucial, especially when the animal species detection model is deployed on embedded device typically with low computation power.

A novel animal species detection system based on YOLOv2 [8] architecture was proposed to improve detection accuracy and speed on embedded devices by three modifications: (i) merge low-level and high-level features to distinguish the subtle features between animal species, (ii) remove two convolutional layers from the high block of YOLOv2 architecture to reduce computational complexity because there is only a small number of species classes, and (iii) add three deformable convolutional layers (DCLs) [9] to the last block of the YOLOv2 backbone network to enhance the extracted features under various geometric deformation of the animals. The proposed model was used to identify and localize six animal species. These animals were selected because their encounters with vehicles or humans usually result in severe injuries. As a proof of concept and as a first step to build an accurate real-time WVC and WHC mitigation systems, our proposed model was designed to be embedded devices.

The contributions of this work are summarized as:

- Accurate: New effective ideas were proposed by adding DCLs and adopting the strategy of multi-level features merging into the YOLO structure, to enhance the extracted features and improve the detection accuracy.
- Fast: The repeated 3×3 convolutional layers from the high layer of YOLO, which causes heavy complex computation and memory requirements were removed, enabling our proposed model to detect animal species in near real time on embedded device while maintaining high detection accuracy.

The rest of the paper is organized as follows. Section II summarizes the related work. Section III describes the dataset. Section IV presents the study's methodology. Section V presents the proposed animal species detector model. Section VI presents the model implementation and its evaluation. Section VII compares and discusses the results of animal species detection using YOLOv2, modified YOLOv2, YOLOv3, and YOLOv4, in terms of accuracy and detection speed as deployed on embedded platforms. Finally, Section VIII concludes the study and discusses potential extensions for future work.

II. RELATED WORK

A. OBJECT DETECTION

In classic object detection, windows of different sizes are moved over the image to obtain the region proposals, and the features of each proposal are then extracted using feature descriptors such as Haar [10], Histogram of Oriented Gradients (HOG) [11], and Scale Invariant Feature Transform (SIFT) [12]. Subsequently, these features are fed into a classifier, such as Support Vector Machine (SVM) [13]. It is difficult to achieve high accuracy in classic object detection when there are diverse images in the dataset, such as those in ImageNet [14] and COCO [15]. Therefore, Deep Neural Networks (DNNs) have been used after much improvement in artificial neural networks for the extraction of significant features to achieve high accuracy [16], [17].

DNN object detection algorithms based on region proposals were introduced in the Pascal Visual Object Classes (VOC) challenge [18]. Girshick et al. [19] merged region proposals using CNNs and their proposed Region-based CNN (R-CNN) detector. As a result of the success of the region proposal method, Fast R-CNN [20] was proposed to reduce the computational complexity of the R-CNN. Ren et al. [21] combined the Region Proposal Network (RPN) and Fast R-CNN into one network ''Faster R-CNN'' to achieve further speed-up and higher object detection accuracy. Mask R-CNN [22] was an extension of Faster R-CNN that computes object masks in parallel with the bounding box. The two-stage detectors (R-CNN based detectors) are slow because: (i) they have a complex pipeline, and (ii) each stage must be trained separately. In 2016, Redmon and Farhadi [23] solved this issue by dealing with object detection as a regression problem directly from the image to bounding box coordinates and class probabilities. They proposed You Only Look Once (YOLO) in an image as a one-stage detector to identify and localize objects. YOLO is three times or more faster than a two-stage detector, but less accurate than the Mask R-CNN because of the static nature of anchor boxes [24]. Later, Redmon et al. [8] proposed a new version (YOLOv2) model using a new network called DarkNet-19 to improve the accuracy and speed of object detection. However, YOLOv2 has a problem in detecting small objects, as discussed later in Section VII. In 2018, YOLOv3 was the last version of the YOLO model proposed by Redmon and Farhadi [25]. Darknet-53 was used as a backbone network to extract features in YOLOv3, instead of DarkNet-19 [26] as in YOLOv2, to improve the detection accuracy of small objects. This makes YOLOv3 slower than YOLOv2 because the computational complexity and the number of model parameters increase. Moreover, YOLOv3's ability to detect small objects is still limited because it does not utilize low-level features. All these restrictions limited the use of YOLOv3 in industrial applications [27], [28]. YOLOv4 was proposed by Bochkovskiy et al. [29] to improve accuracy and detection speed. More work, such as [30], has been published on how to modify the architecture of YOLOv4 to be deployed on real-time applications with high performance.

YOLO object detection models have complex architectures. They require a large number of parameters or weights to make predictions, which are learned from the training dataset. Therefore, these models require a powerful platform, such as a Graphic Processing Unit (GPU) [31] and a large memory, to be efficient and effective in real-time applications. However, most real-time applications utilizing embedded GPU or Central Processing Unit (CPU) devices are constrained by limited storage space and low computational power. To overcome this, many studies proposed lightweight YOLO object detection models that are more suitable for less powerful platforms. These include Fast YOLO [32], YOLOv3-Tiny [33], and YOLOv4-Tiny [34]. These models have simpler architectures with fewer parameters than regular YOLO object detection models. Although the detection speed of these lightweight models has improved with limited hardware platforms, the detection accuracy has decreased [35], [36]. Therefore, accurate and near real time object detection on embedded devices remains a challenge.

B. ANIMAL SPECIES DETECTION

Recent studies have attempted to identify animal species in camera-trap images [37], [38], [39]. However, only a few studies have focused on animal species detection to identify and localize them, [40], [41]. Some of them are specialized to detect a single animal species, such as cattle [41], zebras [40], and pigs [42].

Parham and Stewart [40] used YOLO to detect zebras from a dataset of 2,500 images and create bounding boxes of Plains Zebras with an accuracy of 55.6% and Grevy's Zebras with an accuracy of 56.6%. Zhang et al. [43] created a dataset of 23 different species in both daytime color and nighttime grayscale formats using 800 camera traps. They compared Fast R-CNN and Faster R-CNN with their proposed method, the spatiotemporal object proposal and patch verification framework, which achieved an average F-measure of 82.1% in animal species detection. Beibei et al. [41] evaluated the Mask R-CNN model for the detection and counting of cattle (single class) from quadcopter imagery. The authors achieved an accuracy of 94%. Gupta et al. [44] used the Mask R-CNN model with a pre-trained network, ResNet-101, to detect two animal species (cows and dogs). They achieved an average precision of 79.47% and 81.09%, for detecting cows and dogs, respectively. Schneider et al. [24] compared the accuracy of Faster R-CNN and YOLOv2 models in detecting animal species within camera-trap images. Their results showed that Faster R-CNN outperformed YOLOv2 by 33.4%.

C. ANIMAL DETECTION SYSTEMS

Current research on animal detection systems can be divided into two directions: using traditional machine learning (ML) algorithms, which rely on feature extraction descriptors to detect animals [45], and using deep learning (DL) algorithms, which rely on Convolutional Neural Networks (CNNs) [46].

Parikh et al. [47] proposed an animal detection system with audio and visual alarm signals to help reduce WVC using a template matching algorithm [48]. The authors stated that their system produces many false predictions under various lighting conditions, particularly at night. Mammeri et al. [49] proposed a moose detection system, which uses a roadside camera, to warn drivers. A two-stage strategy was used: first, the Local Binary Pattern Adaptive boost (LBP-Adaboost) algorithm [50] was applied to generate regions of interest (RoIs), and second, each generated RoI was processed by an adapted version of the HOG-SVM detector. Their system achieved an accuracy of more than 83% for moose detection with an inference time of 52.8 ms. Sharma and Shah [51] used the HOG descriptor and boosted cascade classifier in an animal detection system on highways to alert drivers. They focused only on cows and dogs. The accuracy of their proposed system was almost 82.5%, with an average detection time of 100 ms. Matuska et al. [52] presented an animal classification system to monitor animal migration. Their system can identify five animal species: brown bear, deer, fox, wolf, and wild boar with an accuracy of 94%, using a combination of Speeded-Up Robust Features (SURF), SIFT, and FlannBased (FB) as feature descriptors, and SVM as a classifier. Antonio et al. [53] presented an animal detection system to detect the presence of animals on roads and to warn drivers. Synthetic animal images were used to train two supervised ML algorithms: K-Nearest Neighbors (KNN) and Random Forest (RF) [54]. The KNN model outperformed RF in identifying animals and was implemented and tested on a Raspberry Pi 3 B+ [55].

In several applications, DL algorithms outperform traditional ML algorithms, particularly in domains with large and diverse training datasets [56], [57]. Saxena et al. [58] assembled a dataset of 31,774 images for various animals and proposed an animal vehicle collision avoidance system. They evaluated the performance of these two-stage object detectors for animals: single shot detector (SSD) [59] and Faster R-CNN. The Faster R-CNN model achieved a mean average precision (mAP) of 82.11% with a detection speed of 90 ms, whereas the SSD model achieved a mAP of 80.5% with a detection speed of 10 ms.

Adami et al. [60] developed a smart agriculture application to protect crops from animals by repelling them through generated ultrasounds. They deployed and evaluated the performance of YOLOv3 and YOLOv3-tiny (a light version of YOLOv3) on different edge computing devices to detect deer and wild boar. The authors achieved an 82.5% mAP by YOLOv3 and a 62.4% mAP by YOLOv3-tiny. Sato et al. [61] presented an animal detection system to prevent accidents involving animals on highways. They used YOLOv4 and YOLOv4-tiny (a light version of YOLOv4) and compared their performance in detecting horses and donkeys. These models were trained on 2,000 downloaded images from Google and tested on 147 images. The accuracy was 84.87% with a detection time of 0.035 ms and 79.87% with a detection time of 0.025 ms, using YOLOv4 and YOLOv4-tiny,

TABLE 1. Percentage of the animal species in our dataset.

Animal Species	Percentage
Bear	16%
Cougar	13%
Deer	16%
Elk	15%
Moose	14%
Mountain Goat	13%
No Animal	13%

respectively. A desktop computer with GPU GTX 1050Ti was used to evaluate the model's speed. Similar to our work, the authors deployed their proposed animal detection system on an embedded device, as a proof of the capability of the system and as an initial step to build highway detection systems.

All the above-mentioned approaches have limitations. Some of them use two-stage detectors which are not suitable for real-time applications, as they are slow. Others can only detect one or two animal species. Moreover, all these models' detection accuracy and speed need improvement.

III. OUR DATASET

The dataset used in this study was provided by the British Columbia Ministry of Transportation and Infrastructure (BCMoTI). This dataset has 138,482 unlabeled images from various locations and angles of six Canadian animal species: bear, cougar, deer, elk, moose, and mountain goat, and 20,114 images without animals. These images have resolutions ranging from 512 × 384 to 2,048 × 1,536 pixels, which were captured using the Reconyx camera trap [62]. Fig.1 shows example images of the animal species in the dataset. Due to the hazy and snowy night images of cougars, as shown in Fig. 1 (f), a video of cougar was downloaded from YouTube, transferred to a sequence of images, and used for training to improve the identification performance. Table 1 shows the percentage of each animal species and images with no animals in the dataset.

IV. METHODOLOGY

The dataset with Canadian animal species is the focus of this paper. The challenge is to find a fast, accurate, and lightweight animal detector that can operate in real-time and be implemented on portable embedded platforms with limited resources. In our work, a Raspberry Pi 4 Model B (RP4B) [55] was chosen as the embedded platform for the following reasons: (1) low price, (2) low power consumption, (3) wide availability from many international suppliers, (4) compatibility with many operating systems and open-source software, (5) big community of users, and (6) strong technical support.

In this section, the YOLOv2 model which forms the basis of our animal species detector is presented. Then, the Deformable Convolutional Neural Network (D-CNN) is introduced.

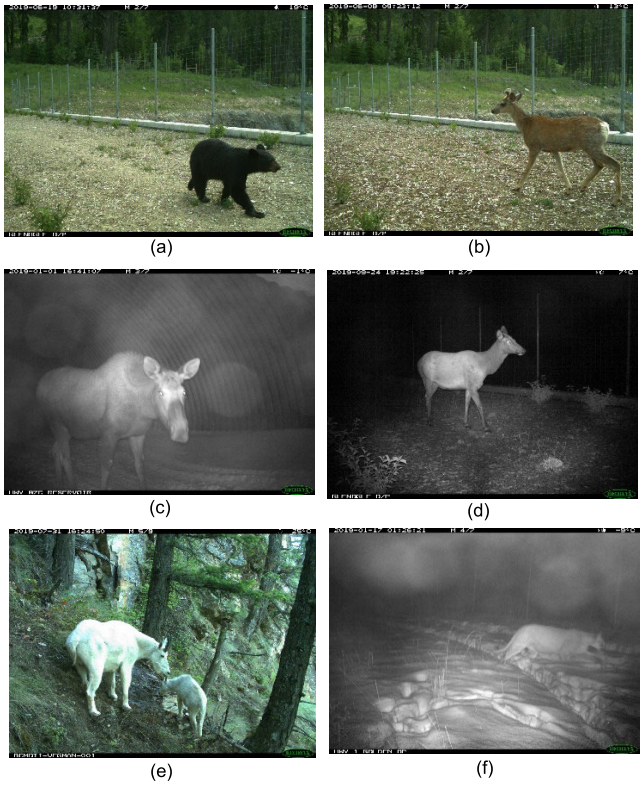

FIGURE 1. Example of images from our dataset. (a) Bear, (b) deer, (c) moose, (d) elk, (e) mountain goat, and (f) cougar.

A. THE YOLO DETECTION ALGORITHM

The main steps of YOLO are shown in Fig. 2. The input image in Fig. 2 (a) is split into S × S grid cells, as shown in Fig. 2 (b), where S is an integer. As shown in Fig. 2 (c), each cell predicts: (i) a fixed number of bounding boxes with different aspect ratios and scales (orange boxes) to cover a wide range of object shapes and sizes, each bounding box has four coordinates and one confidence score which measures the probability of an object in the bounding box, and (ii) object class probabilities. These predictions are used to identify and localize the objects in the image after discarding all the bounding boxes with low confidence scores as shown in Fig. 2 (d), using the Non-Maximum Suppression (NMS) algorithm [23].

YOLOv2 architecture was selected in this work to be modified mainly due to: (i) the detection capability of YOLOv2 has been proven when there is a variety of classes with big differences [8], and (ii) the limited computational resources, YOLOv2 is still one of the most used detectors in many applications in industry [63], [64], [65]. Therefore, a modified YOLOv2 model is proposed in this work. The superiority of adding a pass-through layer, adding three DCLs, and removing two high convolutional layers in the YOLOv2 architecture is proven through comparative analysis between YOLOv2, modified YOLOv2, YOLOv3, and YOLOv4 models, as shown in Section VII.

A CNN architecture consists of: (i) a deep, fully convolutional network to generate feature maps for the input image,

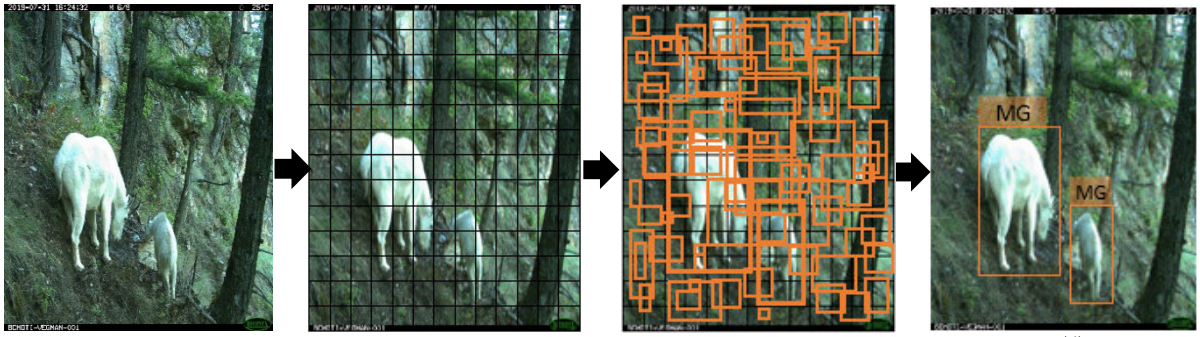

FIGURE 2. The framework of YOLO. (a) Input image of two animals (Mountain Goat: MG). (b) Divide up input image with S × S grid cells. (c) Each cell predicts bounding boxes and confidence probabilities: P(object). If there is an object in the grid cell then P(object) =1, otherwise P(object) =0. Also, each cell predicts class conditional probability: P(class: MG/object)). (d) Output image with the detected animal species (MG).

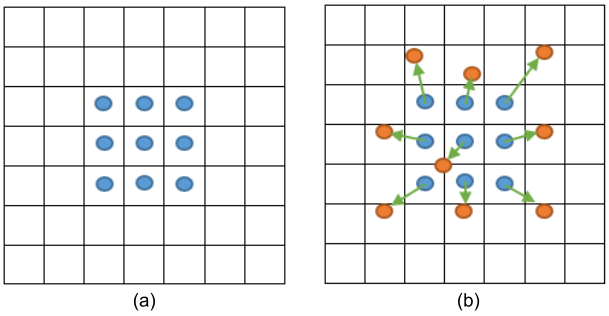

FIGURE 3. Illustration of the sampling locations in the 3 × 3 regular and deformable sampling matrices. (a) A regular sampling matrix (blue points). (b) A deformable sampling matrix (orange points) with learned augmented offsets (green arrows) to redistribute the sampling locations according to the object's scale and shape.

and (ii) a shallow network for a specific task to generate results from the feature maps. DarkNet-19 was used as a backbone network for YOLOv2 for feature extraction [8]. It consists of 19 successive convolutional layers with kernel sizes of 3 × 3, 1 × 1 and five 2 × 2 max pooling layers with stride = 2 for feature extraction on top of the classification layer.

For the original YOLOv2, the last block of DarkNet-19 has been replaced by two blocks with three 3 × 3 convolutional layers and one 1 × 1 convolutional layer for detection. In addition, YOLOv2 has a pass-through layer named Reorganisation layer (Reorg. layer), which is used to reduce the size of mid-level features to match the size of high-level features by using a down-sampling factor equal to 2. Thereby, these two feature maps can be combined to improve network performance.

B. D-CNN

In this work, a D-CNN was integrated into the YOLOv2 model to enhance its accuracy in detecting animal species to deal with the drawback of CNN, as discussed in subsection V-C. A CNN has: (i) a convolution layer which samples the input feature map at fixed locations due to the fixed square kernel; and therefore, the receptive field does not cover the entire pixel of the object properly; (ii) a pooling layer which decreases spatial resolution with a fixed ratio; and (iii) a RoI pooling layer which produces fixed size feature maps. As a result, CNN has a fixed geometric structure that cannot deal with any geometric variations in the object scale, pose, viewpoint, and part deformation [9], [66].

To overcome the drawbacks of CNN by using D-CNN, the regular fixed sampling matrix (convolution kernel) in Fig. 3 (a) was replaced with the deformable, movable sampling matrix, as shown in Fig. 3 (b), where each point in the regular sampling matrix was moved by a learnable offset, depending on the shape of the object. A deformable matrix structure can be obtained by a convolution algorithm that calculates the offset of the sampling position to learn the geometrical properties of the objects to be detected [9].

As shown in Fig. 4, a D-CNN consists of: (i) a regular convolution layer with a regular 3 × 3 sampling matrix as in Fig. 4 (a), to generate a feature map for the whole input image in Fig. 4 (b), and (ii) an additional convolution layer with a 3 × 3 regular sampling matrix as in Fig. 4 (c) for the learned augmented offsets in Fig. 4 (d) to be learned from the feature map. These offsets can be trained easily by using back propagation from end-to-end to generate a deformable 3 × 3 sampling matrix as in Fig. 4 (e). Fig. 4 (f) shows the output feature map after applying the regular sampling matrix. The positions of the sampling points in the sampling matrix (orange circles) are in a fixed 3 × 3 square shape. Fig. 4 (g) shows the output feature map after applying the deformable sampling matrix. The positions of the sampling points are changed from the original 3 × 3 square shape to another shape according to the object scale and shape, to generate a more accurate bounding box. By comparing the two outputs in Fig. 4 (f) and Fig. 4 (g), we found that the deformable convolutional layer increases the detection accuracy of the network at the cost of a small amount of computation and parameters for the offset learning, as shown in subsection V-C.

The advantages of D-CNN compared with the regular CNN can be summarized as:

- D-CNN can be integrated into any CNN architectures, giving them the ability to deform its sampling matrix to fit the structure of the object.

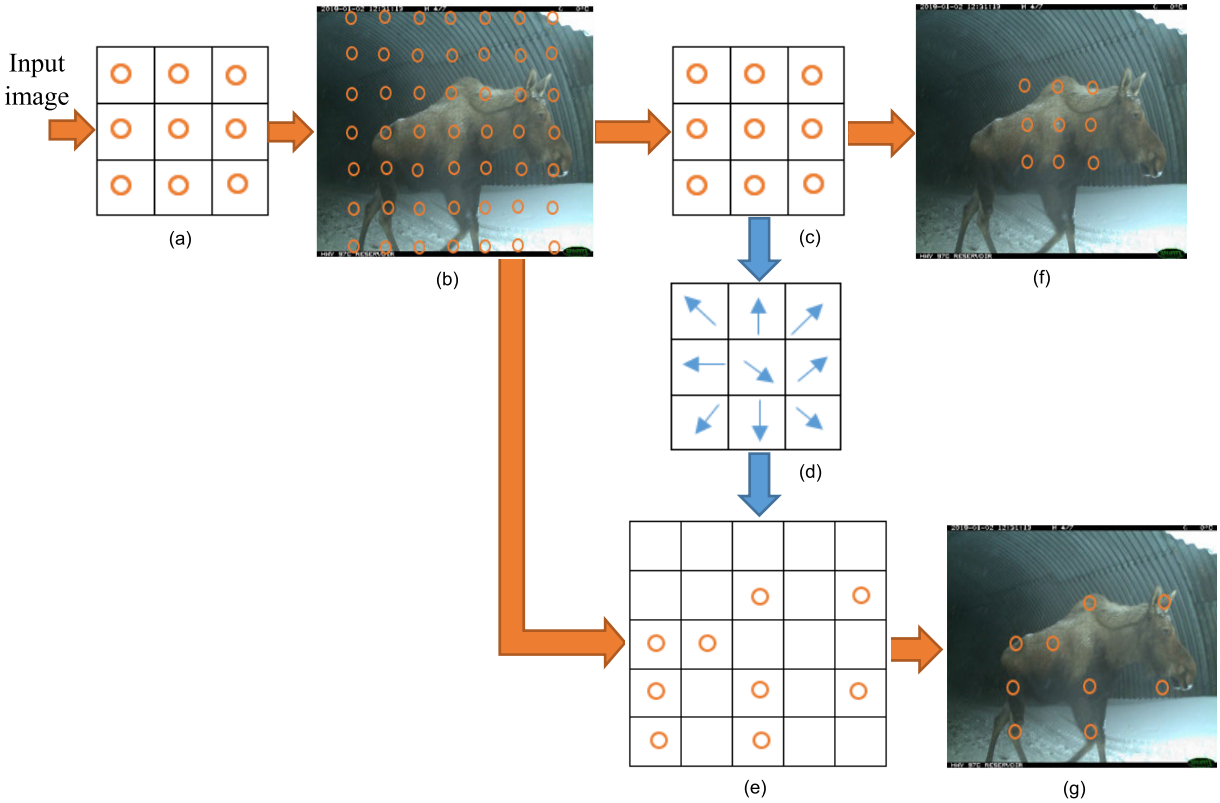

FIGURE 4. Illustration of the difference between applying 3 × 3 regular and deformable sampling matrices on input feature map. (a) Regular sampling matrix (orange circles) of regular convolution layer. (b) Feature map of the whole image after convolution layer. (c) Regular sampling matrix. (d) Learned augmented offsets (blue arrows) from the preceding feature map via an additional convolutional layer to redistribute the sampling locations of the regular sampling matrix to focus on the object. (e) Deformable sampling matrix after adding offsets to the regular sampling matrix. (f) Output feature map after applying regular convolution. (g) Output feature map after applying deformable convolution.

- The offsets in D-CNN are dynamic model outputs which vary according to the object locations in the image.
- D-CNN is capable of learning receptive fields adaptively.
- D-CNN learns sampling locations instead of filter weights to generate more accurate bounding box.
- D-CNN provides improvements in object detection accuracy [66].

V. PROPOSED ANIMAL SPECIES DETECTOR MODEL

The YOLOv2 model has shown impressive results in many generic object detection applications which have different degrees of variation in object classes, such as humans, animals, traffic signs, and vehicles [67]. This motivated us to propose a lightweight animal species detector based on YOLOv2 to detect animal species as shown in Fig. 5. However, there are a lot of challenges in this work to improve detection accuracy and speed.

A. MULTI-LEVEL FEATURES MERGING

The differences between the six animal species are related to edges, intensity, contour, texture, color, etc., which exist in low-level features. To improve animal species identification and localization accuracy of the original YOLOv2, the low-level features are merged with high-level features by adding a new pass-through layer (Reorg. layer) with a down-sampling factor of 4; as shown by the black dashed line in Fig. 5.

As shown in Table 2, after the low- and high-level features merging, the capability of YOLOv2 to recognize the small differences between animal species increased by 1.8%. The biggest improvement was achieved with the deer images as most of these images were captured during daylight as shown in Fig. 1 (b), and the smallest improvement was with the moose images, as most of them were captured at night as in Fig. 1 (c).

B. CONVOLUTIONAL LAYERS REMOVAL

The architecture of YOLOv2 was designed for generic object detection applications. Generic objects imply that the number of classes is large. Thereby, there were five repeated 3 × 3 convolutional layers with 1024 filters in the seventh block of the YOLOv2 architecture. However, only six species of animals need to be detected in this study. Fig. 6 shows the loss curve (sum of errors for each iteration) of detecting animal species in the validation dataset with and without two 3 × 3 convolutional layers in YOLOv2. The two models, with and without two convolutional layers, were trained with the same number of epochs, and both reached the same accuracy. Thus, removing the two 3 × 3 convolutional layers from the seventh block of YOLOv2 does not reduce the accuracy of

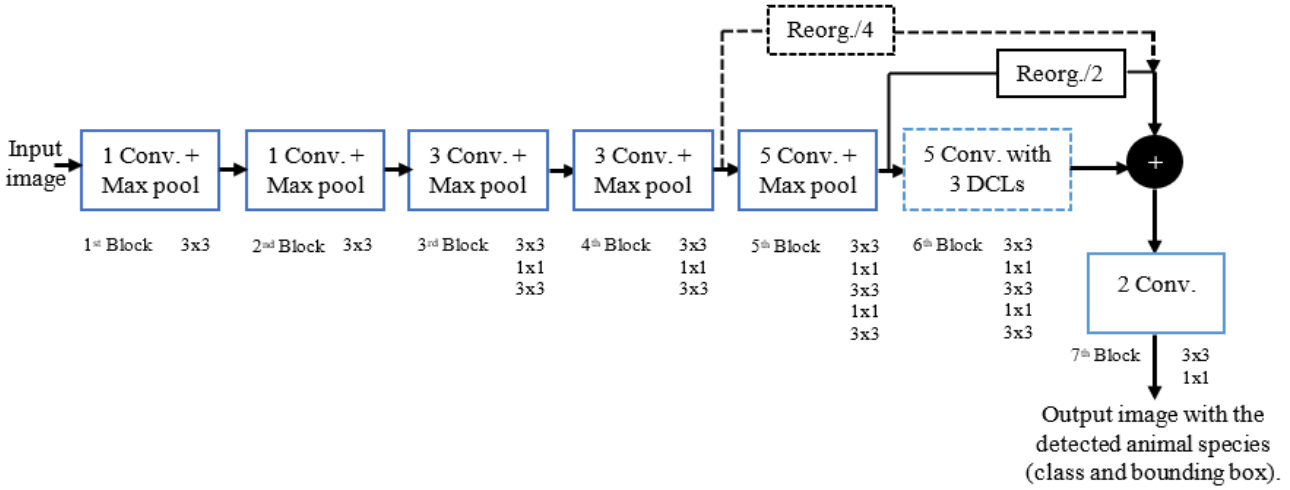

FIGURE 5. Architecture of the proposed YOLOv2 animal species detector. The detector has a new pass-through layer with a down-sampling factor of 4 to merge low- with high-level features (the black dashed line). The sixth block of detector (blue dashed block) has five convolution layers with three added DCLs.

TABLE 2. A comparison between YOLOv2 and modified YOLOv2 with multi-level features merging to detect six animal species in terms of Average Precision (AP) and mAP (details in VI.B.) on the BCMoTI dataset.

Animal Species	YOLOv2 (AP)	YOLOv2 (AP) with multi-level features merging
Bear	0.860	0.877
Cougar	0.750	0.764
Deer	0.900	0.920
Elk	0.823	0.839
Moose	0.791	0.799
Mountain Goat	0.856	0.872
mAP	0.830	0.845

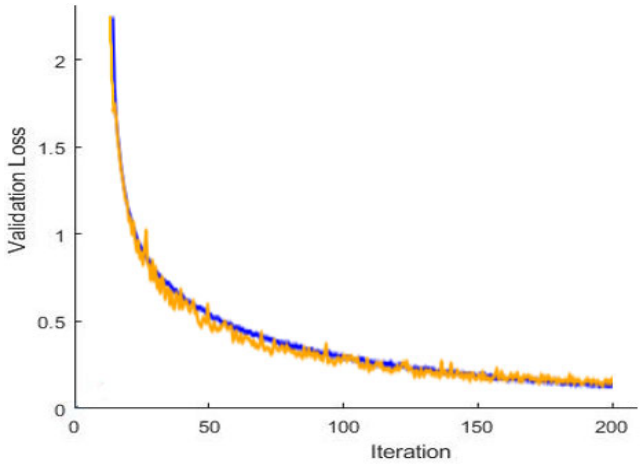

FIGURE 6. The loss curve of the validation dataset with (blue line) and without (orange line) two 3 × 3 convolutional layers in the seventh block of YOLOv2 architecture.

detecting the six selected animal species, however it speeds up the detection process. It can be concluded that, YOLOv2 without the two 3 × 3 convolutional layers, has sufficient features to distinguish between the six animal species.

C. ADDING DCLs

For object classification, regular CNNs extract features from images by using a fixed kernel. If any changes occur in the shape, size, and posture of objects in the image due to the object's motion, or the location of the cameras, the neural network may have difficulties in cannot classifying these objects correctly. Therefore, in this study, the idea of adding DCLs to the YOLOv2 model was proposed to learn the geometric transformation of animals. It can produce an adaptive deformable kernel and offsets according to the scale and shape of the animal, by augmenting the spatial sampling locations in the convolution layers. This is used as our solution instead of augmenting the dataset in advance, which increases the cost of data pre-processing, and the augmented data would never cover all the images in real applications [9]. As shown by the blue dashed box, in Fig. 5, three DCLs are used to learn the offsets, which are added to the regular 3 × 3 grid sampling locations in the sixth block. The detection capability and accuracy were enhanced as reported in Table 3.

Table 3 shows the average precision (AP) [68], mAP, and inference-time, which are elaborated in VI.B. The deformable YOLOv2 reaches higher accuracy in detecting the six animal species as compared to the regular YOLOv2, as expected. In the BCMoTI dataset, the deformable YOLOv2 provides a better result for the six animal species with a mAP of 85.5% in about 0.027 second per image on a Core i7 system, which equates to 37 images in one second. On RP4B, the inference-time of deformable YOLOv2 is 0.15 second per image, whereas for YOLOv2, it is 0.13 second per image. This means that the addition of DCLs slows down the detection speed on RP4B by 15.3%.

To summarize the above discussion, the proposed YOLOv2 animal species detector, as shown in Fig. 5, differs from the original YOLOv2 by: (i) adding a pass-through layer with a down-sampling factor equal to 4 (black dashed line),

TABLE 3. Evaluation of animal species detection by using regular and deformable YOLOv2 in AP, mAP, and Inference-time on the BCMoTI dataset.

Animal Species	YOLOv2 (AP)	Deformable YOLOv2 (AP)
Bear	0.860	0.893
Cougar	0.750	0.764
Deer	0.900	0.911
Elk	0.823	0.851
Moose	0.791	0.836
Mountain Goat	0.856	0.875
mAP	0.830	0.855
Inference-time (second) on Core i7 system	0.025	0.027
Inference-time (second) on RP4B	0.13	0.15

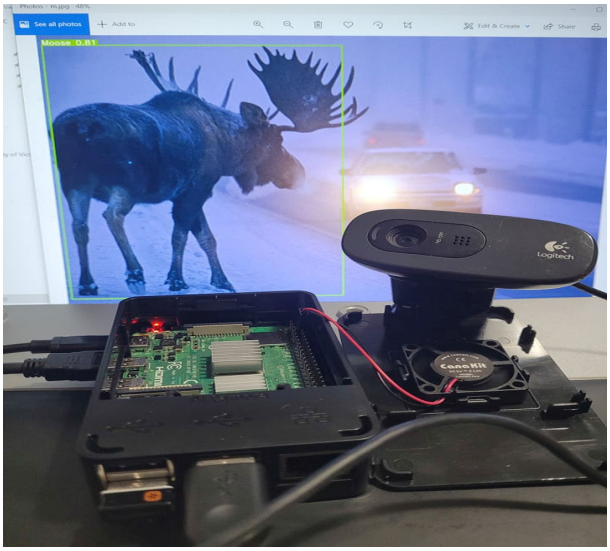

FIGURE 7. A mock-up using RP4B and a web camera to illustrate the capability of the proposed animal detection model.

which increases the mAP of detecting six animal species by 1.8%,(ii) removing two repeated 3 × 3 convolution layers from the seventh block of YOLOv2, but maintains the accuracy of detecting the six animal species, (iii) adding three DCLs to the sixth block (blue dashed box) to deal with the geometric transformation of animals in images, which increases the animal species detection mAP by 3.0%.

VI. IMPLEMENTATION AND EVALUATION METRICS

A. IMPLEMENTATION

1) RoI LABELLING

Image annotation was done using a MATLAB 2022a application called Image Labeler [69]. The input of this application is a set of images, which are then annotated by rectangular bounding boxes with class labels over the objects. These annotated images are exported and converted from MATLAB format (matlab file) to YOLO format (image and corresponding text file), to be compatible for training in Python.

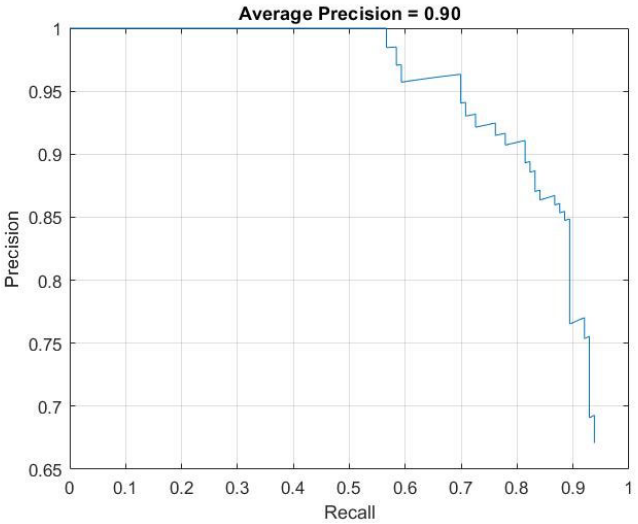

FIGURE 8. Precision-recall curve of detecting deer using the original YOLOv2 model.

2) TRAINING

The dataset was divided into 70% training, 15% validation, and 15% testing. The validation dataset is important for adjusting and finding the significant values of the hyper-parameters through trial-and-error [70]. These hyper-parameters have a vital effect on the performance of the YOLOv2, modified YOLOv2, YOLOv3, and YOLOv4 models. In this work, all four models used the same hyper-parameters. The Adaptive Moment Estimation (Adam) optimizer [71] was used with an empirically determined momentum of 0.9 and a weight decay of 0.0005, to accelerate the convergence and improve the detector performance. The models were also trained and fine-tuned by using a learning rate of 0.001.

The training was performed after resizing the input images to 416 × 416 pixels by a pre-trained ImageNet DarkNet network [26] to extract the objects' features from the input images. All the models were initialized with the ImageNet weights [72] and trained with the same number of epochs.

3) EXPERIMENTAL SETUP

The training and testing of YOLO and the modified YOLOv2 models were performed using Python 3.7 under the PyCharm development environment (DL framework) [73], and implemented on a system with Core i7-10750H Processor, NVIDIA GeForce RTX 2070, 32 GB RAM memory, and Windows 10 Professional x64 operating system. For the effective use of the system's GPU to accelerate the training process, CUDA 10 [74] was installed. For deployment, an RP4B with 1.5GHz 64-bit quad-core CPU, 128 GB, and 8 GB RAM memory [75] was used, as shown in Fig. 7.

B. EVALUATION METRICS

The accuracy and speed of the animal species detection using YOLOv2, modified YOLOv2, YOLOv3, and YOLOv4 were measured on the testing-set which contains 6,183 images

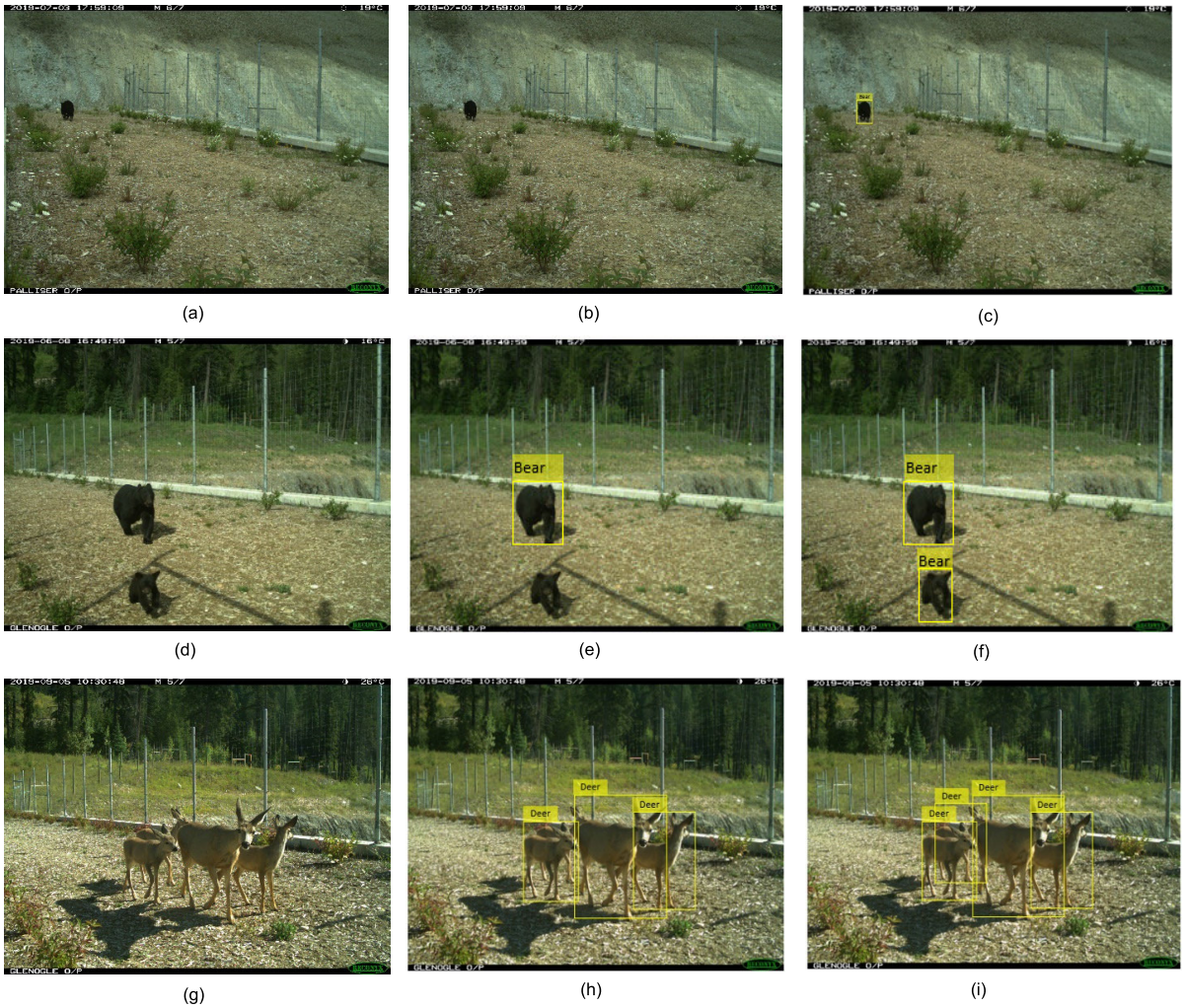

FIGURE 9. Animal species detection results of two models: (a) Original image of a far bear, (b) no detected animal (FN) using YOLOv2, (c) detected far bear using modified YOLOv2, (d) original image of two bears, (e) detected only one bear (big) using YOLOv2, (f) detected two bears (big and small) using modified YOLOv2, (g) original image of four deers, (h) detected only three deers using YOLOv2, and (i) detected four deers using modified YOLOv2.

of six animal species on the core i7 system (high-power machine) and on the RP4B (low-power machine).

The Precision-Recall Curve (PRC) [68] is a detection curve calculated from precision and recall based on four evaluation components: true positive (TP), false positive (FP), false negative (FN), and true negative (TN), as shown in equations (1) and (2). TP presents the truly detected objects as the overlap being over or equal to 0.5 [76], between ground truth and the detected bounding box in all region proposals. FP presents the falsely detected objects as the overlap being less than 0.5, between ground truth and the detected bounding box in all region proposals. FN is the sum of non-proposed regions as the ground truth does not have a match. TN represents the truly non-detected objects in the image.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The most popular metric in literature to evaluate the accuracy of object detectors, is the mAP [68]. As shown in Fig. 8, AP is defined as the area under the PRC for each class where it computes the average of precision over recall in the interval from 0 to 1. This value is then divided by the number of classes to obtain the mAP.

Inference-time (seconds per image) is used to measure the speed of Python to detect animal specie.

VII. RESULTS AND DISCUSSION

This section shows the accuracy and speed comparisons between YOLOv2, YOLOv3, YOLOv4 and the modified YOLOv2 models on the BCMoTI test dataset.

As shown in Table 4, the modified YOLOv2 model was evaluated and compared with the other models. Two machines with different computation power were used to measure the inference-time. For the high-power machine (core i7 system), the modified YOLOv2 model has a higher accuracy (mAP) than the regular YOLOv2 by 5.0% and faster

TABLE 4. Evaluation of animal species detection by using YOLOv2, modified YOLOv2, YOLOv3, and YOLOv4 models in AP, mAP, and Inference-time on the BCMoTI dataset.

Animal Species	YOLOv2 (AP)	Modified YOLOv2 (AP)	YOLOv3 (AP)	YOLOv4 (AP)
Bear	0.86	0.915	0.907	0.920
Cougar	0.750	0.778	0.762	0.831
Deer	0.900	0.936	0.925	0.915
Elk	0.823	0.865	0.854	0.871
Moose	0.791	0.838	0.860	0.898
Mountain Goat	0.856	0.904	0.903	0.930
mAP	0.830	0.872	0.868	0.894
Inference-time (second) on Core i7 system	0.025	0.022	0.049	0.020
Inference-time (second) on RP4B	0.13	0.18	1.54	0.23

detection speed by 3 ms. The accuracy of YOLOv3 is 4.5% higher than the regular YOLOv2 model; however, accuracy of the modified YOLOv2 model is higher than YOLOv3 by 0.4%. In addition, YOLOv2, even after modifications, is still faster than YOLOv3. However, YOLOv4 model is more accurate than the modified YOLOv2 model by 2.5% and faster by 2 ms. The YOLOv4 model has complex architecture and many parameters. Therefore, YOLOv4 requires a powerful platform, which restricts its deployment on embedded devices for portable real-time applications. On the other hand, for the low-power machine (RP4B), the modified YOLOv2 model (lightweight model) is faster than YOLOv4 by 50 ms. Although, accuracy of the modified model is lower than YOLOv4, this difference in accuracy can almost be ignored when compared with the improvement in the detection speed, which makes modified YOLOv2 suitable for deployment on embedded devices.

In Fig. 9, the YOLOv2 and modified YOLOv2 models were applied to some images for comparison. This is to evaluate whether the modified YOLOv2 model can solve the limitations of the original YOLOv2. The left-column images Fig. 9 (a), (d), and (g) are the original images, the middle-column images Fig. 9 (b), (e), and (h) show the detection results with the original YOLOv2 model, and the right-column images Fig. 9 (c), (f), and (i) show the detection results with the modified YOLOv2 model. From Fig. 9 (b) and (e), it is noted that YOLOv2 struggles to detect small animals while the modified YOLOv2 can detect them, as shown in Fig. 9 (c) and (f). Also, it can be seen that YOLOv2 has difficulty with occluded animals, as shown in Fig. 9 (h), but is resolved by adding three DCLs to YOLOv2, as shown in Fig. 9 (i).

The results in Fig. 9 show that the proposed "modified YOLOv2" model is more reliable than YOLOv2 in detecting small and occluded animals. Moreover, it reduces the false negative when there is an animal in the image which are not detected by YOLOv2. These enhancements can be attributed to the multi-level features merge in our modified YOLOv2 model.

VIII. CONCLUSION

A modified YOLOv2 model is proposed that can be deployed on embedded devices to detect animal species with high performance as an initial step for WVC and WHC mitigation systems to reduce the negative impact of these encounters. Our model was trained and tested on the BCMoTI dataset with three modifications to the original YOLOv2. First, to enhance the feature extraction ability, multi-level features merging was performed by adding low-level resolution features, which improved YOLOv2's capability to identify animal species with small differences by 1.8%. Second, to reduce complexity and speed up the detection process without sacrificing the accuracy, the two repeated 3×3 convolutional layers with 1024 filters in the seventh block of the YOLOv2 architecture were removed. Third, to handle geometric transformations of animals in the images, three DCLs were added to the sixth block of YOLOv2, resulting in an increase of 3.0% in the animal species detection mAP. Our results show that the mAP of the modified YOLOv2 model was increased by 5.0% and 0.4% over YOLOv2 and YOLOv3, respectively. Moreover, the modified model can detect small, far, and occluded animals better than the regular YOLOv2 model. Compared with YOLOv4, the modified model has faster detection speed which makes it feasible to be deployed on embedded devices.

To summarize, our proposed model is able to enhance detection speed without trading off accuracy in our animal species detection system.

In future work, the proposed modified model will be applied to a larger number of images with more animal species. We will also study how to improve accuracy by enhancing the extracted features in images with poor texture information and low resolution, similar to the objectives in

[77] and [78]. In addition, we aim to enhance the detection speed of animal species detector.

REFERENCES

[1] D. C. Wilkins, K. M. Kockelman, and N. Jiang, "Animal-vehicle collisions in Texas: How to protect travelers and animals on roadways," *Accident Anal. Prevention*, vol. 131, pp. 157–170, Oct. 2019.

[2] Wikipedia. *List of Fatal Bear Attacks in North America*. Accessed: Sep. 21, 2021. [Online]. Available: https://en.wikipedia.org/wiki/List_of_fatal_bear_attacks_in_North_America#2020s

[3] B. Abrahms, "Human-wildlife conflict under climate change," *Science*, vol. 373, no. 6554, pp. 484–485, Jul. 2021.

[4] E. A. Silva-Rodríguez, N. Gálvez, G. J. F. Swan, J. J. Cusack, and D. Moreira-Arce, "Urban wildlife in times of COVID-19: What can we infer from novel carnivore records in urban areas?" *Sci. Total Environ.*, vol. 765, Apr. 2021, Art. no. 142713.

[5] L. Sahagun. *Coyotes, Falcons, Deer and Other Wildlife Are Reclaiming L.A. Territory as Humans Stay at Home*. Los Angeles Times. Accessed: Sep. 30, 2021. [Online]. Available: https://www.latimes.com/environment/story/2020-04-21/wildlife-thrives-amid-coronavirus-lockdown

[6] T. S. Ajani, A. L. Imoize, and A. A. Atayero, "An overview of machine learning within embedded and mobile devices–optimizations and applications," *Sensors*, vol. 21, no. 13, p. 4412, Jun. 2021.

[7] R. Kays, S. Tilak, B. Kranstauber, P. A. Jansen, C. Carbone, M. J. Rowcliffe, T. Fountain, J. Eggert, and Z. He, "Monitoring wild animal communities with arrays of motion sensitive camera traps," 2010, *arXiv:1009.5718*.

[8] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[9] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," 2017, *arXiv:1703.06211*.

[10] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proc. 6th Int. Conf. Comput. Vis.*, Oct. 1998, pp. 555–562.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2005, pp. 886–893.

[12] D. G. Lowe, "Distinctive image features from scale-invariant key points," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[13] S. Ding, X. Zhang, Y. An, and X. Yu, "Weighted linear loss multiple birth support vector machine based on information granulation for multi-class classification," *Pattern Recognit.*, vol. 67, pp. 32–46, Jul. 2017.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.

[16] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[17] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, *arXiv:1605.07678*.

[18] M. Everingham, S. M. A. Eslami, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, 2015.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[20] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. 28th Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 1137–1149.

[22] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[24] S. Schneider, G. W. Taylor, and S. Kremer, "Deep learning object detection methods for ecological camera trap data," in *Proc. 15th Conf. Comput. Robot Vis. (CRV)*, May 2018, pp. 321–328.

[25] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in *Proc. Comput. Vis. Pattern Recognit.*, Berlin, Germany, 2018, pp. 1–12.

[26] J. Redmon. *DarkNet: Open Source Neural Networks in C (2013–2016)*. Accessed: Dec. 3, 2021. [Online]. Available: https://pjreddie.com/darknet/

[27] M.-T. Pham, L. Courtrai, C. Friguet, S. Lefèvre, and A. Baussard, "YOLO-Fine: One-stage detector of small objects under various backgrounds in remote sensing images," *Remote Sens.*, vol. 12, no. 15, p. 2501, Aug. 2020.

[28] L. Xiao, P. Zhou, K. Xu, and X. Zhao, "Multi-directional scene text detection based on improved YOLOv3," *Sensors*, vol. 21, no. 14, p. 4870, Jul. 2021.

[29] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[30] H. Zhao, C. Wang, R. Guo, X. Rong, J. Guo, Q. Yang, L. Yang, Y. Zhao, and Y. Li, "Autonomous live working robot navigation with real-time detection and motion planning system on distribution line," *High Voltage*, vol. 7, no. 6, pp. 1–13, 2022.

[31] Wikipedia. *Graphics Processing Unit*. Accessed: Dec. 12, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Graphics_processing_unit

[32] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast YOLO: A fast you only look once system for real-time embedded object detection in video," 2017, *arXiv:1709.05943*.

[33] X. Wang, S. Wang, J. Cao, and Y. Wang, "Data-driven based tiny-YOLOv3 method for front vehicle detection inducing SPP-Net," *IEEE Access*, vol. 8, pp. 110227–110236, 2020.

[34] C. Guo, X.-L. Lv, Y. Zhang, and M.-L. Zhang, "Improved YOLOv4-tiny network for real-time electronic component detection," *Sci. Rep.*, vol. 11, no. 1, p. 22744, Nov. 2021.

[35] Z. Yi, Y. Shen, and Z. Jun, "An improved tiny-YOLOv3 pedestrian detection algorithm," *Optik*, vol. 183, pp. 17–23, Apr. 2019.

[36] S. Zhang, Y. Wu, C. Men, and X. Li, "Tiny YOLO optimization oriented bus passenger object detection," *Chin. J. Electron.*, vol. 29, no. 1, pp. 132–138, Jan. 2020.

[37] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proc. Nat. Acad. Sci. USA*, vol. 115, no. 25, pp. E5716–E5725, Jun. 2018.

[38] M. S. Norouzzadeh, D. Morris, S. Beery, N. Joshi, N. Jojic, and J. Clune, "A deep active learning system for species identification and counting in camera trap images," 2019, *arXiv:1910.09716*.

[39] S. Li, J. Li, H. Tang, R. Qian, and W. Lin, "ATRW: A benchmark for Amur tiger re-identification in the wild," 2019, *arXiv:1906.05586*.

[40] J. Parham and C. Stewart, "Detecting plains and Grevy's zebras in the real-world," in *Proc. IEEE Winter Appl. Comput. Vis. Workshops (WACVW)*, Mar. 2016, pp. 1–6.

[41] B. Xu, W. Wang, G. Falzon, P. Kwan, L. Guo, G. Chen, A. Tait, and D. Schneider, "Automated cattle counting using mask R-CNN in quadcopter vision system," *Comput. Electron. Agricult.*, vol. 171, Apr. 2020, Art. no. 105300.

[42] A. Khan and S. Khan, "YOLOv2 for pigs detection in industrial farming," Norwegian Univ. Sci. Technol., Norway, Gjøvik, Tech. Rep., 2020.

[43] Z. Zhang, Z. He, G. Cao, and W. Cao, "Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification," *IEEE Trans. Multimedia*, vol. 18, no. 10, pp. 2079–2092, Oct. 2016.

[44] S. Gupta, D. Chand, and I. Kavati, "Computer vision based animal collision avoidance framework for autonomous vehicles," 2020, *arXiv:2012.10878*.

[45] D. Li, L. Deng, and Z. Cai, "Design of traffic object recognition system based on machine learning," *Neural Comput. Appl.*, vol. 33, pp. 8143–8156, Jan. 2020.

[46] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, Mar. 2020.

[47] M. Parikh, M. Patel, and D. Bhatt, "Animal detection using template matching algorithm," *Int. J. Res. Mod. Eng. Emerg. Technol.*, vol. 1, no. 3, pp. 26–32, 2013.

[48] F. Jurie and M. Dhome, "A simple and efficient template matching algorithm," in *Proc. 8th IEEE Int. Conf. Comput. Vis.*, Oct. 2001, pp. 544–549.

[49] A. Mammeri, D. Zhou, A. Boukerche, and M. Almulla, "An efficient animal detection system for smart cars using cascaded classifiers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1854–1859.
[50] K.-B. Ge, J. Wen, and B. Fang, "AdaBoost algorithm based on MB-LBP features with skin color segmentation for face detection," in *Proc. Int. Conf. Wavelet Anal. Pattern Recognit.*, Jul. 2011, pp. 40–43.
[51] S. U. Sharma and D. J. Shah, "A practical animal detection and collision avoidance system using computer vision technique," *IEEE Access*, vol. 5, pp. 347–358, 2017.
[52] S. Matuska, R. Hudec, M. Benco, P. Kamencay, and M. Zachariasova, "A novel system for automatic detection and classification of animal," in *Proc. ELEKTRO*, May 2014, pp. 76–80.
[53] W. H. S. Antônio, M. Da Silva, R. S. Miani, and J. R. Souza, "A proposal of an animal detection system using machine learning," *Appl. Artif. Intell.*, vol. 33, no. 13, pp. 1093–1106, Nov. 2019.
[54] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014, pp. 167–242. [Online]. Available: https://www.cambridge.org/core/books/understanding-machine-learning/3059695661405D25673058E43C8BE2A6#
[55] *Raspberry Pi Website*. Accessed: Dec. 8, 2021. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
[56] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, Dec. 2015.
[57] Y. Lai, "A comparison of traditional machine learning and deep learning in image recognition," *J. Phys., Conf. Ser.*, vol. 1314, no. 1, Oct. 2019, Art. no. 012148.
[58] A. Saxena, D. K. Gupta, and S. Singh, "An animal detection and collision avoidance system using deep learning," in *Advances in Communication and Computational Technology* (Lecture Notes in Electrical Engineering), Singapore: Springer, 2021.
[59] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
[60] D. Adami, M. O. Ojo, and S. Giordano, "Design, development and evaluation of an intelligent animal repelling system for crop protection based on embedded edge-AI," *IEEE Access*, vol. 9, pp. 132125–132139, 2021.
[61] D. Sato, A. J. Zanella, and E. X. Costa, "Computational classification of animals for a highway detection system," *Brazilian J. Veterinary Res. Animal Sci.*, vol. 58, May 2021, Art. no. e174951.
[62] *Reconyx Camera Traps Website*. Accessed: Dec. 21, 2021. [Online]. Available: https://www.reconyx.com/product/Professional_Series
[63] K. Boudjit and N. Ramzan, "Human detection based on deep learning YOLO-V2 for real-time UAV applications," *J. Exp. Theor. Artif. Intell.*, vol. 34, no. 3, pp. 527–544, May 2022.
[64] X. Huang, X. Wang, W. Lv, X. Bai, X. Long, K. Deng, Q. Dang, S. Han, Q. Liu, X. Hu, D. Yu, Y. Ma, and O. Yoshie, "PP-YOLOv2: A practical object detector," 2021, *arXiv:2104.10419*.
[65] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "Fighting against COVID-19: A novel deep learning model based on YOLO-V2 with ResNet-50 for medical face mask detection," *Sustain. Cities Soc.*, vol. 65, Feb. 2021, Art. no. 102600.
[66] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets V2: More deformable, better results," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9300–9308.
[67] J. Zhang, M. Huang, X. Jin, and X. Li, "A real-time Chinese traffic sign detection algorithm based on modified YOLOv2," *Algorithms*, vol. 10, no. 4, p. 127, Nov. 2017.
[68] P. Henderson and V. Ferrari, "End-to-end training of object class detectors for mean average precision," 2016, *arXiv:1607.03476*.
[69] *MATLAB Website*. Accessed: Dec. 1, 2021. [Online]. Available: https://www.mathworks.com/help/vision/ug/get-started-with-the-image-labeler.html
[70] Q. Fan, L. Brown, and J. Smith, "A closer look at faster R-CNN for vehicle detection," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 124–129.
[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[72] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[73] *PyCharm Website*. Accessed: Jan. 11, 2021. [Online]. Available: https://www.jetbrains.com/pycharm
[74] *Nvidia Developer Website*. Accessed: Dec. 12, 2021. [Online]. Available: https://developer.nvidia.com/cuda-toolkit
[75] *Electronic Design, Embedded Revolution*. Accessed: Jan. 13, 2022. [Online]. Available: https://www.electronicdesign.com/technologies/embedded-revolution/article/21134806/is-raspberry-pi-the-right-platform-for-embedded-product-development
[76] M. Ibraheam, K. F. Li, F. Gebali, and L. E. Sielecki, "A performance comparison and enhancement of animal species detection in images with various R-CNN models," *AI*, vol. 2, no. 4, pp. 552–577, Oct. 2021.
[77] R. Zhang, S. Yang, Q. Zhang, L. Xu, Y. He, and F. Zhang, "Graph-based few-shot learning with transformed feature propagation and optimal class allocation," *Neurocomputing*, vol. 470, pp. 247–256, Jan. 2022.
[78] R. Zhang, L. Xu, Z. Yu, Y. Shi, C. Mu, and M. Xu, "Deep-IRTarget: An automatic target detector in infrared imagery using dual-domain feature extraction and allocation," *IEEE Trans. Multimedia*, vol. 24, pp. 1735–1749, 2022.

MAI IBRAHEAM received the B.Sc. and M.Sc. degrees from Mansoura University, Mansoura, Egypt, in 2004 and 2009, respectively. She is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Victoria, Victoria, BC, Canada. Her research interests include artificial intelligence, data science, machine learning, computer vision, and software engineering.

KIN FUN LI (Senior Member, IEEE) is currently the Director of the two highly sought-after professional master of engineering programs in telecommunications and information security (MTIS) and applied data science (MADS) with the University of Victoria, Canada, where he teaches hardware and software courses with the Department of Electrical and Computer Engineering. He dedicates his time to instructing and researching computer architecture, hardware accelerators, education analytics, and data mining applications. He is actively involved in the organization of many international conferences, including the biennial IEEE Pacific Rim in Victoria and the internationally held IEEE AINA. He is also a passionate supporter and participant in numerous international activities to promote the engineering profession, education, and diversity. He is an Honorary Member of the Golden Key and a registered Professional Engineer in the Province of British Columbia.

FAYEZ GEBALI (Life Senior Member, IEEE) received the B.Sc. degree (Hons.) in electrical engineering from Cairo University, the B.Sc. degree (Hons.) from Shams University, and the Ph.D. degree in electrical engineering from The University of British Columbia. He held an NSERC Postgraduate Scholarship from The University of British Columbia. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Victoria. His research interests include parallel algorithms, network-on-chip, 3-D integrated circuits, digital communications, and computer arithmetic.