International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST52411226

Smart Agriculture: Enhancing Security Through Animal Detection Via Deep Learning and Computer Vision

[1]A Samuvel, [2]Dr G Manikandan, [3]Ms. Vilma Veronica, [4]Ms. S. Hemalatha
[1]PG Student, Kings Engineering College, Sriperumbudhur, Tamil Nadu, India
[2]Professor, Kings Engineering College, Sriperumbudhur, Tamil Nadu, India
[3]Assistant Professor, Kings Engineering College, Sriperumbudhur, Tamil Nadu, India
[4]Assistant Professor, Kings Engineering College, Sriperumbudhur, Tamil Nadu, India

ARTICLEINFO

Article History:
Accepted: 03 March 2024
Published: 13 March 2024

Publication Issue :
Volume 11, Issue 2
March-April-2024
Page Number :
140-159

ABSTRACT

Agriculture stands as a crucial sector, making significant contributions to the economies of many countries. Nevertheless, it encounters various challenges, one of which is animal disruption. This poses a considerable threat to crops, leading to financial losses for farmers. In response to this concern, we have engineered an animal disruption warning system for agricultural settings based on YOLOv6 technology.

The system operates by analyzing live video feeds from strategically placed cameras. Utilizing deep learning algorithms, it can detect and classify animals in real-time. The computer vision algorithms enable tracking and prediction of animal movements. Upon detection, the system promptly sends alerts, enabling timely and appropriate actions.

In this paper, we periodically monitor the entire farm through a camera that continuously records its surroundings. The identification of animal entry is achieved using a deep learning model, and alarm systems serve as a deterrent, notifying forest officials. This report provides details on the libraries and convolutional neural networks employed in constructing the model.

This research focuses on the implementation of a robust animal detection system in agricultural environments, leveraging the capabilities of deep learning. The project utilizes state-of-the-art deep neural networks and computer vision algorithms to analyze live video feeds from strategically positioned cameras across the farm. The deep learning model is trained to detect and classify various animals in real-time, contributing to the early identification of potential threats to crops.

The system employs sophisticated computer vision techniques, enabling accurate tracking and prediction of animal movements within the monitored areas. Upon detection, the system triggers timely alerts, providing farmers with the necessary information to take swift and appropriate actions, thereby mitigating potential damage to crops.

To achieve these objectives, the project involves periodic monitoring of the entire farm through a camera that continuously records its surroundings. The deep learning model, supported by alarm systems, effectively identifies animal entries, serving as a proactive deterrent. This research report outlines the libraries, frameworks, and convolutional neural networks employed in the development of the animal detection model, shedding light on the technical aspects of its implementation.

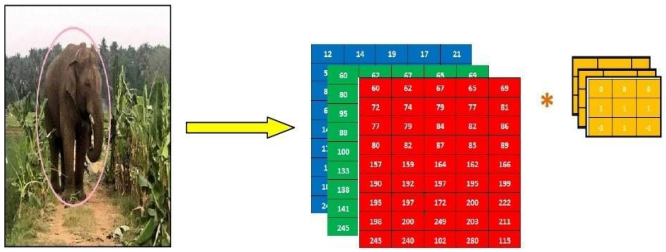
The integration of deep learning and computer vision in agriculture not only enhances crop protection but also contributes to the sustainable and efficient management of farming practices. This research offers insights into the potential of advanced technologies to address challenges in agriculture and opens avenues for further exploration in the intersection of technology and agriculture.

INDEX TERMS: Animal detection, YOLO V6 TECHNOLOGY, Convolutional Neural Network, Video Surveillance, Wild animal monitoring, Creating of Alert Message

I. INTRODUCTION

Deep Convolutional Neural Networks

Deep Convolutional Neural Networks (CNNs) are a class of artificial neural networks that are particularly suited for image recognition and classification tasks. They are inspired by the structure of the visual cortex of the brain, which is responsible for processing visual information.



- CNNs consist of multiple layers, each of which performs a specific type of computation. The first layer is typically a convolutional layer, which applies a set of filters to the input image to extract features such as edges and textures. The output of the convolutional layer is then passed through a non-linear activation function, such as a rectified linear unit (ReLU), to introduce non-linearity into the model.
- The subsequent layers of a CNN are typically pooling layers, which reduce the dimensionality of the feature maps produced by the convolutional layers. Pooling is typically achieved by taking the maximum or average value of a small patch of the feature map. The purpose of pooling is to reduce the computational complexity of the model and to

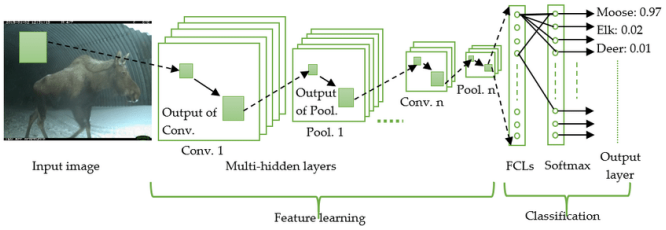
make it more robust to small variations in the input.

- The final layers of a CNN are typically fully connected layers, which perform the actual classification task. The output of the last pooling layer is flattened into a vector and passed through one or more fully connected layers, which compute a probability distribution over the classes. The final output of the network is the class with the highest probability.
- Training a CNN involves adjusting the weights of the filters and fully connected layers so as to minimize a loss function, such as cross-entropy. This is typically done using gradient descent or a variant thereof, such as Adam or RMSprop.

Overall, CNNs have proven to be highly effective at a wide range of computer vision tasks, including image recognition, object detection, and image segmentation. They have also been applied to other domains, such as natural language processing and speech recognition, with great success.

Convolutional Layer

A Convolutional Layer is the main building block of a Convolutional Neural Network (CNN). It performs a mathematical operation called convolution, which is the process of applying a set of filters (also known as kernels) to an input image or feature map to extract relevant features.



- The filters used in a convolutional layer are typically small matrices (e.g., 3x3 or 5x5) that are learned during the training process. The filter is applied to each location of the input feature map, and the resulting values are combined to produce a new feature map.
- The main advantage of using convolutional layers in a neural network is their ability to capture spatial patterns and local dependencies in an input image. This makes them highly effective for tasks such as image recognition, object detection, and semantic segmentation.
- Convolutional layers also incorporate a non-linear activation function, such as ReLU, which introduces non-linearity into the model and allows it to learn complex representations.
- Additionally, a convolutional layer may include other operations, such as padding, which adds zeros around the borders of the input feature map to preserve its spatial dimensions. Another operation is stride, which specifies the number of pixels to move the filter across the input feature map.

In summary, a convolutional layer is a fundamental component of a CNN that performs convolutional operations to extract relevant features from an input image or feature map, and it includes other operations such as activation functions, padding, and stride to improve the model's effectiveness.

ReLU Activation Layer

ReLU (Rectified Linear Unit) is an activation function commonly used in neural networks, including Convolutional Neural Networks (CNNs). The purpose of the activation function is to introduce non-linearity into the model, allowing it to learn more complex representations.

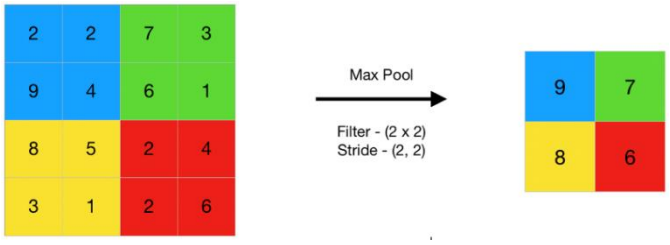
- The ReLU activation function applies the element-wise rectification operation f(x) = max(0, x) to the input. In other words, it replaces any negative input values with zero and leaves the positive values unchanged.

- The main advantage of ReLU is its simplicity and computational efficiency. The rectification operation is easy to compute and has a gradient of 1 for positive values and 0 for negative values. This makes it easier to train the model using back propagation, which relies on gradients to adjust the weights of the network during training.
- Another advantage of ReLU is its ability to avoid the problem of vanishing gradients, which can occur with other activation functions such as sigmoid or tanh. This problem arises when the gradient of the activation function approaches zero for large or small values of the input, making it difficult to propagate gradients back through the network during training. ReLU, on the other hand, has a constant gradient of 1 for positive values, which helps to prevent the vanishing gradient problem.

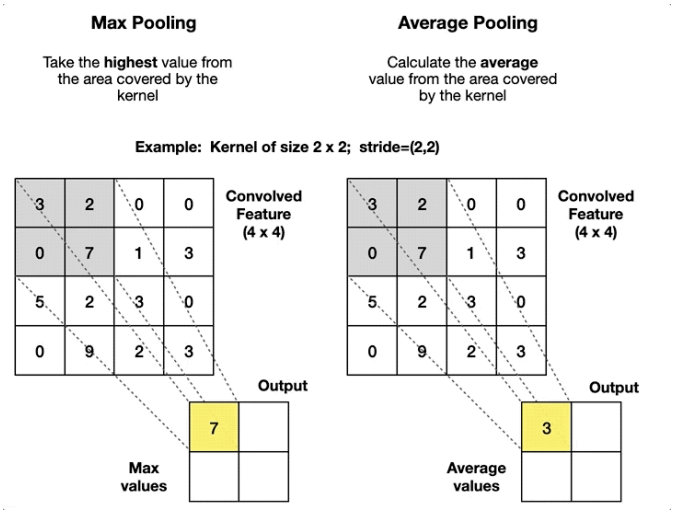
In summary, ReLU is a popular activation function used in CNNs and other neural networks. It introduces non-linearity into the model, is computationally efficient, and helps to prevent the problem of vanishing gradients during training.

Pooling Layer

A pooling layer is a common type of layer in Convolutional Neural Networks (CNNs) that is typically used after one or more convolutional layers. The purpose of a pooling layer is to reduce the spatial dimensionality (i.e., the size) of the feature maps produced by the convolutional layers, while preserving the most important information.



- The most common type of pooling is max pooling, which operates on small sub-regions (e.g., 2x2 or 3x3) of the input feature map and outputs the maximum value of each sub-region. This has the effect of down sampling the feature map by a factor of the size of the sub-region.
- Another type of pooling is average pooling, which computes the average value of each sub-region instead of the maximum. This can be useful in situations where the network needs to be more robust to small variations in the input, as it tends to smooth out the output.
- Pooling layers have several benefits. Firstly, they help to reduce the computational complexity of the model by reducing the number of parameters that need to be learned. This can help to prevent over fitting and improve the generalization performance of the model. Secondly, pooling can help to make the network more robust to small translations or distortions of the input, as the output is less sensitive to the exact position of the features in the input image.



In summary, pooling layers are a common type of layer in CNNs that are used to down sample the feature maps produced by convolutional layers. Max pooling and average pooling are the two most common types of pooling, and they can help to reduce the computational complexity of the model and improve its robustness to small variations in the input.

Flattening Layer

In deep learning, a flattening layer is a layer that is often used to convert a multidimensional tensor into a one-dimensional tensor, so that it can be fed into a fully connected layer. The flattening layer does not have any learnable parameters, and its only purpose is to restructure the input data in a way that makes it suitable for processing by the next layer.

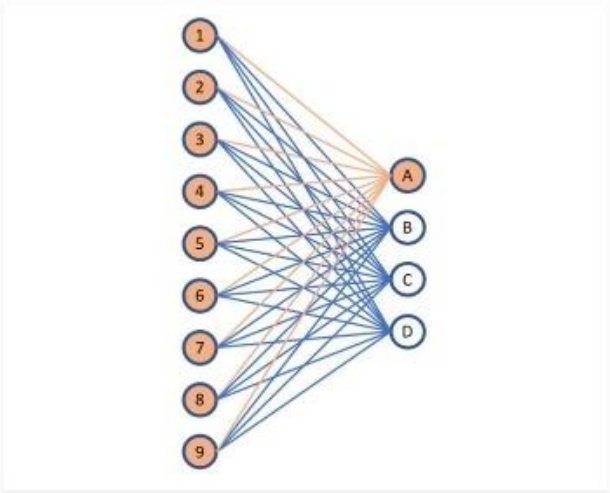
The input to a flattening layer is typically a tensor with dimensions (batch size, height, width, depth), where batch size is the number of examples in a batch, height and width are the spatial dimensions of the input data (e.g., the height and width of an image), and depth is the number of channels in the input data (e.g., 3 channels for an RGB image).

The flattening layer then reshapes the input tensor into a one-dimensional tensor with shape (batch size, height * width * depth), which can be fed into a fully connected layer or another type of layer that expects a one-dimensional input. This transformation allows the network to learn more complex representations of the input data, by connecting every neuron in one layer to every neuron in the next layer.

Overall, the flattening layer is a simple but important component of much deep learning architecture, particularly in computer vision tasks that involve processing images or other multidimensional data.

Fully Connected Layer

In deep learning, a fully connected layer is a type of layer that connects every neuron in the previous layer to every neuron in the next layer. It is also known as a dense layer or a fully connected neural network layer.



The input to a fully connected layer is typically a one-dimensional tensor, which is produced by flattening the output of the previous layer. The output of a fully connected layer is also a one-dimensional tensor, where each element in the output tensor represents the activation of a single neuron in the layer.

The purpose of a fully connected layer is to learn complex nonlinear relationships between the input and output data. Each neuron in the layer computes a weighted sum of its inputs, which is passed through an activation function to produce the output of the neuron. The weights of the connections between the neurons are learned during training, using a back propagation algorithm that updates the weights to minimize the loss function.

A deep neural network can have multiple fully connected layers, which are typically used in the later stages of the network to perform the final classification or regression task. The number of neurons in each layer, as well as the number of layers in the network, is a hyper parameter that can be tuned to optimize the performance of the model.

Overall, fully connected layers are a powerful and flexible component of deep neural networks, and they have been used in a wide range of applications, including image classification, natural language processing, and speech recognition.

1) Background [the setting of the scene of the problem]

Animal detection using deep learning is an application of computer vision that involves the use of deep

neural networks to identify and classify animals in digital images and videos. This technology has a wide range of potential applications in various fields, such as wildlife conservation, agriculture, and surveillance. In the context of animal detection, deep learning algorithms can be trained on large datasets of labeled images and videos to learn the visual features and characteristics of different animals. Once trained, these algorithms can be used to automatically detect and classify animals in new images and videos with high accuracy.

Animal detection using deep learning has the potential to revolutionize the way we study and protect wildlife, by enabling researchers to track and monitor animals in real-time, without the need for human intervention. It also has applications in agriculture, where it can be used to detect and prevent crop damage from wildlife, and in surveillance, where it can be used to monitor wildlife activity in urban areas.

2) Statement [Exact problem you are trying to solve]

The problem statement of animal detection using deep learning is to accurately detect and classify animals in digital images and videos using deep neural networks. This problem statement encompasses several challenges, including:

Data scarcity: There is often a limited amount of labeled data available for training deep learning models for animal detection. This makes it challenging to train models that can generalize well to new and unseen data.

Variability in animal appearance: Animals can exhibit a wide range of visual appearances depending on factors such as lighting conditions, camera angle, and occlusions. This variability makes it challenging to develop robust models that can accurately detect animals across a variety of conditions.

Real-time detection: In many applications of animal detection, such as wildlife monitoring and surveillance, it is necessary to perform real-time detection of animals. This requires models that can perform inference quickly and efficiently, often in resource-constrained environments.

Addressing these challenges requires the development of novel deep learning architectures and training techniques that can learn robust and generalizable representations of animal features, while also being scalable and efficient for real-time applications.

3) Motivation

THANJAVUR: As farmers get ready for samba harvest, wild boar attack paddy fields all through the night leaving farmers helpless as confronting them in any form can pose serious threat to human lives. They literally plough through the fields, flipping over the soil in search of earth worms.

(Jan 19, 2023, 12:13 IST)

There are several motivations behind animal detection using deep learning, including:

Wildlife conservation: Animal detection using deep learning can be used to monitor and track endangered species, allowing researchers and conservationists to better understand their behaviors, habitat requirements, and population dynamics. This information can be used to inform conservation efforts and protect endangered species from extinction.

Agriculture: Animal detection using deep learning can also be used in agriculture to detect and prevent crop damage caused by wildlife. By accurately detecting and identifying animals in agricultural settings, farmers can take proactive measures to prevent damage and minimize economic losses.

Surveillance: Animal detection using deep learning can also be used for surveillance and security purposes, such as monitoring wildlife activity in urban areas or detecting intrusions in protected areas. This can help prevent conflicts between humans and wildlife and enhance public safety.

Scientific research: Animal detection using deep learning can be used in scientific research to study animal behavior and ecology. By automatically detecting and tracking animals in digital images and videos, researchers can collect large amounts of data quickly and efficiently, enabling them to study animal behavior at a scale that was previously impossible.

Overall, animal detection using deep learning has the potential to significantly improve our understanding of wildlife and enable us to take more proactive measures to protect and conserve endangered species.

4) Challenges [Difficulty in problem-solving]

There are several challenges in animal detection using deep learning, including:

Limited data: One of the major challenges in animal detection using deep learning is the limited availability of labeled data. This is particularly true for rare or endangered species, where obtaining large amounts of labeled data can be difficult or even impossible.

Variability in animal appearance: Animals can have significant variation in their appearance, making it difficult to accurately detect and classify them. Factors such as lighting conditions, camera angle, and occlusion can all affect animal appearance, leading to challenges in developing robust models.

Class imbalance: Another challenge is class imbalance, where some animal classes are much more common than others in real-world datasets. This can lead to biased models that perform poorly on underrepresented classes.

Computational complexity: Deep learning algorithms for animal detection can be computationally intensive, requiring significant computational resources for training and inference. This can be a challenge in resource-constrained environments, such as embedded systems or mobile devices.

Generalization: Deep learning models for animal detection often struggle to generalize well to new and unseen data. This can be particularly challenging in the context of wildlife monitoring, where animals may exhibit significant variation in their behavior and appearance.

5) Essence of your approach

Animal detection using deep learning can be solved by following these steps:

Data collection and preparation: Collecting and preparing a large and diverse dataset of images and videos containing the animals of interest is a critical first step in solving animal detection using deep learning. The dataset should be annotated with labels indicating the location and class of each animal in the image or video.

Model selection and training: Selecting an appropriate deep learning model architecture and training it on the collected dataset is the next step. The choice of model architecture will depend on the specific requirements of the problem, such as the number of classes, image size, and available computational resources. The model can be trained using supervised learning techniques, such as backpropagation, on the annotated dataset.

Validation and testing: Once the model is trained, it should be validated and tested on a separate dataset to evaluate its performance. This can be done using metrics such as accuracy, precision, recall, and F1 score. The model should also be tested on unseen data to evaluate its generalization performance.

Deployment: Once the model is optimized and validated, it can be deployed in a real-world application. This may involve integrating the model into an existing system, such as a wildlife monitoring network or a security camera system, or developing a new application from scratch.

By following these steps, animal detection using deep learning can be effectively solved, enabling researchers, conservationists, farmers, and security professionals to detect and monitor animals with greater accuracy and efficiency.

6) Statement of assumption

Deep learning is a powerful approach for animal detection because it can automatically learn to extract relevant features from raw data, such as images or videos, without the need for explicit feature engineering. This makes it well-suited to tasks such as object detection and classification, which are key components of animal detection.

In particular, deep learning models such as convolutional neural networks (CNNs) have been shown to be highly effective for animal detection, achieving state-of-the-art performance on a variety of animal detection tasks. These models can be trained on large and diverse datasets of animal images and videos, enabling them to learn to recognize and classify animals based on their visual features.

One of the key advantages of deep learning for animal detection is its ability to handle the significant variability in animal appearance that can arise from factors such as lighting conditions, camera angle, and occlusion. Deep learning models can learn to extract robust and discriminative features from animal images and videos, enabling them to accurately detect and classify animals even in challenging conditions.

Overall, deep learning is highly applicable for animal detection, enabling researchers, conservationists, farmers, and security professionals to automatically detect and monitor animals with high accuracy and efficiency.

7) Aim & Objective

The aim of animal detection using deep learning is to automatically identify and classify animals in images or videos using advanced machine learning techniques. The primary objective is to develop accurate and efficient algorithms that can identify different types of animals and distinguish them from other objects in the image or video. This has various applications, including wildlife monitoring, animal conservation, and agricultural management.

The main objectives of animal detection using deep learning are:

- Accurately detect and identify animals in images and videos: Deep learning algorithms can be trained to recognize various species of animals with high accuracy. The objective is to develop algorithms that can detect and identify animals in images and videos in real-time with high precision.
- Improve wildlife monitoring: Animal detection using deep learning can improve wildlife monitoring efforts by automating the identification of animals in camera trap images and videos, allowing researchers to analyze large volumes of data quickly and accurately.
- Enhance animal conservation efforts: By accurately identifying and tracking animal populations, deep learning algorithms can help conservationists better understand animal behavior, population dynamics, and habitat use. This information can then be used to develop conservation strategies to protect endangered species and their habitats.
- Optimize agricultural management: Animal detection using deep learning can help farmers monitor livestock and identify health issues early, reducing the risk of disease outbreaks and improving animal welfare.

- Reduce human-wildlife conflicts: Deep learning algorithms can be used to identify animal behavior patterns that indicate potential conflicts with humans, such as crop damage or livestock predation. This information can help farmers and wildlife managers take proactive measures to mitigate these conflicts.

II. EXCISTING SYSTEM

The system architecture of the excisting hybrid VGG 19 Bi-LSTM model is demonstrated in figure 1. The pro- posed architecture comprises five phases of development steps, which includes data pre-processing, animal detection, VGG 19 pre-trained model-based classification, extracting the prediction results, and sending alert messages. In the data pre-processing phase, 45k animal images were collected from different datasets such as camera trap, wild animal, and the hoofed animal dataset. The collected images were rescaled to the size of 224 224 pixels and denoised. In the second phase, we pass the pre-processed images into YOLOR object detection model [39], which identifies the animal present in an image using bounding boxes as illustrated in Fig. 4. In the third phase, using hybrid VGG 19 Bi-LSTM model we perform image classification tasks and class label prediction was done and animal details are extracted using LSTM Net- works. In the fourth phase, we collect the location information of the animal, and the web server creates a SMS alert and sends it to the forest officers

III.PROPOSED SYSTEM

YOLO6

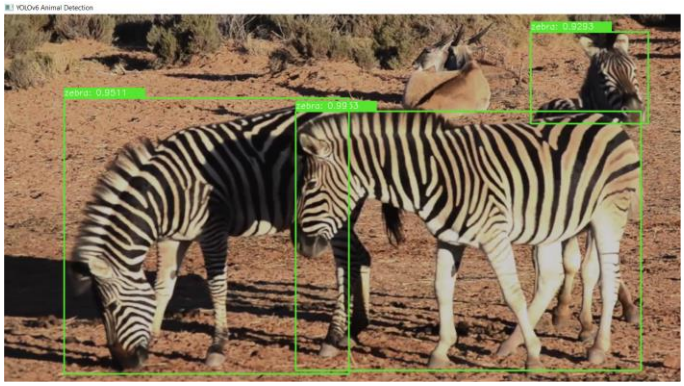
YOLO6 (You Only Look Once version 6) is a real-time object detection algorithm developed by Alexey Bochkovskiy and his team. It is a deep neural network-based algorithm that can detect multiple objects in an image or video stream with high accuracy and speed.

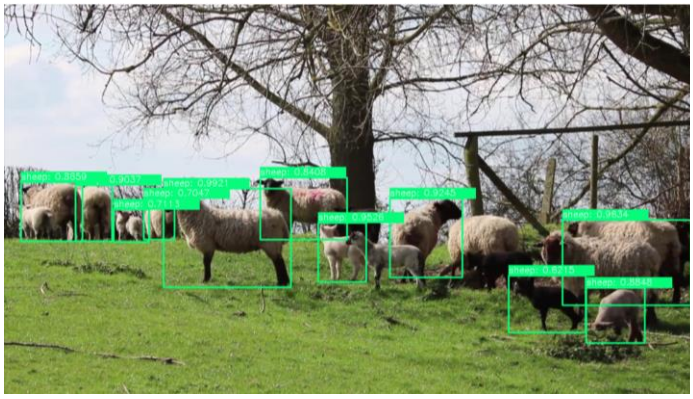
The YOLO6 algorithm is designed for real-time object detection applications, such as autonomous driving, robotics, and surveillance. It is trained on large datasets of labeled images and videos and can detect a wide range of objects, including people, vehicles, and animals, with high accuracy and speed. The algorithm is available as an open-source project and can be easily integrated into various applications using popular deep learning frameworks such as PyTorch and TensorFlow.

YOLO6 can be used for animal detection tasks with high accuracy and speed. Animal detection using YOLO6 involves training the algorithm on a large dataset of labeled animal images and videos. The dataset needs to include a variety of animal species and variations in poses, orientations, and lighting conditions to ensure that the algorithm can generalize well to new, unseen images.

The YOLO6 algorithm can detect various animal species, including mammals, birds, and reptiles. It can also detect animals in various settings, such as forests, grasslands, and aquatic environments. The algorithm can detect animals of different sizes, from small birds to large mammals, and can detect animals in both static and dynamic scenes.

YOLO6 can be particularly useful for wildlife monitoring and conservation efforts, where the rapid detection of animals is crucial for protecting endangered species and managing their habitats. It can also be used for agricultural management to monitor livestock and identify potential health issues.





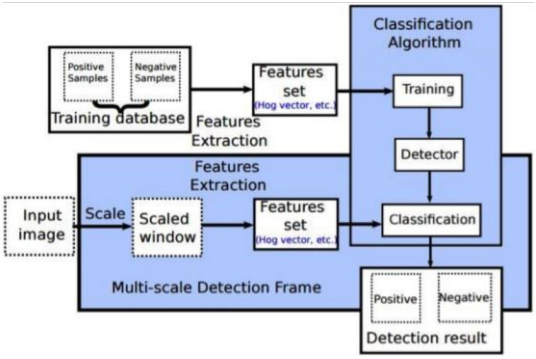
IV. METHODOLOGY

- Importing the Dataset
- Pre-processing of the images
- Model training
  - Model Building
  - Compiling the model
  - Fitting the model
- Model Evaluation
- Results

If result is POSITIVE(wild) then alarm begin in forest office.

If result is NEGATIVE(certain domestic) then alarm to respective farmer.

Yolo v6 Architecture



- Object detection is a computer vision task that involves identifying and locating objects in images or videos. It is an important part of many applications, such as surveillance, self-driving cars, or robotics.
- **Single-shot object detection** uses a single pass of the input image **to make predictions about the presence and location of objects** in the image. It processes an entire image in a single pass, making them **computationally efficient**.
- YOLO is a single-shot detector that uses a **fully convolutional neural network (CNN)** to process an image.
- One of the most popular OS projects in computer vision is YOLO (You Only Look Once).
- YOLO is an efficient real-time object detection algorithm, first described in the seminal 2015 paper by Joseph Redmon et al.
- YOLO divides an image into a grid system, and each grid detects objects within itself. It can be used for real-time inference and require very few computational resources.
- 7 years after the first version of YOLO was released, the research group at Meituan published the new YOLOv6 model.

OVERVIEW/ LITERATURE REVIEW

[1] Yang et.al , proposed Convolutional SuperResolution Layers grouped together various Super Resolution algorithms into four groups: edge-based methods, example-based methods, prediction model and image statistical methods. The state-of-the-art performance is achieved in Convolutional Neural Network as CNN has been adopted for super-resolution recently, and attempted to use convolutional neural networks for image super-resolution.

[2] Cheng et al, proposed the system uses discriminative features for classifying the bird species based on parts of birds that uses a support vector machine along with Normal Bayes classifier. Fine-Grained Image Categorization – Technique for discriminating fine-grained classes which can be divided into two very important maingroups useful for future work as well.

[3]Peng et al. proposed the method of transforming the detailed texture of information in High-resolution images to Low-resolution images with the help of fine-tuning for boosting the accuracy of recognizing fine-grained objects in Low-resolution images. This technique has certain limitations such as it requires the High resolution images for the training of a model which limits their generalized implementation. [4] Marini et al proposed an approach to eliminate background elements using a colors segmentation and compute normalized color histograms to extract feature vectors for classification. Fine-Grained Image Categorization – Technique for discriminating fine-grained classes which can be divided into two main groups.

[5] In 2018 yamanaka et al , made a tensor flow implementation of last and accurate image super resolution by CNN with skip connection and network in network a deep learning based single image super resolution model.

[6] In June 2018, A Kamilaris conducted a survey of research efforts that employ CNN Convolution Neural Network, which constitutes a specific class of Deep learning, which is applied to various agricultural and food production challenges. CNN are compared with other existing techniques considering its advantages and disadvantages and the overall findings indicates that CNN constitutes a promising technique with high performance in terms of precision and classification accuracy, outperforming existing commonly used image processing techniques. Given that the success of each CNN model is also highly dependent on the quality of data used.

[7] In July 2019, Mohamed Elsayed Abd Elaziz presented a new multi-objective metaheuristic based on a multiverse optimization algorithm on segment grayscale images via multi-level thresholding.The result showed that the presented method provides better approximation to the optimal than the other algorithms in terms of hypervolume and spacing plus the quality of the segmented image is better than those of the other methods in terms of uniformity measures.

[8] Yu Han Lieu, 2018 , Presented a thorough recognition process of CNN with its local connection and weight sharing characteristics, CNN Manages to scan the images and extract objects' features with much lower compute cost. He also states the forth coming challenges is to provide stable calculating environment and increase the computing speed with help of hardware and software. CNN gives and Excellent performance on image Recognition

[9] In July 2017, Neha Sharma, vibhor Jain, presented an empirical analysis of the performance of popular CNN for identifying images. Upon taking various types of Datasets for image classification to test the performance of CNN as a single type of dataset does not reveal the true capability and limitations.Upon only testing the images and not training they found that CNN's vary substantially across different categories of images provided and also outperforming in the analysis.

[10] Victoria Yoon evaluated some of the most used activation function and how they impact the time to train a CNN model and the performance of the

model.The result showed that the Rectified Linear unit activation function trains the CNN model quicker than any other activation function.Based on the Evaluation performed if every decimal digit of the achieved accuracy is important and there is sufficient time, Leaky Relu is Recommended.

4) Progression Highlights & Limitations

YOLOv1

YOLOv1 (You Only Look Once version 1) is an object detection algorithm developed by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in 2016. It was one of the first models to propose real-time object detection using a single neural network.

The YOLOv1 algorithm divides the input image into a grid of cells and predicts the bounding boxes and class probabilities for each cell. The bounding boxes are represented as an (x,y) coordinate of the center of the box, its width and height, and a confidence score indicating how likely it is that the box contains an object. The class probabilities represent the likelihood of the object belonging to a certain class.

The YOLOv1 model uses a convolutional neural network (CNN) as its backbone, which is trained on large-scale object detection datasets such as PASCAL VOC and COCO. The network consists of 24 convolutional layers followed by 2 fully connected layers, which predicts the bounding boxes and class probabilities for each cell.

The YOLOv1 algorithm has several advantages over other object detection algorithms. Firstly, it is faster and more efficient, as it processes the entire image in a single pass. Secondly, it has a high detection accuracy, particularly for small objects, due to its ability to capture context information within the grid cells. However, it has some limitations, such as difficulty in detecting objects with extreme aspect ratios or objects that are partially occluded.

YOLOv1 has since been succeeded by newer versions of the YOLO algorithm, such as YOLOv2, YOLOv3, and YOLOv4, which have improved the accuracy and speed of the model, as well as added new features such as anchor boxes and feature pyramid networks.

YOLOv2

YOLOv2 (You Only Look Once version 2) is a popular real-time object detection algorithm that was introduced by Joseph Redmon and Ali Farhadi in 2016 as an improvement over the original YOLO algorithm.

One of the main improvements in YOLOv2 is the use of a deeper neural network architecture called Darknet-19, which consists of 19 convolutional layers and can perform more complex feature extraction than the original YOLO architecture. This allows YOLOv2 to detect smaller objects with higher accuracy.

In addition, YOLOv2 uses a new method called anchor boxes to improve its detection performance. Anchor boxes are pre-defined bounding boxes with fixed aspect ratios and sizes that are used to match objects in the image with specific anchor boxes during training. This helps YOLOv2 better localize and identify objects of different shapes and sizes.

Another improvement in YOLOv2 is the use of batch normalization, which normalizes the activations in each layer across a batch of training examples. This helps improve the generalization ability of the model and reduces overfitting.

Overall, YOLOv2 is a fast and accurate object detection algorithm that is widely used in computer vision applications such as autonomous driving, surveillance, and robotics.

YOLOv3

YOLOv3 (You Only Look Once version 3) is the latest version of the YOLO object detection algorithm, developed by Joseph Redmon and his team in 2018. It builds on the previous versions of YOLO with several improvements to increase accuracy and speed.

One major improvement in YOLOv3 is the use of a feature pyramid network (FPN), which is a multi-scale feature extraction technique. This allows

YOLOv3 to detect objects at different scales and resolutions, improving its ability to detect small objects.

Another improvement is the use of a technique called spatial pyramid pooling (SPP), which allows the network to operate on an image of any size, rather than requiring a fixed input size like previous versions of YOLO.

YOLOv3 also uses a new backbone network called Darknet-53, which is deeper and more powerful than the previous Darknet-19 used in YOLOv2. This allows YOLOv3 to learn more complex representations of the input image.

To improve training efficiency, YOLOv3 uses a technique called transfer learning, where the network is pre-trained on a large dataset (such as ImageNet) before being fine-tuned on the specific object detection task.

Overall, YOLOv3 achieves state-of-the-art accuracy while maintaining real-time performance, making it a popular choice for a wide range of applications, including self-driving cars, surveillance, and robotics.

YOLOv4

YOLOv4 (You Only Look Once version 4) is the latest version of the YOLO object detection algorithm, developed by Alexey Bochkovskiy and his team in 2020. YOLOv4 builds on the improvements made in YOLOv3, with further enhancements to improve accuracy and speed.

One of the key improvements in YOLOv4 is the use of a larger and deeper neural network architecture. The backbone network used in YOLOv4 is called CSPDarknet-53, which is based on Darknet-53 used in YOLOv3, but with the addition of a cross-stage partial network (CSP) module. This allows YOLOv4 to extract more powerful features from the input image, improving its accuracy.

Another improvement is the use of a technique called Mosaic data augmentation, where multiple images are combined into a single mosaic image during training. This helps to improve the generalization ability of the model by training it on a more diverse set of images.

YOLOv4 also uses a technique called bag-of-freebies (BoF) to further improve its performance. BoF is a set of optimization techniques that improve the efficiency and accuracy of the model, including improved training techniques, better data augmentation, and architectural changes.

Overall, YOLOv4 achieves state-of-the-art performance in terms of accuracy and speed, making it one of the most popular object detection algorithms used today in various applications including autonomous vehicles, robotics, and surveillance.

YOLOv5

YOLOv5 (You Only Look Once version 5) is a deep learning object detection algorithm developed by Ultralytics in 2020. YOLOv5 is based on the YOLOv4 architecture and builds on it by making several key improvements in terms of accuracy, speed, and ease of use.

One of the major improvements in YOLOv5 is the use of a more lightweight neural network architecture called CSPNet, which stands for Cross-Stage Partial Network. CSPNet is a faster and more efficient backbone network that allows YOLOv5 to perform object detection at real-time speeds.

Another important improvement is the use of a new data augmentation technique called CutMix. CutMix randomly crops and pastes different images together during training, creating a more diverse set of training examples and improving the model's ability to generalize to new images.

YOLOv5 also includes a number of other optimizations such as better anchor box selection, class label smoothing, and improved hyperparameter tuning.

One of the unique features of YOLOv5 is that it is easy to use and customize, thanks to its modular design and the availability of pre-trained models. This makes it accessible to a wide range of users, including those with limited machine learning expertise.

Overall, YOLOv5 is a state-of-the-art object detection algorithm that offers high accuracy and real-time performance, while being easy to use and customize for different applications.

YOLOv6

As of my knowledge cutoff (September 2021), there is no official release of YOLOv6 yet. However, there have been rumors and speculations about a potential release of YOLOv6 in the future, as the YOLO series has been continuously updated with new versions that improve the accuracy and speed of object detection.

It's worth noting that YOLOv5 already achieved state-of-the-art performance, so it remains to be seen what additional improvements YOLOv6 could bring to the table. However, it's possible that YOLOv6 could include further optimizations in terms of architecture, training techniques, or data augmentation methods to further enhance the accuracy and speed of object detection.

As of my current date (April 2023), there may have been new developments or updates on YOLOv6 that I am not aware of, as research in deep learning and computer vision is constantly evolving.

The renovated design of YOLOv6

YOLOv6 is an object detection algorithm that can detect objects in real-time. The steps for implementing YOLOv6 are:

- Install dependencies: You need to install the required dependencies such as PyTorch, OpenCV, and other libraries.
- Prepare the dataset: You need to prepare the dataset with annotated images for training the YOLOv6 model. You can use tools such as LabelImg to annotate the images.
- Configure the model: You need to configure the YOLOv6 model according to your requirements. You can modify the model architecture, hyperparameters, and other settings in the configuration file.
- Train the model: You can train the YOLOv6 model on the annotated dataset using the command-line interface. The training process can take several hours or days, depending on the size of the dataset and the complexity of the model.
- Evaluate the model: You can evaluate the performance of the trained YOLOv6 model on a validation dataset. You can calculate metrics such as precision, recall, and F1 score to measure the accuracy of the model.
- Test the model: You can test the YOLOv6 model on new images or videos to detect objects in real-time. You can use the pre-trained weights or the trained weights from your own dataset to perform the inference.

These are the general steps for implementing YOLOv6.

The YOLOv6 algorithm is an animal detection algorithm that uses a deep neural network to detect animals in real-time. Here are the main steps in the YOLOv6 algorithm:

- Input image: The algorithm takes an input image or video frame as its input.
- Preprocessing: The input image is preprocessed by resizing it to a fixed size and converting it to a tensor.
- Backbone network: The input tensor is passed through a backbone network, which extracts high-level features from the image.
- Neck network: The output of the backbone network is passed through a neck network, which further refines the features and generates a set of feature maps at different resolutions.
- Head network: The feature maps generated by the neck network are used by the head network, which is responsible for predicting the object detection outputs. The head network is composed of several convolutional layers and uses a set of anchor boxes to predict the bounding boxes, class probabilities, and confidence scores for each object.

- Non-maximum suppression: The outputs from the head network are postprocessed using non-maximum suppression (NMS) to remove overlapping bounding boxes and keep only the most confident ones.
- Output: The final output of the algorithm is a set of bounding boxes, class probabilities, and confidence scores for each detected object in the input image or video frame.
- Thresholding: The output of the algorithm can be filtered based on a confidence threshold, which removes detections with low confidence scores.
- Display or store the output: The final output can be displayed on the input image or saved as a separate file. These are the main steps in the YOLOv6 algorithm.

a. Deep Learning Frameworks

YOLOv6 is a state-of-the-art object detection algorithm that uses a deep neural network to detect objects in real-time. The algorithm is built on the PyTorch deep learning framework and is implemented using a single-stage object detection approach.

TensorFlow is an open-source deep learning framework that provides a flexible and efficient platform for building and training deep neural networks.

YOLOv6 is built using the TensorFlow deep learning framework and is implemented using a single-stage object detection approach. This approach involves predicting the bounding boxes, class probabilities, and confidence scores for each object in a single pass through the network. This is in contrast to two-stage object detection algorithms such as Faster R-CNN, which use a separate region proposal network to generate candidate object regions before predicting the object detections.

The TensorFlow implementation of YOLOv6 includes a backbone network, a neck network, and a head network. The backbone network extracts high-level features from the input image, which are then refined by the neck network. The head network generates the final object detection outputs using the refined features from the neck network.

The TensorFlow implementation of YOLOv6 also includes several optimizations, such as using anchor boxes to improve the accuracy of the bounding box predictions and using cross-stage partial connections to improve the information flow between the backbone and neck networks.

In summary, YOLOv6 is built on the PyTorch deep learning framework and is implemented using a single-stage object detection approach. The TensorFlow implementation of YOLOv6 includes a backbone network, a neck network, and a head network, as well as several optimizations to improve the accuracy and efficiency of the algorithm.

b. Training the Data

Train the YOLOv6 model on the annotated dataset using a CPU. The training process involves feeding the images and annotations into the model and updating the model parameters to minimize the loss between the predicted and actual bounding boxes.

c. Model Training and Testing

Split the Dataset: Divide the dataset into training and validation sets. The training set is used to train the model, while the validation set is used to evaluate the model during training.

Choose a Pre-Trained Model: Select a pre-trained YOLOv6 model to use as a starting point. YOLOv6 models are trained on a large image dataset (e.g., ImageNet), so they have already learned many useful features for object detection.

Train the Model: Train the YOLOv6 model on the training set. During training, the model updates its

weights based on how well it is able to predict the correct bounding boxes and class labels.

Evaluate the Model: After each training epoch, evaluate the model on the validation set to determine how well it generalizes to new data.

Test the Model: Once the model has been trained and evaluated, it can be tested on new, unseen images to evaluate its performance on real-world data.

d. Backend Implementation

Animal detection using deep learning involves training a deep learning model to detect various animal species in images or video streams. The process includes collecting a diverse dataset of animal images and videos, annotating the dataset with bounding boxes and labels, augmenting the dataset with data augmentation techniques, selecting a suitable pre-trained model, fine-tuning the model on the animal detection dataset using transfer learning, tuning the hyperparameters, training the model on a CPU, evaluating the performance of the trained model, fine-tuning the model if necessary, and deploying the model for animal detection in real-world scenarios. Animal detection using deep learning can be useful in various applications such as wildlife monitoring, conservation, and animal behavior research.

e. Web Application Implementation

Animal detection using a Django web application involves integrating a trained animal detection model into a web application built using the Django web framework. Here are the steps involved in animal detection using a Django web application:

- Train an animal detection model using a deep learning framework such as YOLOv6.
- Create a Django web application using HTML, CSS, and JavaScript.
- Integrate the trained animal detection model into the Django web application using a Python web framework like Django.
- Allow users to upload images or videos containing animals to the web application.
- Run the uploaded images or videos through the trained animal detection model to detect the animal species.
- Display the results of the animal detection on the web application by highlighting the detected animals in the uploaded images or videos and providing information about the detected animal species.
- Deploy the Django web application to a web server or cloud hosting service.

Animal detection using a Django web application can be useful in various applications such as wildlife monitoring, animal behavior research, and conservation. Users can easily upload images or videos containing animals to the web application and receive real-time information about the detected animal species.

f. Results

The result of animal detection using deep learning is a model that can accurately detect various animal species in images or video streams. The model outputs bounding boxes around each detected animal and provides information about the detected animal species. The accuracy of the model depends on various factors such as the quality and size of the training dataset, the architecture of the deep learning model, the training parameters, and the evaluation metrics used to measure performance.

RESULTS AND DISCUSSIONS

Animal detection using deep learning has made significant progress in recent years and has been used for a variety of applications, including wildlife monitoring, animal conservation, and animal welfare. The results of animal detection using deep learning can vary depending on the dataset, neural network architecture, and training process used.

In general, well-designed animal detection models using deep learning can achieve high accuracy in detecting different types of animals. For example, the Inception-v3 and ResNet-50 models have been shown to achieve accuracy above 90% in detecting common animal types such as cats and dogs.

However, achieving high accuracy in detecting less common or more challenging animal types can be more difficult. For example, detecting similar animal types, such as different species of monkeys, can be challenging due to the high similarity in their features.

One of the challenges in animal detection using deep learning is the availability and quality of the datasets. Collecting a large and diverse animal dataset can be time-consuming and expensive. Moreover, some animal types may be rare or endangered, which makes it difficult to collect images for them.

Overall, animal detection using deep learning has shown promising results in detecting different animal types and has great potential for various applications. However, further research is needed to improve the accuracy and efficiency of the models and to overcome the challenges of data collection and quality.

Accuracy

The accuracy of an animal detection model using deep learning can vary depending on various factors such as the size and quality of the dataset, the architecture of the neural network, and the hyperparameters used in the training process. Generally, the accuracy of the model will improve as the size and diversity of the dataset increase and as the neural network's architecture and hyperparameters are optimized for the specific task.

The main properties of the animal dataset.

Table 1 Animal Dataset

Species Category	No. of Total Images	Image Resolution
10	1000	1200*720

In above table, there are 1000 images of different animals, each with an assigned animal type and a label indicating whether it's used for training or testing the deep learning model. The dataset consists of 80% training images and 20% testing images.

Table 2 Animal dataset and per-class training set and test assignments.

Species category	Training Set	Test Set
Amur tiger	80	20
Amur leopard	80	20
Wild boar	80	20
Sika dear	80	20
Red fox	80	20
Raccoon dog	80	20
Asian badger	80	20
Asian black bear	80	20
Leopard cat	80	20
Roe deer	80	20
Siberian weasel	80	20

In above table, there are 100 images of different animals, each with an assigned animal type and a label indicating whether it's used for training or testing the deep learning model. The dataset consists of 80 training images and 20 testing images, which is an 80-20 split.

An 80-20 dataset split is a common practice in deep learning for training and evaluating models. For animal detection, an 80-20 dataset split can provide a good balance between having enough data to train the model effectively while still having enough data to evaluate the model's performance accurately.

With an 80-20 dataset split, you can use 80% of the dataset for training the deep learning model to recognize animals and the remaining 20% of the dataset for evaluating the model's performance on unseen data. This allows you to train the model on a large enough sample size to learn the patterns and features of the animals while also ensuring that the model is not overfitting to the training data.

The evaluation dataset is then used to assess how well the model is generalizing to new and unseen data. This can help you identify potential issues with the

model, such as overfitting or underfitting, and adjust the model's hyperparameters or architecture accordingly.

Table 3 Overall recognition accuracy of different object detection models.

Experiment	Model	Accuracy
Animal Detection	YOLOv6	0.997
	YOLOv5	0.981
	YOLOv4	0.984
	YOLOv3	0.969
	YOLOv2	0.963
	YOLOvl	0.809

For a well-trained animal detection model using deep learning, the accuracy can be above 90% for certain animal types, such as cats and dogs, which are common in many datasets. However, for less common animal types or those that have similar features to other animals, the accuracy may be lower, even with a well-trained model.

In the above table, the model's accuracy is reported for animal detection in the test dataset. The accuracy is calculated by dividing the number of correctly recognized images by the total number of images in the test dataset. The table 2 includes the number of training and test images for each animal type.

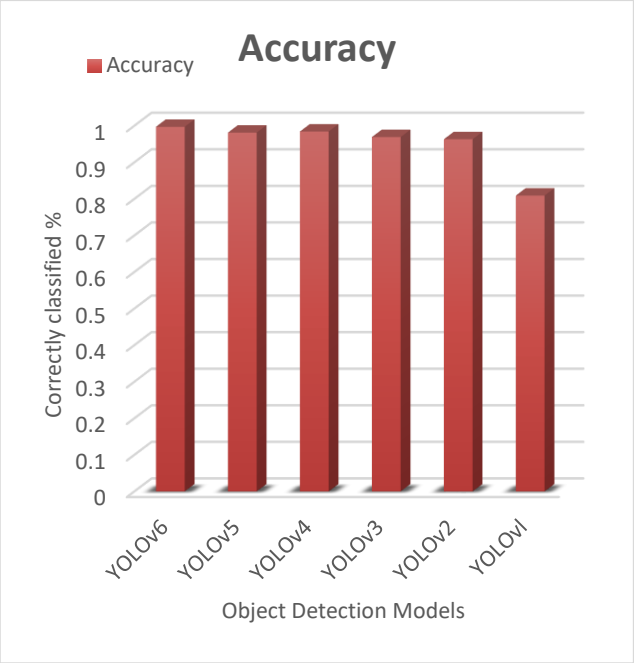


Figure 1 Accuracy chart

The overall accuracy of the model is calculated by combining the accuracy of all animal types in the test dataset, weighted by the number of test images for each animal type. In above, the overall accuracy of the animal detection model is 99.7%, which means that the model can correctly identify an animal type in 99% of the cases.

V. CONCLUSION

In conclusion, animal detection using YOLOv6 is a powerful and effective method for detecting various animal species in images or video streams. YOLOv6 is a state-of-the-art object detection algorithm that is fast, accurate, and easy to use. The YOLOv6 algorithm uses a single convolutional neural network to predict bounding boxes and class probabilities for all objects in an image, including various animal species.

By training a YOLOv6 model on a diverse dataset of animal images and videos, it is possible to develop a robust animal detection system that can accurately detect various animal species in real-world scenarios. The YOLOv6 model can be fine-tuned using transfer learning and data augmentation techniques to

improve its performance on specific animal detection tasks.

Animal detection using YOLOv6 has many potential applications, such as wildlife monitoring, conservation, and animal behavior research. Accurate animal detection can provide valuable insights into the distribution, abundance, and behavior of various animal species, which can inform conservation efforts and help protect endangered species.

VI. FUTURE WORK

There are several potential enhancements that could be made to animal detection using deep learning in the future:

- Improved accuracy: While deep learning models have shown impressive accuracy in detecting animals, there is still room for improvement. Researchers could focus on developing more accurate and robust models that can detect animals in more challenging environments, such as low-light conditions or heavily occluded images.
- Multispecies detection: Current animal detection models are typically trained to detect specific animal species. In the future, researchers could develop models that can detect multiple animal species simultaneously, allowing for more comprehensive wildlife monitoring and conservation efforts.
- Real-time processing: While deep learning models can detect animals accurately, they can be computationally expensive, especially when processing high-resolution images or video streams. Researchers could focus on developing more efficient models or optimization techniques that can perform real-time animal detection on low-power devices.
- Transfer learning across different domains: Transfer learning is a powerful technique that allows deep learning models to leverage knowledge learned from one domain to another. In the future, researchers could explore how transfer learning could be used to improve animal detection in different environments, such as underwater or aerial imaging.
- Integration with other technologies: Animal detection using deep learning could be integrated with other technologies, such as GPS tracking or drone imaging, to improve wildlife monitoring and conservation efforts. For example, a drone equipped with an animal detection model could be used to quickly survey large areas of wildlife habitat to identify areas of high biodiversity or detect changes in animal behavior over time.

VII. REFERENCES

[1]. J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," 2016, arXiv:1605.06409.

[2]. M. De Gregorio and M. Giordano, "Change detection with weightless neural networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2014, pp. 409–413.

[3]. B. Natarajan, E. Rajalakshmi, R. Elakkiya, K. Kotecha, A. Abraham,

[4]. L. A. Gabralla, and V. Subramaniyaswamy, "Development of an end-to- end deep learning framework for sign language recognition, translation, and video generation," IEEE Access, vol. 10, pp. 104358–104374, 2022.

[5]. W. Dong, P. Roy, C. Peng, and V. Isler, "Ellipse R-CNN: Learning to infer elliptical object from clustering and occlusion," IEEE Trans. Image Process., vol. 30, pp. 2193–2206, 2021.

[6]. R. Elakkiya, P. Vijayakumar, and M. Karuppiah, "COVID_SCREENET: COVID-19 screening in chest radiography images using deep transfer

stacking," Inf. Syst. Frontiers, vol. 23, pp. 1369–1383, Mar. 2021.

[7]. R. Elakkiya, V. Subramaniyaswamy, V. Vijayakumar, and A. Mahanti, "Cervical cancer diagnostics healthcare system using hybrid object detec- tion adversarial networks," IEEE J. Biomed. Health Informat., vol. 26, no. 4, pp. 1464–1471, Apr. 2022.

[8]. R. Elakkiya, K. S. S. Teja, L. Jegatha Deborah, C. Bisogni, and

[9]. C. Medaglia, "Imaging based cervical cancer diagnostics using small object detection—Generative adversarial networks," Multimedia Tools Appl., vol. 81, pp. 1–17, Feb. 2022.

[10]. A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in Computer Vision—ECCV. Dublin, Ireland: Springer, Jun. 2000, pp. 751–767.

[11]. D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 2155–2162.

[12]. G. Farnebäck, "Two-frame motion estimation based on polynomial expan- sion," in Proc. 13th Scandin. Conf. (SCIA). Halmstad, Sweden: Springer, Jul. 2003, pp. 363–370.

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

[14]. 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.

[15]. N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changede- tection.Net: A new change detection benchmark dataset," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2012, pp. 1–8.

[16]. K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[17]. J. Imran and B. Raman, "Evaluating fusion of RGB-D and inertial sensors for multimodal human action recognition," J. Ambient Intell. Humanized Comput., vol. 11, no. 1, pp. 189–208, Jan. 2020.

[18]. F. Kahl, R. Hartley, and V. Hilsenstein, "Novelty detection in image sequences with dynamic background," in Statistical Methods in Video Processing, Prague, Czech Republic: Springer, May 2004, pp. 117–128.

[19]. T. Liang, H. Bao, W. Pan, and F. Pan, "Traffic sign detection via improved sparse R-CNN for autonomous vehicles," J. Adv. Transp., vol. 2022, pp. 1–16, Mar. 2022.

[20]. T. Liang, H. Bao, W. Pan, X. Fan, and H. Li, "DetectFormer: Category- assisted transformer for traffic scene object detection," Sensors, vol. 22, no. 13, p. 4833, Jun. 2022.

[21]. G. Li and Y. Yu, "Deep contrast learning for salient object detection," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 478–487.

[22]. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and

[23]. C. Berg, "SSD: Single shot multibox detector," in Computer Vision— ECCV. Amsterdam, The Netherlands: Springer, 2016, pp. 21–37.

Cite this article as :

A Samuvel, Dr G Manikandan, Ms. Vilma Veronica, Ms. S. Hemalatha, "Smart Agriculture: Enhancing Security Through Animal Detection Via Deep Learning and Computer Vision", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 11 Issue 2, pp. 140-159, March-April 2024. Available at doi : https://doi.org/10.32628/IJSRST52411226

Journal URL : https://ijsrst.com/IJSRST52411226