TRANSFER LEARNING FOR REAL-TIME WILDLIFE INTRUSION DETECTION

A project report submitted
in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY
in
ELECTRONICS & COMMUNICATION ENGINEERING
by

S. AAKASH	**2005A42003**
V. KEDHARESHWAR RAO	**2005A42026**
S. VINDHYA	**2005A42023**
V. CHANDRASHEKAR	**2105A42L04**

Under the guidance of
Mr. S. Srinivas
Assistant Professor, Department of ECE.

Department of Electronics and Communication Engineering

SR UNIVERSITY

Ananthasagar, Warangal.

SR UNIVERSITY

CERTIFICATE

This is to certify that this project entitled **“TRANSFER LEARNING FOR REAL-TIME WILDLIFE INTRUSION DETECTION”** is the bonafied work carried out by **S. AAKASH**, **V. KEDHARESHWAR RAO, S. VINDHYA and V. CHANDRASHEKAR** as a Major project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **ELECTRONICS & COMMUNICATION ENGINEERING** during the academic year 2023-2024 under our guidance and Supervision.

Mr. S. Srinivas	**Dr. Sandip Bhattacharya**
Asst. Prof. (ECE),	Prof. & HOD (ECE),
SR University,	SR University,
Ananthasagar, Warangal.	Ananthasagar, Warangal.

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Vice Chancellor, **Prof. DEEPAK GARG,** for his continuous support and guidance to complete this project in the institute.

We would like to extend our gratitude to **Dr. Ram Raghotham Rao Deshmukh**, **Professor,** the Dean of the School of Engineering, and **Dr. Sandip Bhattacharya**, **Professor** and Head of the ECE Department, for their encouragement and support.

We owe an enormous debt of gratitude to our project guide **Mr. S. Srinivas, Asst. Prof.** as well as project co-ordinators **Mr. S. Srinivas, Asst. Prof., Dr.R. Shashank, Asst. Prof.**, for guiding us from the beginning through the end of the Major project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive, encouraging and ultimately drove us to the right direction.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

In rural areas adjacent to forested regions, the persistent threat of wild animal attacks poses significant challenges to the safety and well-being of villagers. To address this issue, this study explores the efficacy of convolutional neural networks (CNNs) in real-time wildlife intrusion detection. Drawing inspiration from existing research, including methods such as,CNN, VGG-16, ResNet-50, and YOLOv5, we adopted a transfer learning approach to enhance the ResNet50 architecture for animal recognition. Leveraging TensorFlow techniques, we augmented the base model with additional dense and dropout layers to improve its performance. Our comprehensive dataset encompassed images of six distinct animal species commonly found in wildlife habitats, enabling the model to learn intricate features and discriminate between species with high accuracy. Through rigorous training and testing, we demonstrated the robustness and generalization capability of our approach, even in challenging environmental conditions. Furthermore, we successfully deployed the trained model for real-time animal detection using live camera feeds, facilitating wildlife monitoring and conservation efforts. With a test loss of 0.1522 and an impressive accuracy of 96.19%, our system showcases the potential of deep learning techniques in wildlife management. This research represents a significant advancement in the field, offering a practical and efficient solution for mitigating human-wildlife conflicts and promoting coexistence in rural communities.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACRONYM	ABBREVIATION
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit
ReLU	Rectified Linear Activation Unit
SVM	Support Vector Machine
VGG	Visual Geometry Group
YOLO	You Only Look Once

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW OF PROJECT

A concerning increase in human-wildlife conflicts has been facilitated by the increasing encroachment of human populations onto natural ecosystems. These confrontations frequently result in property damage, injuries, and even fatalities, putting both human safety and animal welfare in jeopardy.

In Yelenahalli town, Ooty, Tamil Nadu, a bear and a leopard wandered into a residential area, alarming the locals. The wild creatures were seen on CCTV, which led the villagers to ask forest officials for help in capturing them. This event in the Nilgiris district, which is known for its abundant wildlife, brings to light the difficulties in interacting with humans and animals when leopards, drawn by easy prey, enter villages. Residents stay indoors after dark while officials investigate the matter out of fear.

There have been two reports of wildlife incursions at Tirupati temples lately. At the Srinivasa Mangapuram temple, a king cobra fell from a tower, and at the Kapila Teertham temple, which is next to a forest, leopards have been observed for the past three days. Both locations now have more security in place to stop future incidents. Due to an elephant that wandered into Siliguri from Baikunthpur Forest, the forest authorities had to issue warnings and restrictions to the surrounding villages. Using crews from two divisions, the department is attempting to force the elephant back into the forest. Not long after two other elephants were safely moved to a nature refuge, this event occurred.

When the elephants attempted to cross a railroad track and head toward the Gala area of the Terai eastern forest division, a male elephant and a calf were struck by an oncoming train. The calf, who was about eight years old, suffered serious injuries, but the male jumbo, who was about twenty years old, died. About three in the morning on Thursday, the tragedy happened close to the Lalkuan-Bareillv railway segment.

A leopard caught in a net in Gangapur village was successfully rescued by Wildlife SOS and the Maharashtra Forest Department. After the 7–8-year-old female leopard was discovered, villagers informed the authorities, and a rescue team from Wildlife SOS's Manikdoh Leopard Rescue Centre helped to guarantee its safe capture. After then, the leopard was given to the forest department.

Conventional wildlife monitoring techniques are frequently labor-intensive, time-consuming, and prone to human error because they mostly rely on manual observation and camera traps. Because of inadequate surveillance, confrontations between humans and wildlife are possible. Researchers are increasingly relying on the revolutionary potential of deep learning techniques to answer this urgent demand for a more reliable and effective wildlife infiltration detection system.

In this field, the publication "Animal Intrusion Detection Using Various Deep Learning Models" explores the effectiveness of convolutional neural networks (CNNs) in conjunction with transfer learning for real-time animal incursion detection. Their method takes the highly praised ResNet-50 architecture as a base and modifies it for the purpose of animal recognition by adding customized dense and dropout layers. Images of six different animal species that are commonly found in wildlife environments are included in their dataset. Although this research constitutes a significant addition, the authors recognize that their methodology has limits. Even with the latest deep learning models, animals that are hidden or in difficult environments might affect the accuracy of animal intrusion detection. Furthermore, uneven class distribution or insufficient variance across all classes may impose limitations on their dataset size.

Images we gathered manually are from a variety of open-source online resources such as Alamy website, which offers a vast collection of royalty-free stock photos. The collected images were then classified into seven distinct categories: Lion, Tiger, Hyena, Fox, Bear, Wolf, Calm Forest (images without any animals). The total number of images in the dataset is 2,233.

Transfer learning uses models that have already been trained to speed up training and enhance performance on new tasks. Imagine how long it would take to train a cook from scratch! Transfer learning is like teaching an experienced chef how to recognize new ingredients (later layers) for a particular meal (your new work) after they have mastered knife skills (early levels). Like that master chef, a pre-trained model such as ResNet-50's early layers encode fundamental visual elements like forms and edges. For many computer vision jobs, these general properties are helpful. We can "fine-tune" the model for a particular situation by retaining these layers and adding new ones on top.

When working with small datasets, this is significantly quicker and more effective than building a model from the start. Suppose you give the chef the duty of classifying several varieties of mushrooms (your new wildlife). Instead of having to teach them how to use a knife again, you might concentrate on the unique characteristics that set each mushroom apart. This is made possible through transfer learning, which increases the effectiveness and accessibility of deep learning for a range of applications.

Building on this framework, the work presented here aims to improve and boost the efficiency of deep learning-based real-time animal infiltration detection. Using convolutional neural network architecture and transfer learning, our study takes a similar approach. To obtain better performance, however, and to address the weaknesses noted in the original paper, we incorporate numerous significant improvements. First, to improve the model's capacity to distinguish between a broader variety of wildlife, we broadened the dataset to include seven animal species.

Furthermore, to reduce background distractions and concentrate on the animals themselves, we conduct a more thorough data preprocessing step that includes methods like central cropping. Enhancing the model's capacity to extract important information from the photos is the goal of this preprocessing stage. Thirdly, we employ an altered version of the ResNet-50 architecture, adding more information about its components, feature extraction process, and the rationale behind design decisions. This thorough justification enables scholars to not only duplicate our methodology but also perhaps modify and enhance it for future initiatives.

Our new contributions extend beyond these initial gains. We explore the details of the ResNet-50 architecture, elucidating ideas such as the feature extraction pipeline, identification and convolutional blocks, and residual connections. This thorough investigation provides a better knowledge of how the model derives meaningful representations from the unprocessed visual input, which eventually results in accurate animal classification. In addition, we offer an in-depth description of each step in the model's operation, including the initial picture input, convolutional layers, batch normalization, zero padding, max pooling, ReLU activation functions, and fully connected layers with softmax activation for the final classification. This detailed explanation provides valuable insight on the inner workings of the model.

Our study aims to develop a more resilient, effective, and flexible real-time animal invasion detection system by fusing these developments with a carefully selected and preprocessed dataset. By enabling proactive steps to be implemented in the event of wildlife intrusions and by delivering real-time notifications, this technology has the potential to protect both human lives and animal populations. Our goal is to support efficient wildlife monitoring and conservation efforts so that people and wildlife can live in harmony.

The transfer learning is used to speed up the creation of a real-time animal intrusion detection system by using a pre-trained ResNet-50 model from the Keras package within TensorFlow. The vast ImageNet dataset, which includes hundreds of object classifications, served as the original training ground for these layers. The lower-level properties that ResNet-50 acquired to recognize edges, forms, and textures are broadly relevant to other computer vision tasks, even though ImageNet does not explicitly include photos of wildlife.

CHAPTER 2
LITERATURE SURVEY

Human-wildlife conflicts pose pressing challenges in rural areas proximal to forested regions, endangering the safety and well-being of local communities. To address this issue, recent studies have explored the application of advanced technologies, particularly convolutional neural networks (CNNs), in real-time wildlife intrusion detection. Drawing on the foundations laid by prior research, which leveraged renowned methodologies including VGG-16, ResNet-50, and YOLOv5, researchers have embarked on a journey to enhance the ResNet50 architecture through transfer learning techniques. By integrating TensorFlow methodologies, these endeavors aim to fortify the base model with supplementary dense and dropout layers, thereby augmenting its efficacy for animal recognition tasks. Central to this advancement is the utilization of comprehensive datasets comprising images of diverse animal species commonly encountered in wildlife habitats. Through meticulous training and rigorous testing regimes, researchers have demonstrated the resilience and adaptability of their proposed methodologies, showcasing their capacity to discern intricate features even amidst challenging environmental conditions. Furthermore, these efforts culminated in the successful deployment of trained models for real-time animal detection via live camera feeds, significantly bolstering wildlife monitoring and conservation initiatives.

In this paper [1], the authors propose a method for animal intrusion detection utilizing various deep learning models. They experiment with convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their combinations to classify images captured by outdoor cameras into different categories of animal intrusions. The models achieve promising results in accurately detecting various types of animals, such as deer, bears, and raccoons, thereby enhancing wildlife monitoring and conservation efforts. However, the paper acknowledges several limitations, including challenges in handling diverse environmental conditions, such as varying lighting and weather, as well as the need for large amounts of labeled training data to effectively train the deep learning models. Additionally, the models may struggle with detecting rare or previously unseen animal species, highlighting the importance of continuous model refinement and adaptation to evolving wildlife contexts.

In this paper, the authors propose an Intrusion Detection System (IDS) that leverages a Deep Convolutional Neural Network (DCNN) architecture coupled with Twilio for real-time alerting. The system processes network traffic data through the DCNN, which learns to distinguish between normal and malicious activity patterns. When suspicious activity is detected, alerts are sent via Twilio, enabling timely response to potential security breaches [2]. While the approach shows promise in automating intrusion detection and notification processes, its effectiveness may be influenced by the quality and diversity of the training data, the adaptability of the DCNN model to evolving attack techniques, and the reliability and latency of the Twilio messaging service in real-world scenarios. Additionally, the scalability and resource requirements of deploying such a system in large-scale network environments should be considered.

In this paper, the authors propose a method for detecting animal intrusion using Convolutional Neural Networks (CNNs) and image processing techniques. They utilize a dataset of images containing instances of animal intrusion and employ a CNN architecture to automatically classify these images. The approach involves preprocessing steps such as image segmentation and feature extraction to enhance the detection accuracy. While the results demonstrate promising performance in identifying animal intrusion, there are some limitations to consider. The effectiveness of the method may be influenced by factors such as variations in lighting conditions, environmental clutter, and the diversity of animal species. Additionally, the generalizability of the model to different locations and scenarios may require further investigation and validation [3]. Overall, while this approach presents a viable solution for detecting animal intrusion, continued research is needed to address its limitations and enhance its practical applicability.

In this paper, the authors propose a CNN-based Wildlife Intrusion Detection and Alert System aimed at mitigating human-wildlife conflicts in regions where such encounters pose a threat to both human populations and wildlife conservation efforts. The system utilizes convolutional neural networks (CNNs) to analyze surveillance footage and detect instances of wildlife intrusion into human settlements or protected areas [4]. Through a combination of real-time monitoring and automated alert generation, the system aims to provide timely warnings to authorities or local communities, enabling proactive measures to prevent conflicts and minimize potential harm to both humans and wildlife. However, the effectiveness of the system may be limited by factors such as variable environmental conditions, the presence of camouflage or occlusion in footage, and the need for continuous training to adapt to evolving wildlife behaviors and detection challenges. Additionally, the deployment of such a system may require substantial infrastructure investment and ongoing maintenance to ensure reliable operation in remote or rugged terrain.

In this paper, the authors propose an alerting system utilizing YOLOv3, a real-time object detection algorithm, to identify and monitor animals encroaching upon agricultural lands. The system aims to mitigate crop damage and human-wildlife conflicts by sending SMS alerts to landowners and forest officials upon animal detection. While this approach presents promising benefits such as increased crop yields and enhanced protection against wildlife threats, it also possesses limitations. One limitation is the potential for false positives or missed detections, which could result in unnecessary alerts or overlooked intrusions. Additionally, the effectiveness of the system may vary depending on factors like environmental conditions and the behavior of different animal species. Moreover, the implementation of such a system may require significant infrastructure and resources, which could pose challenges for adoption in certain regions or by small-scale farmers [5].

In this paper, the authors propose a computational approach for detecting and deterring wildlife intrusion into agricultural areas using deep learning and transfer learning techniques. They employ camera surveillance to capture images of the surrounding environment and monitor the farm regularly. Utilizing advanced machine learning models, they aim to differentiate between various species of animals and their developmental stages, deploying appropriate auditory cues to deter them from damaging crops. The study acknowledges the utilization of convolutional neural network libraries and innovative methodologies in model development [6]. However, limitations may arise concerning the effectiveness of the proposed approach in real-world scenarios, as the accuracy of species identification and the practicality of deterring animals solely through sound cues may vary based on environmental factors, such as ambient noise levels and the behavior of different wildlife species. Additionally, the scalability and cost-effectiveness of implementing this surveillance system across large agricultural areas may pose logistical challenges.

In this paper, the abstract highlights the revolutionary development of Smart Crop Guardian, an agricultural security system designed to counter wild animal invasions using cutting-edge sensors and algorithms. It emphasizes the system's ability to detect and track wildlife hazards in real time, thereby enabling prompt intervention and reducing crop damage [7]. The core technology of Smart Crop Guardian lies in its intelligent image-based skills, which distinguish between environmental conditions and real threats, achieving high accuracy through high-resolution cameras and sophisticated algorithms. It also emphasizes the system's versatility, cost-effectiveness, user-friendly interface, and promotion of sustainable farming practices. However, limitations may include potential challenges in accurately distinguishing between harmless environmental factors and actual threats, as well as the need for continuous updates and maintenance to ensure optimal performance in various agricultural contexts. Additionally, while the system minimizes environmental impact compared to conventional deterrents, its effectiveness may still depend on factors such as local wildlife behavior and landscape characteristics.

In this paper [8], the authors propose an IoT-based detection system aimed at reducing animal-related accidents in forest areas by identifying animals and alerting vehicles to slow down. Utilizing a Raspberry Pi 3 Model B equipped with a camera, the system captures images and detects animal movement, employing the YOLO algorithm for real-time object detection at a rate of 45 frames per second. Upon detection, the system triggers alerts to drivers via display or sound signals, prompting them to reduce speed to avoid accidents. While the approach appears promising in mitigating wildlife collisions, several limitations need consideration. Firstly, the effectiveness of the system may vary depending on factors such as weather conditions, lighting, and the size and speed of the animals. Additionally, the reliance on visual detection may lead to false alarms or missed detections, especially in dense vegetation or during nighttime conditions. Moreover, the practical implementation and maintenance of such a system in remote forest areas could pose logistical challenges, including power supply, network connectivity, and system robustness against environmental factors and wildlife interference. Addressing these limitations is crucial for the successful deployment and efficacy of the proposed system in real-world scenarios.

In this paper, the authors highlight the substantial impact of agriculture on India's economy while acknowledging the challenges posed by animal-related issues such as crop damage, livestock attacks, and disease transmission. They propose a solution leveraging deep learning advancements to enhance farm security through autonomous animal intrusion detection. Their unique intrusion detection model, based on MobileNetV2 architecture and employing architectural augmentation and transfer learning, demonstrates promising results in accurately identifying intruders across diverse image categories. The model's construction involves effective optimization techniques and incorporates a data generator and custom layers to bolster performance [9]. While this approach shows potential for improving farm security, its limitations may include scalability issues in real-world deployment, reliance on quality and quantity of training data, and challenges in adapting to dynamic agricultural environments and varying wildlife behaviors. Additionally, the feasibility and cost-effectiveness of implementing such advanced technology in resource-constrained agricultural settings warrant further investigation.

In this paper, the authors propose a system utilizing Internet of Things (IoT) and Wi-Fi based wireless microcontroller units to identify intrusion of wild animals in agricultural farms, aiming to prevent casualties in areas with high human-wildlife interaction. The system involves placing pillars equipped with various sensors and electronic components at the corners of the field, which upon detecting intrusion, transmit alert messages via Wi-Fi to a forest officer [10]. The prototype is developed using Energia IDE, and a Python server is employed for alerting the officer. However, while the proposed system presents a novel approach to wildlife intrusion detection, several limitations should be considered. Challenges such as false alarms due to environmental factors or non-target movements, limited range of Wi-Fi transmission, and the need for continuous maintenance and power supply for the electronic units may impact the system's effectiveness and practicality in real-world deployment. Additionally, the scalability and adaptability of the system to different agricultural landscapes and wildlife behaviors warrant further investigation for comprehensive implementation.

In this paper [11], a system leveraging IoT sensors, cloud technology, and image processing techniques is proposed to address the pervasive issue of human-animal conflicts in forest and agricultural areas. The system employs ESP32 camera modules with Wi-Fi capabilities to capture live footage, which is then transmitted to a remote server for real-time processing using a YOLOv5-based pre-trained algorithm for object detection, achieving a high accuracy rate of around 98% after extensive training. Upon detection of animal intrusion, alerts are promptly sent to farmers or relevant authorities, and speakers can be activated to deter animals. However, the system's efficacy may be limited by factors such as connectivity issues in remote areas, potential false positives or negatives in detection accuracy, and the need for continuous monitoring and maintenance of hardware components and algorithm updates to adapt to varying environmental conditions and animal behaviors. Additionally, the scalability and cost-effectiveness of deploying such a system across large geographical areas remain to be thoroughly evaluated.

In this paper [12], the authors present a model leveraging deep convolutional neural networks (CNNs) for detecting animals in agricultural areas adjacent to forests. The model utilizes various features extracted from segmented images of animals, including color, Gabor, and Local Binary Patterns (LBP), and explores the combination of these features to enhance detection and classification performance. Detection is achieved through CNNs and symbolic classifiers. To validate the proposed algorithmic models, the authors created animal image and video datasets due to the lack of large benchmarking datasets.

Experimental results demonstrate improved detection accuracy, even with partial animal images. The system aims to enhance accuracy in terms of F1 score, precision, and recall, thereby contributing to the protection of agriculture, farmland, wildlife, and human safety. However, limitations may include challenges in real-world implementation such as variations in environmental conditions, scalability issues with large-scale deployment, and potential biases in the created datasets affecting generalization to diverse settings and species.

In this paper, a novel approach utilizing a wireless sensor network based on Ultra-Wide Band (UWB) technology is proposed for effectively addressing human-animal conflict. By leveraging the distinctive characteristics of UWB signals, a convolutional neural network (CNN) is employed to automatically learn and analyze these signals, followed by classification using SVM or Softmax classifiers to distinguish between human and animal intrusions. The experimental validation conducted in corn fields demonstrates a significant improvement in detection accuracy, achieving nearly a 16% enhancement compared to traditional manual extraction methods. However, limitations exist, including potential challenges in scaling up the deployment of wireless sensor networks, variations in environmental conditions impacting signal reliability, and the need for continuous refinement and optimization of the neural network models to adapt to diverse real-world scenarios and species behaviors [13].

In this paper, a transfer learning classification method for animal image recognition is proposed, utilizing a pre-trained ResNet18 network and fine-tuning its fully connected layer on the animal-10 dataset from Kaggle, comprising eight different animals. The approach achieves a notable improvement in classification accuracy, with a best-performing single animal accuracy of 97% and an overall accuracy of 92%. This surpasses models trained without transfer learning both in accuracy and fitting speed. However, while the results are promising, there are several limitations to consider. Firstly, the generalizability of the approach beyond the specific animal dataset used needs further investigation, as performance could vary with different datasets or classes of animals. Additionally, the computational requirements of fine-tuning a pre-trained network may still pose challenges in resource-constrained environments despite the improved efficiency. Further exploration of the method's robustness across diverse datasets and its scalability to real-world applications would be beneficial [14].

In this paper, the authors propose employing artificial intelligence (AI), specifically a deep learning model within the realm of computer vision, to mitigate crop damage caused by animal attacks, particularly those from monsters in India. They advocate for using a camera system to monitor the ranch, capturing daily images to detect animal presence and employing sound cues to deter them. The paper emphasizes the importance of protecting both humans and animals while maintaining crop yields. However, while the proposed AI system shows promise in identifying and deterring animals, several limitations should be noted. Firstly, the effectiveness of sound cues in deterring animals needs empirical validation, as different species may respond differently. Secondly, the scalability and cost-effectiveness of implementing such a system across large agricultural areas need further exploration. Additionally, ensuring the accuracy and reliability of the AI model in distinguishing between harmless and harmful animals is crucial to avoid false alarms and unnecessary disturbances [15].

In this paper, the authors propose a novel approach to mitigate animal attacks in human-populated areas by utilizing deep learning techniques, specifically the YOLOv5 object detection model, to detect the presence of wild animals in images. Upon detection, the system assesses the animals' intentions and emits varying frequencies of sound tailored to each species to deter them from human settlements, while also sending alerts to residents. While this method shows promise in addressing the issue of wildlife encroachment and reducing human-wildlife conflicts, it has several limitations. Firstly, the effectiveness of sound deterrents may vary depending on the species and their response to auditory stimuli. Additionally, the reliance on computer vision algorithms for accurate animal detection may be hindered by environmental factors [16], such as poor lighting or obstructed views, leading to false positives or missed detections. Moreover, the system's ability to accurately assess animal intentions based on visual cues alone may be limited, as animal behavior can be influenced by various factors beyond what can be captured in images. Therefore, while this approach offers a potential solution to mitigate animal attacks, further research and field testing are needed to validate its effectiveness and address its limitations in real-world scenarios.

In this paper [17], the authors propose WildlifeNet, a two-stage Deep Neural Network designed for low power and low memory consumption, aimed at automatically detecting specific wildlife to prevent wildlife-human conflicts. The first stage employs MobileNet for object detection, followed by a custom Convolutional Neural Network for species classification. Utilizing images from surveillance or low-cost cameras placed along animal paths, WildlifeNet aims to accurately alert nearby residents via mobile phones. The system demonstrates high accuracy in detecting coyotes, utilizing a large dataset for training. However, limitations may arise in the generalizability of the model to different wildlife species and environments, potential challenges in real-world deployment such as varying lighting conditions, occlusions, and the need for continual model refinement and updates to maintain accuracy over time. Additionally, concerns regarding privacy implications and community acceptance of constant surveillance should be addressed for successful implementation.

In this paper, a prototype for animal intrusion detection in crop fields is presented, utilizing a modified CNN algorithm integrated with a system comprising a PIR sensor, Thermal Imaging camera, GSM module, and hologram connected to a Raspberry Pi module. The system aims to efficiently detect animal intrusion and provide timely alerts to farmers, enabling them to avert potential crop damage without causing harm to the animals. While offering a promising solution to the persistent problem of protecting crops from wildlife intrusion, the paper's limitations include the need for further validation of the algorithm's accuracy across various environmental conditions and the practical challenges associated with implementing such a system on a large scale, including cost, maintenance, and compatibility with existing farm infrastructure. Additionally, considerations regarding the potential impact on wildlife behavior and ethical implications of deterring animals from crop fields should be addressed [18].

In this paper, the authors present a hybrid algorithm combining YOLO v5 with CNN to classify animal images in human habitats, aiming to mitigate human-wildlife conflicts. The proposed system first identifies whether an animal is present in a human environment and then accurately categorizes the detected animals using CNN. Experimental results demonstrate a promising 92.5% accuracy in classifying animals, indicating the potential effectiveness of the model in reducing the risks associated with animal incursions into human territory. However, despite its advancements, the paper does not thoroughly address certain limitations. Firstly, the generalizability of the model to diverse environmental conditions and species needs further validation, as its performance might vary across different regions and wildlife contexts. Additionally, the practical implementation of the algorithm, especially in real-time scenarios or large-scale deployments, warrants investigation to assess its feasibility and scalability [19]. Moreover, the ethical implications and potential biases in the dataset used for training and testing should be carefully considered to ensure fair and unbiased outcomes in wildlife management efforts.

In this paper, a novel approach combining image processing and machine learning is proposed to address the persistent issue of animal intrusion in agricultural fields, particularly problematic during the monsoon season when farmers strive to optimize yield [20]. The method utilizes the Watershed algorithm for image segmentation, followed by feature extraction using a 2D Gabor filter bank, and classification through the Support Vector Machines algorithm to identify animals posing a threat and promptly alert the farmer. The study incrementally expands the training set to determine the optimal combination of test images and filter bank parameters, aiming to enhance model efficiency compared to existing methods. However, despite its innovative approach, this paper may face limitations in real-world implementation, such as the variability of environmental conditions affecting image quality, the need for extensive computational resources for processing large datasets, and potential challenges in accurately distinguishing between various animal types and non-threatening entities. Moreover, the scalability and generalizability of the proposed model across different agricultural landscapes and animal species remain to be validated.

Table.2.1 Limitations of existing methods

Authors	Methods	Limitations
M. Praveen, K. Viswavardhan Reddy	CNN, VGG-16, ResNet-50, YOLO-v5	1. Choosing the optimal deep learning model architecture can be complex. 2. Overfitting to the training data can occur if the dataset is not large enough. 3. Computational resources needed to train and compare numerous models can be significant accuracy 90%, 95%, and 60%,93-95%
M. Naveenkumar, R. Manoj, B. Nandhakumar, R. Rahul	DenseNet201, CNN	1. Might require significant training data for specific animal types. 2. Generalization to unseen animal variations could be limited. 3. Repellent effectiveness might depend on weather conditions or animal habituation.
X. Wenling, J. Ting and S. Jiong	Convolutional Neural Network, SVM	1. Labelling a large dataset of images with animals can be time-consuming and expensive. 2. Biases in the training data could lead to misidentification of certain animals. 3. Night-vision or low-light conditions might challenge detection accuracy.

M.Hu, F. You	ResNet18	1. Kaggle data might not represent real-world variations. 2. Simple preprocessing might not address all image issues. 3. SMS alerts might not be the most robust solution.
T. Vidhyalatha, Y. Sreeram, E. Purushotham	CNN, VGG-16, VGG-19, Mobile-Net	1. The success of transfer learning hinges on the relevance of the pre-trained model's task to animal detection. 2. Fine-tuning the pre-trained model for optimal animal detection accuracy might require additional data. 3. Potential for biases from the pre-trained model to be transferred.
K. Joel John, A. Aliya, B. James, T. Jerin, F. Joffin	YOLOv5	1. Privacy concerns might arise depending on the location of camera deployment. 2. Data storage and management for extensive video recordings can be challenging. 3. High power consumption for continuous real-time video recording.
B. Vivek	Wildlife-Net, Mobile-Net, Customized CNN.	1. False positives (detecting non-threatening animals) could trigger unnecessary mitigation actions. 2. Effectiveness of mitigation strategies highly dependent on animal behaviour and species. 3. Scalability of the system to cover large areas might be an issue.
S. Sheik Mohammed, T. Sheela, T. Muthumanickam	Modified CNN Algorithm	1. The specific modifications might not be publicly available, hindering transparency and reproducibility. 2. The modified algorithm's computational efficiency for real-time applications needs evaluation. 3. Potential for overfitting if the modifications lead to the model being too specific to the training data.
M. Divya, P. Hari Krishna, C. Jahnavi, P. Manasa Lalithya, J. Sheela	YOLO-v5 combined with CNN	1. A complex hybrid model might be more prone to overfitting to the training data, potentially hindering performance on unseen scenarios.

		2. Combining algorithms can increase complexity and computational demands compared to simpler approaches. 3. Optimizing the interaction and weighting of different algorithms within the hybrid system can be challenging.
S. Radhakrishnan, R. Ramanathan	Support Vector Machine with 2D Gabor Filter Bank	1. SVMs might require more data pre-processing compared to CNNs. 2. Tuning the SVM hyperparameters for optimal performance can be more challenging than CNNs. 3. Potential for lower accuracy compared to CNNs on complex image recognition tasks like animal detection.

2.1 MOTIVATION AND SCOPE OF THE WORK

The motivation behind this project stems from the critical need for effective wildlife intrusion detection systems in habitats where human-animal conflicts are prevalent. As human populations expand and encroach upon natural habitats, incidents of wildlife intrusion into human settlements have become increasingly common, posing risks to both human safety and animal conservation efforts. Traditional methods of monitoring and deterring wildlife intrusions have often been labor-intensive, time-consuming, and prone to errors.

Recognizing these challenges, our work aims to harness the power of deep learning techniques, specifically transfer learning coupled with convolutional neural networks (CNNs), to develop a robust and efficient real-time wildlife intrusion detection system. By leveraging the ResNet50 architecture, a widely acclaimed deep learning model, as our base, we seek to enhance its capabilities through the addition of dense and dropout layers tailored to the specific task of animal recognition. This approach not only capitalizes on the wealth of knowledge encoded within the pre-trained ResNet50 model but also fine-tunes it to excel in the domain of wildlife monitoring.

The scope of our work encompasses the development, training, and validation of the proposed CNN model using a diverse dataset comprising images of six different animal species commonly encountered in wildlife habitats. Through rigorous training, the model learns to extract intricate features from input images, enabling it to discern between various animal species with high accuracy. Our project's significance lies in its potential to provide an effective solution for real-time wildlife detection, even in challenging environmental conditions.

Furthermore, our system's deployment for real-time animal detection using live camera feeds demonstrates its practical utility and operational feasibility. By preprocessing incoming frames and feeding them into the network, our system achieves efficient and accurate identification of wildlife intrusions, thereby aiding in wildlife monitoring efforts and safeguarding the lives of villagers living in proximity to wildlife habitats.

2.2 PROBLEM STATEMENT

In rural areas adjacent to forested regions, villagers encounter significant challenges due to interactions with wild animals. The recurring incidents of wild animal attacks have led to a dire situation, resulting in loss of human lives and numerous injuries among the villagers. These attacks not only threaten the safety and well-being of the local population but also disrupt their livelihoods and daily activities. Despite efforts to mitigate the conflicts through various means such as fencing or patrols, the problem persists, indicating a need for more effective and sustainable solutions. Addressing this issue requires comprehensive understanding, collaboration between stakeholders, and implementation of strategies that balance human safety with wildlife conservation efforts.

CHAPTER 3
PROPOSED METHODOLOGY

3.1 INTRODUCTION

Our project uses transfer learning with a pre-trained ResNet-50 model from Keras (TensorFlow) to save time and resources. We take out of ResNet-50 the final classification layers (trained on ImageNet for general objects) and retain the first layers that identify edge and form recognition. A wide range of computer vision tasks can generally benefit from these lower-level features. We save a lot of training time by using these already trained layers as a base. Lastly, we apply new, specially made layers on top that are intended to categorize animals in wildlife camera footage. The algorithm can adjust its expertise to the new goal of detecting certain wildlife species due to this fine-tuning.

The process begins with input images that may include pictures of peaceful forests or images with a variety of wildlife species. To guarantee that the model trains effectively, these photos go through several preprocessing procedures. This might involve standardizing the pixel values to a specific range and scaling every image to a consistent size. To reduce the background distractions and concentrate on the animals themselves, strategies like central cropping could be used.

After that, the preprocessed data is divided into three sets: test (about 10%), validation (about 20%), and training (about 70%). The separation facilitates the evaluation of the model's generalizability on unobserved data (test set) and helps prevent the model from overfitting on the training set. Researchers randomly choose pictures from each animal class and the "calm forest" category to guarantee a good mix of images within each class (variations in pose, lighting, etc.) for all sets of data while separating the data.

The model leverages a pre-trained ResNet-50 architecture, known for its effectiveness in image classification tasks. Compared to creating a model from scratch, using this pre-trained model offers a solid foundation and shortens the training period. Retrained ResNet-50 is layered with additional dense and dropout layers. These unique layers aid in optimizing the model for the objective of identifying animals within the framework of detecting wildlife invasions.

The model is trained on the training set, iteratively adjusting its internal parameters to minimize the difference between its predictions (animal species) and the actual labels provided for the training images. To prevent overfitting, the validation set is used to track the model's performance during training. When a model performs poorly on untested data and becomes overly specialized to the training set, this is known as overfitting. Lastly, the model's accuracy and generalizability are evaluated using the test set that hasn't been seen yet.

The model can be used with real-time animal intrusion detection systems after it has been trained and verified. Incoming frames from live camera feeds would be preprocessed by the system before being fed into the network that has been trained. The model would determine whether an animal is there and, if so, what species it is by examining the preprocessed frames. This data can be utilized to decrease conflicts between humans and wildlife by setting off alarms or starting preventative actions.

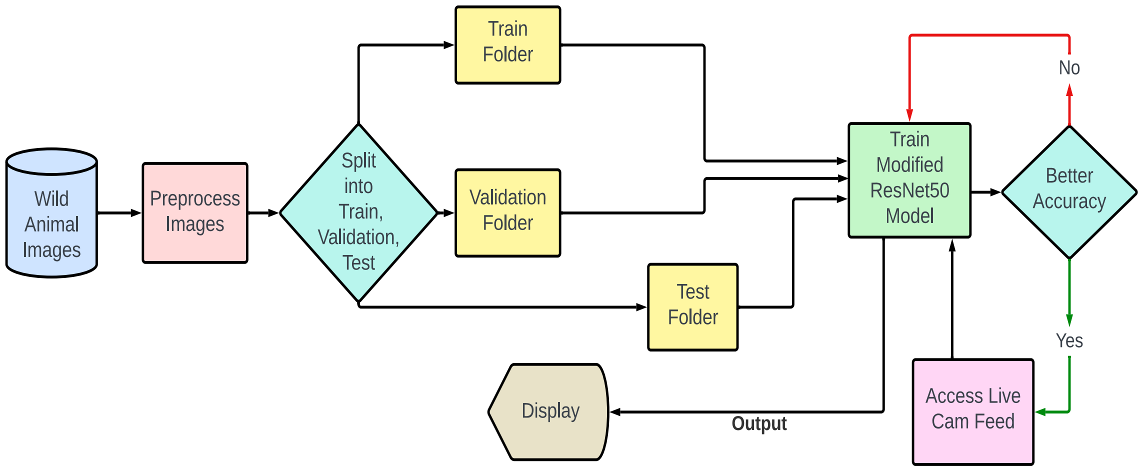

Fig.3.1 Block diagram of proposed method

3.2 DESCRIPTION

3.2.1 DATASET

The dataset used consists of images belonging to seven classes: Lion, Tiger, Hyena, Fox, Bear, Wolf, Calm Forest (images without any presence of the 6 wild animals).

- **Lion:** As shown in fig 3.2 lion is a magnificent feline with a strong physique. The striking mane, a luxuriant growth of hair surrounding the head, neck, and shoulders, makes males easily identifiable. The lion's age and biology determine the colour of its mane, which can range from blond to black. In comparison to men, females, or lionesses, are usually smaller and leaner and do not have a mane. A black tuft appears at the tip of the tail, and both sexes have a tawny golden coat. Their heads are short and round. We have collected a total of 294 images of a lion.

Fig.3.2 A Sample image of Lion

- **Tiger:** As shown in fig 3.3 a tiger cuts a striking figure. Its fierce orange fur coat is highlighted by strong black stripes that run vertically down its sides, giving its solid physique a striking appearance. It may blend in with trees and shadows while hunting prey thanks to these stripes, which have the effect of camouflage. Its belly and chest could show a flash of white. With a ruff of white fur framing its face and long, white whiskers on top, it has a powerful visage. Those sharp yellow eyes, always on the lookout, search the foliage. Its long, powerful tail aids in balance while it moves through the deep forest. We have collected 269 images of a tiger.

Fig.3.3 A Sample image of Tiger

- **Hyena:** As shown in fig 3.4 hyenas are not only forest residents, though they can occasionally be seen in forested environments, especially spotted hyenas in hilly places. If you came across one, though, it would probably have a rough coat of fur with spots or stripes that ranged in colour from brown to tan to even black, depending on the species. It would have strong legs and a broad chest, giving it an impressive frame. Its big, round head with strong jaws and a neck would be one obvious sign. Interestingly, they have a sloping back because their hindquarters are lower than their shoulders. There are a total of 305 images of a hyena.

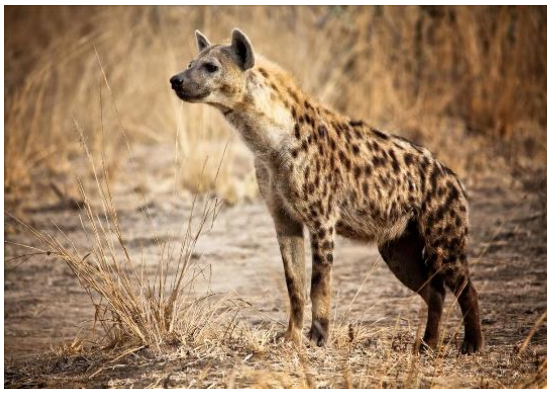

Fig.3.4 A Sample image of Hyena

- **Fox:** As shown in fig 3.5 fox would be an interesting sight in a woodland. Imagine a small to medium-sized dog, preferably black, though some foxes have grey or even reddish-brown coats. It would be slender, with legs that were agile and a bushy tail that was raised high like a plume. Its sharp snout twitching as it sniffs the air and its vigilant, triangular, pointy ears are its most remarkable traits. A total of 250 images of a fox in forest are collected.

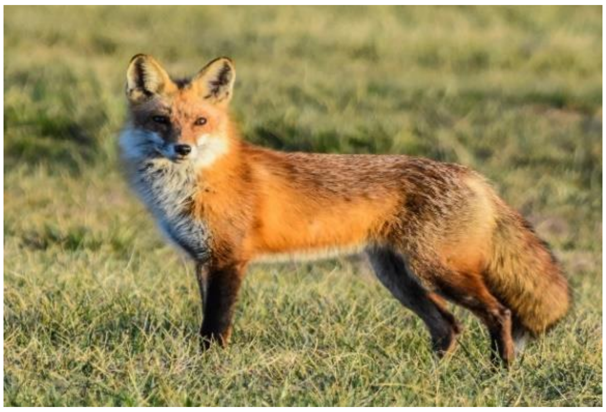

Fig.3.5 A Sample image of Fox

- **Bear:** As shown in fig 3.6 bears are big, strong animals whose colours might vary based on the species. They have thick fur. Their huge head, short neck, and muscular legs with plantigrade feet—walking on the full sole of their foot like humans—are their most distinguishing features. Their fur can be grizzly, black, brown, or even white in the case of polar bears. A bear's exact appearance in a woodland setting is reliant on the light and vegetation around it, as sunlight can dapple the bear's coat through the leaves. We have collected 408 images of a bear.

Fig.3.6 A Sample image of Hyena

- **Wolf:** As shown in fig 3.7 wolf slashes through the woods deep in the woodland. Its muzzle and chest are lighter in colour than its fur, which is a mixture of grey, brown, and black. It is kept warm in the cool woods air by its thick coat. Its muscular legs easily support its slender frame as it trots, following with a bushy, black-tipped tail. Pointed ears snap to attention, picking up on each leaf stir. Its perceptive amber eyes look around, constantly on the lookout for threats or potential prey. The wild energy of the forest is personified by this magnificent predator. We collected 263 images of a wolf.

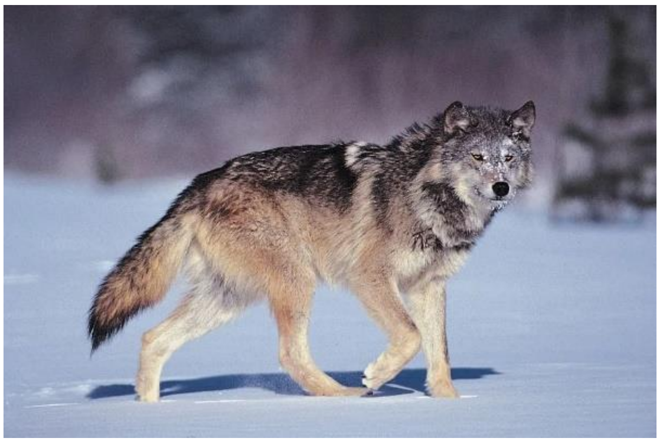

Fig.3.7 A Sample image of Wolf

- **Calm forest:** As shown in fig 3.8 calm forest conveys a sense of peace. A mosaic of light and shadow is created on the woodland floor as sunlight softly seeps through the leaves. The only sound in the silent air is the occasional rustle of leaves or the chirp of a bird hiding in cover. The atmosphere is that of a temple because to the towering trees and their thick, luxuriant foliage. The ground is covered in a lovely moss carpet that both muffles footsteps and invites investigation. Overall, one gets the feeling that this is a serene spot where anxieties disappear and nature takes centre stage. There are 444 images of a calm forest or null in the dataset we have collected.

Fig.3.8 A Sample image of Calm Forest

3.2.2 PREPROCESSING

Wildlife photos go through several preprocessing stages to guarantee the model trains efficiently. To make the model uniform, they are first all resized to a common size. The training procedure is then enhanced by standardizing the pixel values to a predetermined range. Depending on the dataset, central cropping may occasionally improve training even more by concentrating on the animals themselves and reducing background distractions. In addition to these fundamental preprocessing procedures, other elements that should be considered during preprocessing include the resolution of the source image, the color format (grayscale or RGB), and the existence of animal annotations (labels or bounding boxes) that can affect the model's performance.

3.2.3 SPLITTING THE DATA

The dataset was first shuffled to remove collection order bias before training a robust model. Then, it was divided into three sections: a 70% train set, or roughly 1560 images, from which the model was trained to learn features a 20% validation set, or roughly 443 images, from which hyperparameters were adjusted and overfitting avoided and a final 10% test set, or roughly 230 images, from which the model was not exposed during training to assess the model's generalizability on fresh data. Random selection from each animal class and the "calm forest" category was used in the splitting process to guarantee a good mix of photographs within each class (variations in pose, lighting, etc.) for all sets.

The breakdown of distribution would be:

- **Lion:** There are 205 images of lions in the train folder, 58 images in the validation folder, and 31 images in the test folder.
- **Tiger:** There are 188 images of tigers in the train folder, 53 images in the validation folder, and 28 images in the test folder for a total of 269 images.

- **Hyena:** There are 213 images of hyenas in the train folder, 61 images in the validation folder, and 31 images in the test folder for a total of 305 images.
- **Fox:** There are 175 images of foxes in the train folder, 50 images in the validation folder, and 25 images in the test folder for a total of 250 images.
- **Bear:** There are 285 images of bears in the train folder, 81 images in the validation folder, and 42 images in the test folder for a total of 408 images.
- **Wolf:** There are 184 images of wolves in the train folder, 52 images in the validation folder, and 27 images in the test folder for a total of 263 images.
- **Calm forest/Null:** There are 310 images in the train folder, 88 images in the validation folder, and 46 images in the test folder for the "null class" which likely refers to images without any animals. In total, there are 444 images for the "null class" across all three folders.

Finally, the distribution across the train, validation and test folders is as follows

- Train Set: 1560 images (7 classes)
- Validation Set: 443 images (7 classes)
- Test Set: 230 images (7 classes)

3.2.4 MODIFIED RESNET-50

ResNet-50 is a computer vision powerhouse which is a member of the Residual Neural Networks (ResNets) family of deep learning models. It was introduced in 2015 and is capable of handling complicated image identification tasks thanks to its 50-layer design, which consists of 48 convolutional layers, one MaxPool layer, and one average pool layer). ResNets use a clever idea called residual blocks, which helps them overcome the challenges posed by increasing depth, in contrast to regular neural networks. These blocks effectively create a shortcut for information flow by enabling the network to learn from the sum of the original input and the output of a hidden layer. The vanishing gradient problem, which arises in deep networks when information becomes less effective as it moves through layers, is partially resolved by this bypass method. Because of this residual learning, ResNet-50 can classify images more accurately than its predecessors. In addition, there are versions of ResNet-50 that have already been pre-trained, which means the model has already been trained on a sizable dataset such as ImageNet. This enables users to apply transfer learning, a method of applying the model's expertise to their own unique problems. Because of this, ResNet-50 is an adaptable tool for a range of computer vision applications, such as facial recognition, object detection, and even analysis of medical images.

ResNet50 utilises residual connections, a key innovation, to address the vanishing gradient problem. This allows the network to effectively train much deeper architectures, leading to improved performance in image classification tasks.

Vanishing Gradients and the Power of Residual Connections:

i. Traditional CNNs suffer from the vanishing gradient problem. As information travels through many layers, the signal from earlier layers can become vanishingly small. This makes it difficult for the network to learn complex features that depend on information from multiple layers.

ii. ResNet50 addresses this by introducing residual connections. These connections create shortcuts that allow the original input signal to bypass some layers and directly reach later ones. This ensures that the information from earlier layers is preserved and can still contribute to the final output.

iii. With residual connections, ResNet50 can effectively train much deeper architectures compared to traditional CNNs. This allows the network to learn more intricate and nuanced features from the images, leading to improved performance in image classification tasks.

Building Blocks: Identity and Convolutional Blocks - The Engine of ResNet50:

i. ResNet50 relies on two types of building blocks to construct its deep architecture:

   a. Identity Block: This block acts as a shortcut. It simply adds the input it receives directly to the output without any modification. This allows the original information to flow through the network unaltered and contribute to the result.

   b. Convolutional Block: This block performs the core computation of the network. It uses convolutional filters to extract features from the input data and then adds this processed information to the original input through another shortcut connection.

ii. By cleverly combining these two types of blocks, ResNet50 can efficiently train deeper networks while ensuring the gradients can flow back effectively during training. The identity block preserves information, while the convolutional block allows for learning new features based on the existing ones.

Feature Extraction Pipeline: From Raw Pixels to Image Classification:

i. At the heart of feature extraction lies the convolutional layer. In ResNet50, each convolutional layer applies filters to the input data, extracting specific features like edges, textures, or shapes.

ii. To improve the training process and speed, ResNet50 incorporates batch normalization after each convolutional layer. This technique helps to stabilize the learning process and allows the network to converge faster.

iii. Following the convolution and normalization, a ReLU (Rectified Linear Unit) activation function is applied. This function introduces non-linearity into the network, allowing it to learn more complex relationships between features.

iv. The extracted features are then fed into the residual blocks, where they are either directly added through the identity block or processed further by the convolutional block. This iterative process allows the network to build increasingly complex representations of the image.

v. Finally, fully connected layers analyse the high-level features extracted from the residual blocks. These layers are trained using a softmax function to output the probabilities of the image belonging to different categories. This final step performs the image classification task.

By combining these elements – residual connections, building blocks with shortcuts, and a well-defined feature extraction pipeline – ResNet50 offers a powerful architecture for image classification tasks.

Input:

The input to the model is an image. The size of the image for ResNet50 is typically 224x224 pixels with three channels (RGB).

Zero Padding:

As shown in the fig 3.9 padding adds a border of zeros around the periphery of the image. This is a common technique used in CNNs to ensure the output has the same dimensions as the input when using filters with a stride of 1.

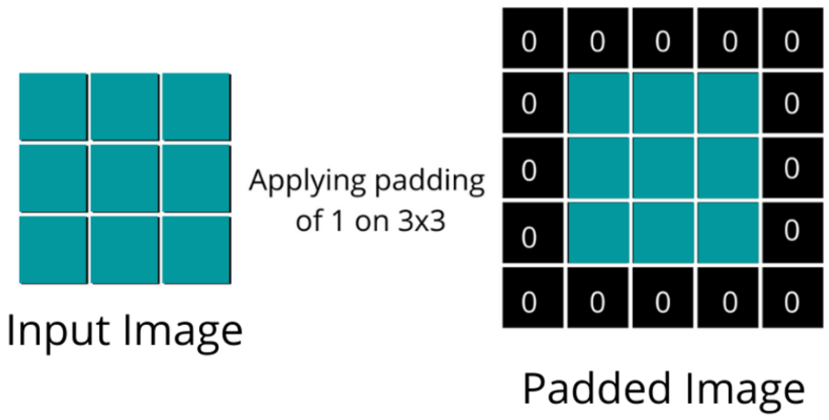

Fig.3.9 Visualization of Zero Padding [21]

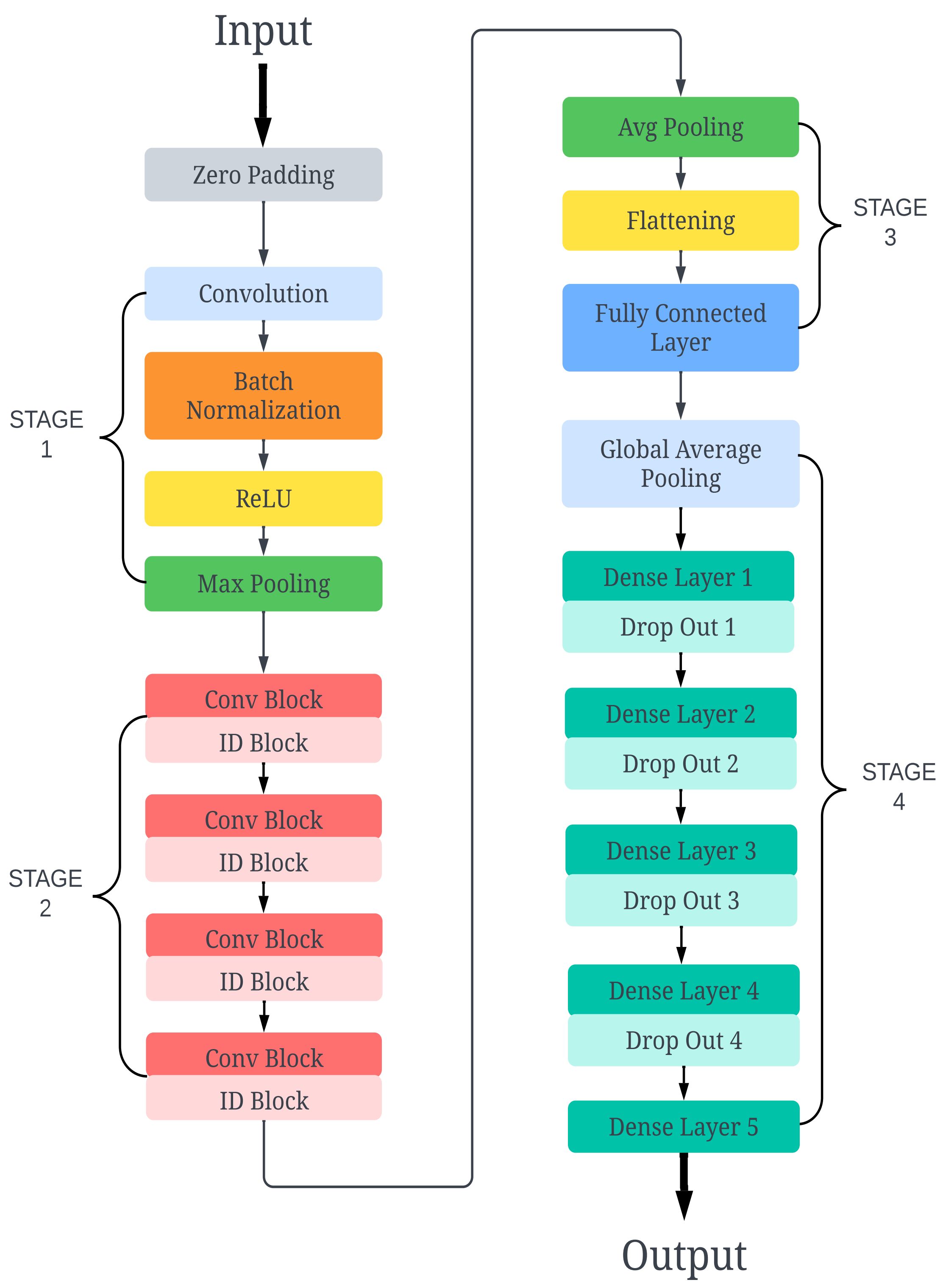

Fig.3.10 Architecture Diagram of modified ResNet-50

Stage 1:

i. **Convolution:** As shown in the fig 3.11 convolution is the core operation of a CNN. It involves applying a filter (or kernel) to the previous layer to extract features. The filter slides over the input data with a specific stride (step size), capturing spatial relationships between pixels.

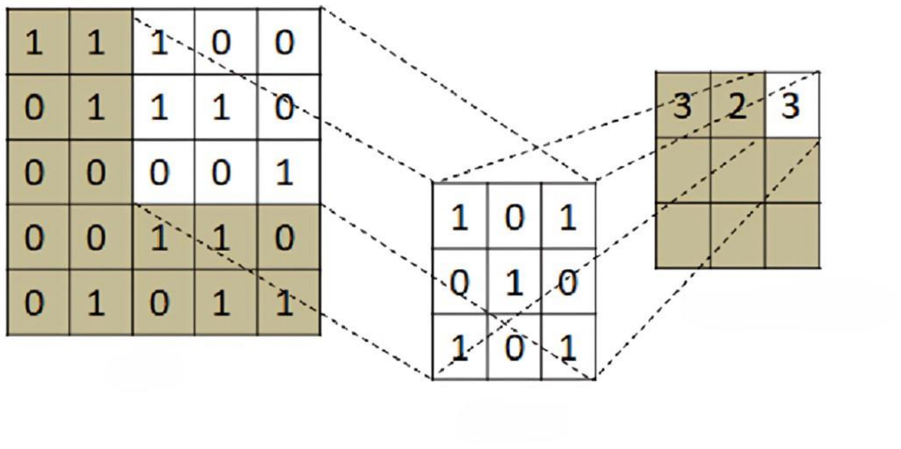

Fig.3.11 Figure depicting Convolution in Neural Network [22]

ii. **Batch Normalization:** As shown in the fig 3.12 batch normalization is a technique for training very deep neural networks. It standardizes the inputs to each layer which can improve the speed of training and address the problem of internal covariate shift.

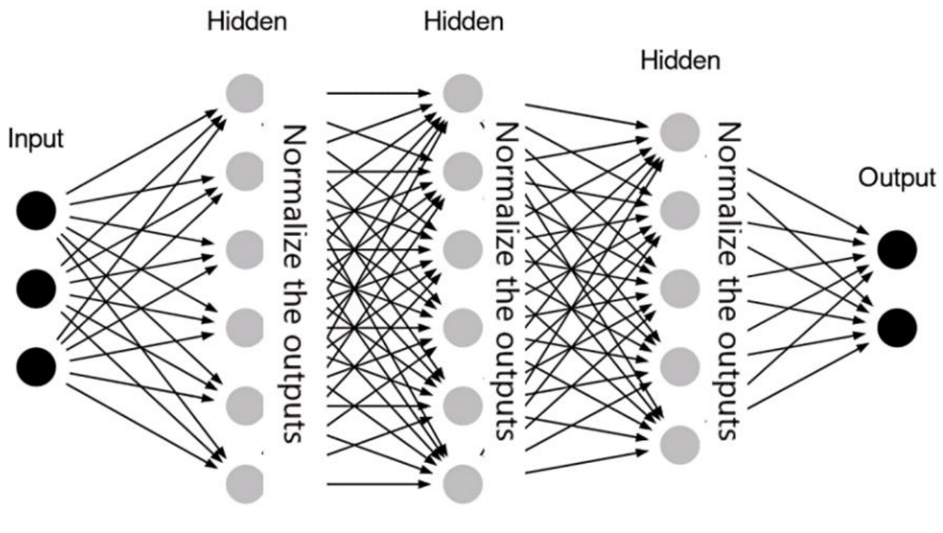

Fig.3.12 Batch normalization in Neural Network [23]

iii. **ReLU:** As shown in the fig 3.13 ReLU stands for rectified linear unit. It's a type of activation function used in CNNs. Activation functions introduce non-linearity into the network, allowing it to learn complex patterns. ReLU simply outputs the input directly if it's positive, otherwise it outputs zero.

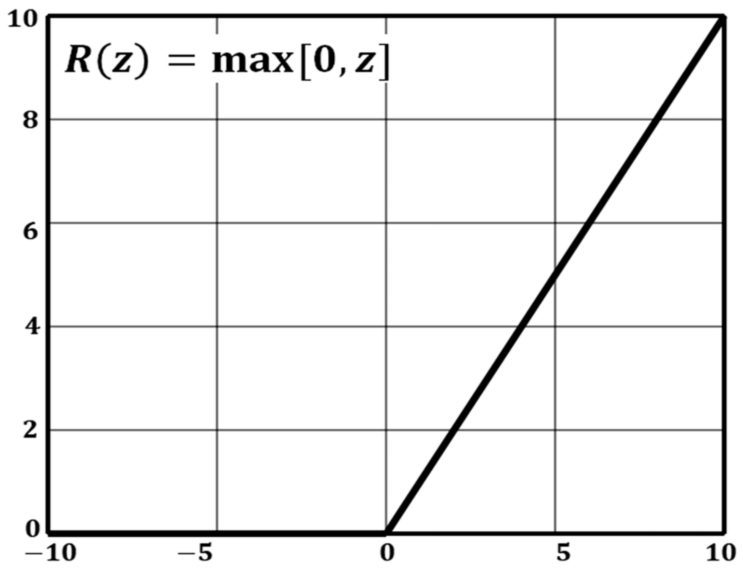

Fig.3.13 Visualization of ReLU function [24]

iv. **Max Pooling:** As shown in the fig 3.14 max pooling is a type of pooling operation that reduces the dimensionality of the data by selecting the maximum value from a subregion. This helps to control overfitting and reduces the computational cost of training the network.

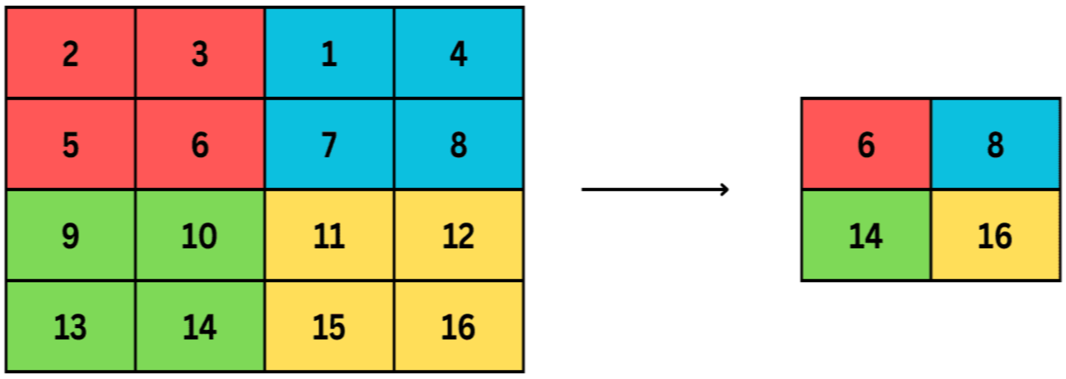

Fig.3.14 Illustration of the Max Pooling operation in Convolutional Neural Networks [25]

Stage 2:

i. **Conv Block:** As shown in the fig 3.15 these layers refer to convolutional blocks. They typically consist of a sequence of convolutional layers, batch normalization layers, and ReLU activation layers.

a. **Convolutional layer (CONV 2D):** This layer applies a convolutional filter to the input data. The convolutional filter is a small matrix of weights that is learned by the network. The convolution operation involves sliding the filter over the input data and computing the dot product of the filter weights with the input data at each position. This produces a feature map that highlights specific features in the input data.

b. **Batch normalization layer (Batch Norm):** This layer normalizes the activations of the previous layer. This helps to improve the training speed and stability of the network.

c. **Activation layer (ReLU):** This layer applies a non-linear activation function to the output of the previous layer. The ReLU (Rectified Linear Unit) activation function is a popular choice for CNNs because it is simple and efficient to compute.

The residual block also includes a shortcut connection that skips the convolutional layers and adds the input data directly to the output of the block. This shortcut connection helps to address the vanishing gradient problem, which can occur in deep neural networks.

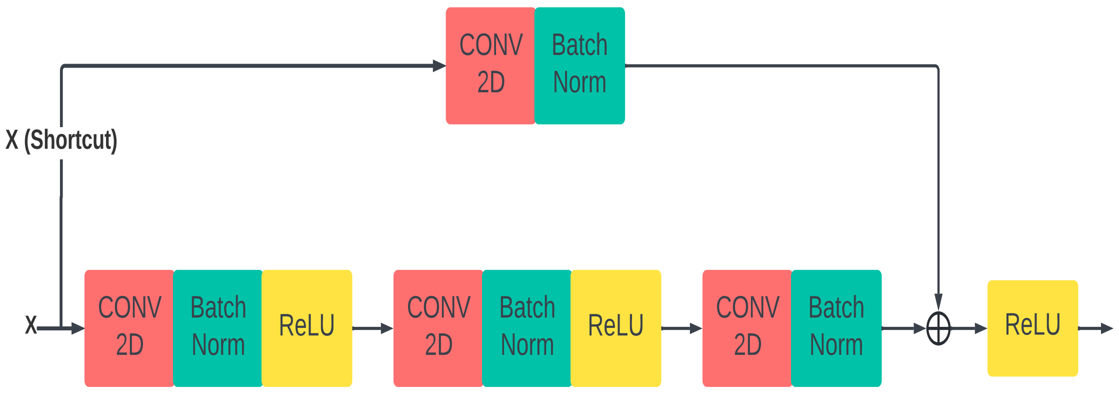

Fig.3.15 A diagram showcasing the layers and shortcut in a ResNet-50 block

ii. **ID Block:** As shown in the fig 3.16 these layers refer to identity blocks. Identity blocks are a type of residual block introduced in the ResNet architecture. They were introduced specifically to address the problem of vanishing gradients, a common problem in deep neural networks. Residual blocks allow the network to learn the identity function which can, in turn, allow for deeper networks to be trained more effectively. An identity block consists of a shortcut connection that adds the output of the earlier layer to the output of the convolutional layers within the block.

a. **Convolutional layer (CONV 2D):** This layer applies a filter, which is a small matrix of weights, to the input data. As the filter slides over the data, it computes the dot product of its weights with the data at each position. This generates a feature map emphasizing specific features within the data.

b. **Batch normalization (Batch Norm):** This layer normalizes the activations of the prior layer, which improves the training speed and stability of the network.

c. **Activation layer (ReLU):** This layer introduces non-linearity using a ReLU (Rectified Linear Unit) activation function. ReLU is popular in CNNs for its simplicity and efficiency.

iii. The defining characteristic of a residual block is the **shortcut connection**. This connection bypasses the convolutional layers and directly adds the input data to the block's output. This shortcut helps address the vanishing gradient problem, which can occur in deep neural networks. In essence, the residual block utilizes the shortcut connection to create a path for the gradient to flow directly through the block.

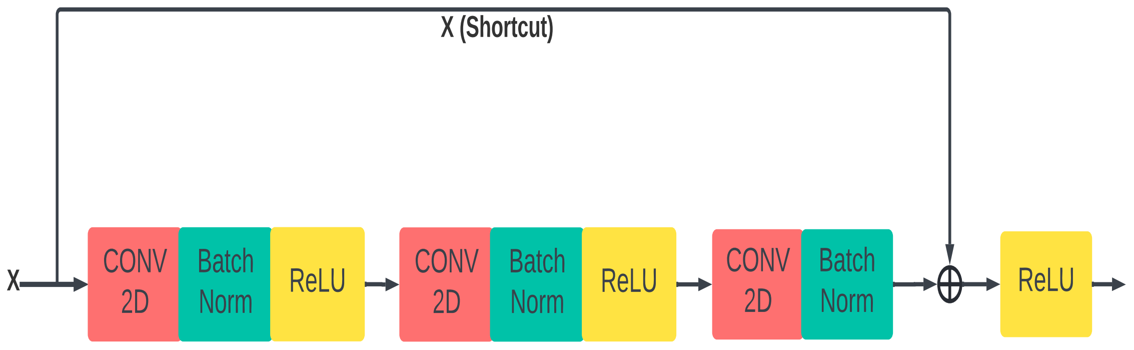

Fig.3.16 Diagram of a neural network residual block with a bypassing shortcut path

Stage 3:

i. **Avg Pooling:** As shown in the fig 3.17 global average pooling is a type of pooling operation that reduces the spatial dimensions of the data to a single value. This is often used in CNNs for image classification tasks as it captures the overall sentiment of the features extracted from the convolutional layers.

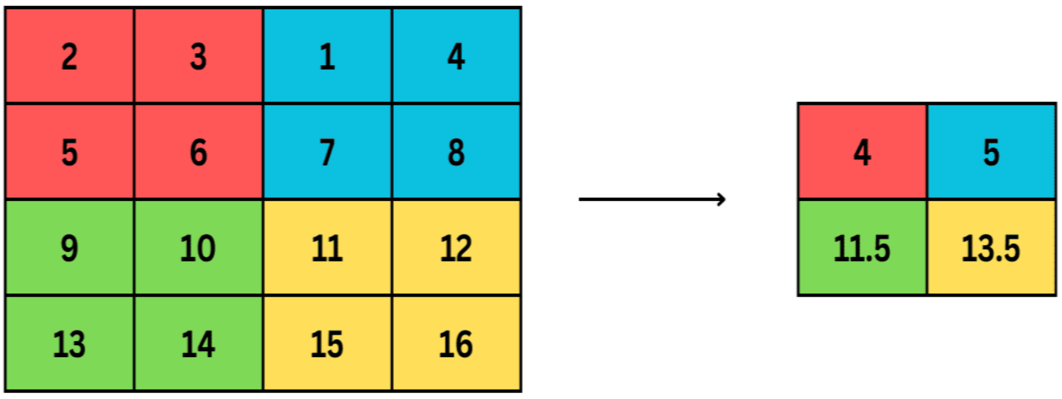

Fig.3.17 Depiction of dimensionality reduction through average pooling [26]

ii. **Flattening:** As shown in the fig 3.18 flattening reshapes the data from a multi-dimensional tensor into a one-dimensional vector. This is necessary before feeding the data into a fully connected layer.

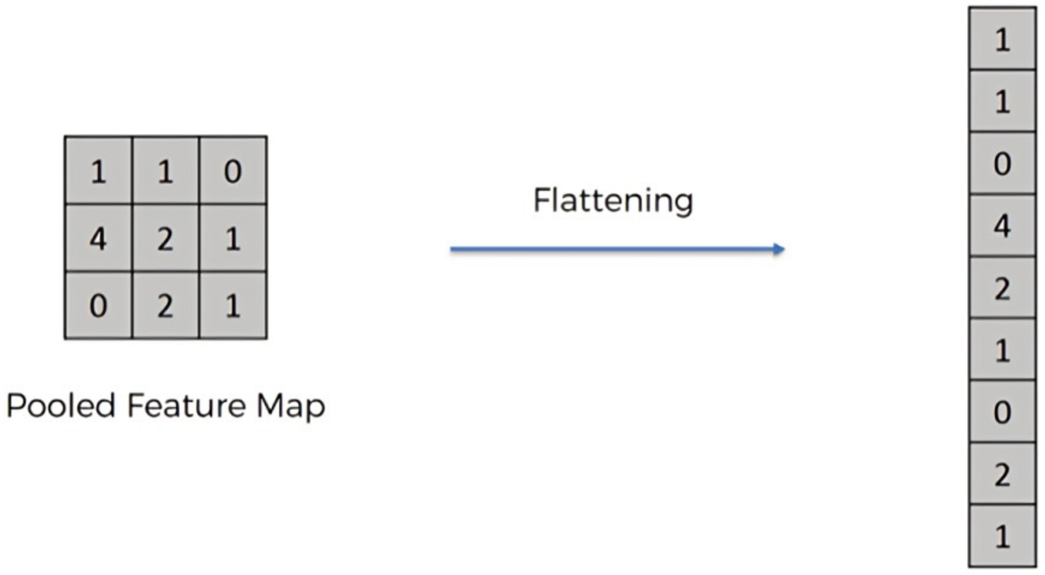

Fig.3.18 Illustration of Flattening layer in a neural network [27]

iii. **Fully Connected Layer:** As shown in the fig 3.19 this is the final layer of the network and it's responsible for image classification. It consists of several neurons equal to the number of classes the model is predicting. Each neuron outputs a score which indicates the probability of the input image belonging to a particular class.

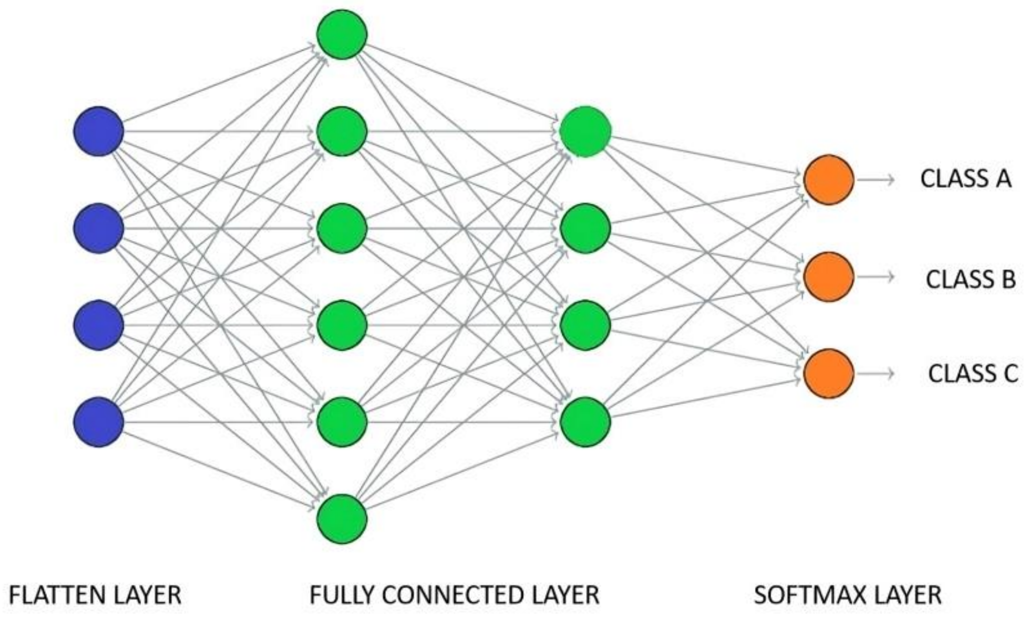

Fig.3.19 Illustration of a fully connected layer in a neural network [28]

Stage 4:

i. **Global Average Pooling (GAP):**

a. As shown in the fig 3.20 this layer, denoted by layers. GlobalAveragePooling 2D(), reduces the spatial dimensions (height and width) of the feature maps extracted by the final convolutional layers in ResNet50 to a single value for each channel. It essentially summarizes the information within each feature map.

b. This technique offers several advantages:

- ➢ Reduced Parameters: Significantly fewer parameters compared to fully-connected layers, leading to a more efficient model.
- ➢ Computational Efficiency: Faster computations due to the lower number of parameters.
- ➢ Rotation Invariance: Provides some degree of invariance to rotations in the input image.

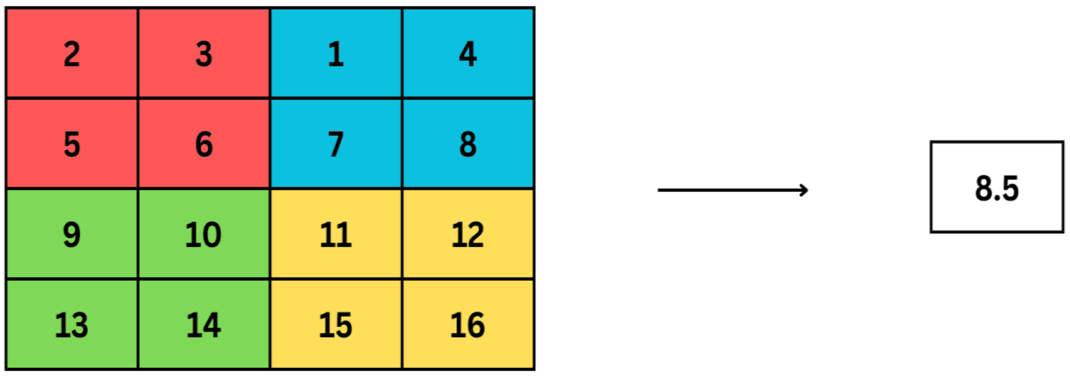

Fig.3.20 Illustration of a Global Average Polling Layer in a neural network [29]

ii. **Dense Layers with Activation and Dropout:**

a. As shown in the fig 3.21 subsequent layers are all Dense layers, also known as fully-connected layers. These layers connect each neuron in the previous layer to all the neurons in the current layer.

b. There are four Dense layers in this stage, each with a decreasing number of neurons (512, 256, 128, and 64). This creates a bottleneck architecture, gradually reducing the dimensionality of the data before feeding it to the final output layer.

c. Each Dense layer is followed by a ReLU (Rectified Linear Unit) activation function, denoted by activation='relu'. ReLU introduces non-linearity into the network, allowing it to learn complex relationships between features. It simply outputs the input directly if positive, otherwise it outputs zero.

d. Dropout layers, denoted by layers.Dropout(0.5), are inserted after each Dense layer and the ReLU activation. Dropout randomly sets a certain proportion (here, 50%) of the activations from the previous layer to zero during training. This helps to prevent overfitting by forcing the network to not rely too heavily on any specific features.

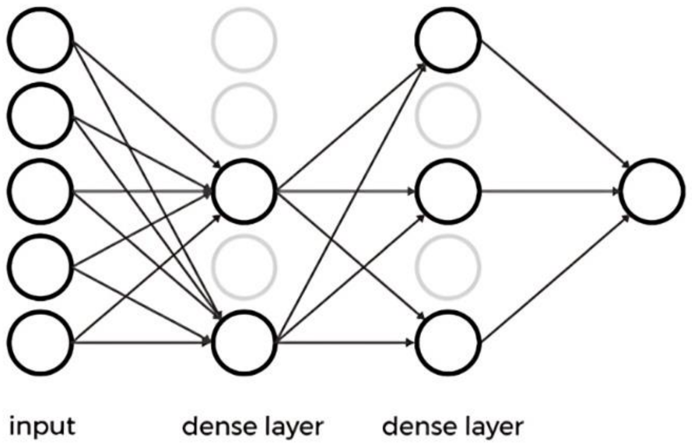

Fig.3.21 Illustration of a dense layer and dropout in a neural network [30]

iii. Final Output Layer:

a. As shown in the fig 3.22 the final layer, denoted by predictions = layers.Dense(7, activation='softmax')(x), is another Dense layer with 7 neurons (assuming you're classifying between 7 different classes).

b. A softmax activation function is applied to the output of this layer. Softmax normalizes the outputs of the layer to lie between 0 and 1, essentially representing probabilities for each class. The higher the value, the more likely the model believes the input image belongs to that class. Softmax activation function is applied to the output of the fully connected layer. It converts the raw scores into probabilities, representing the likelihood of each class.

Softmax is defines as:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{3.1}$$

Where x_i is the input value of the i^{th} element, and N is the total number of elements in the input vector.

Softmax is commonly used as the activation function for the output layer of a neural network in multi-class classification tasks, where it provides probabilities for each class.

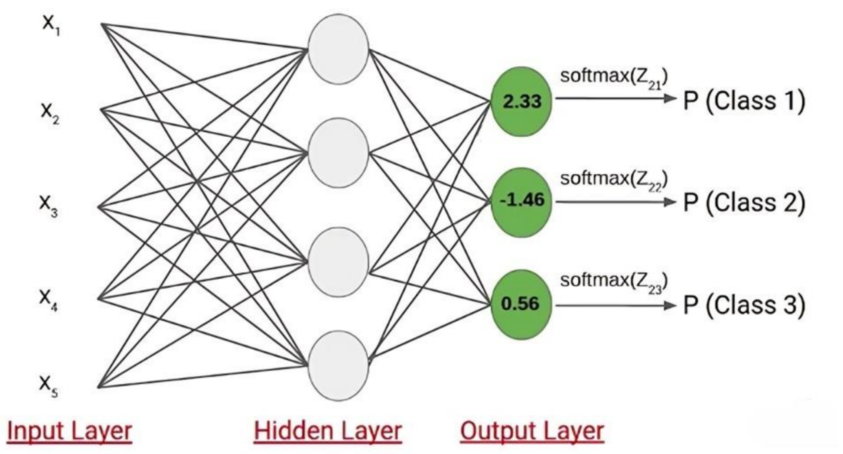

Fig.3.22 Illustration of a final layer in a neural network [31]

So, if the accuracy is good then the model is used to deploy on the live cam feed, otherwise the model is trained again and again until it achieves a better accuracy.

3.2.5 ACCESS LIVE CAM

Using a webcam, snapshots are taken in the boundaries of villages and the images are pre-processed and sent to the model as input

3.2.6 DISPLAY

The model detects the class it belongs to and displays the class name in the camera frame.

CHAPTER 4
RESULT & DISCUSSION

4.1 SYSTEM PARAMETERS

When training a TensorFlow model, especially a deep neural network like ResNet-50 for Animal Intrusion detection, several system parameters and configurations need to be considered. Here’s a breakdown:

Hardware:

- GPU: Utilizing a GPU for training can significantly speed up the process compared to using a CPU. NVIDIA GPUs are commonly used with TensorFlow due to their excellent CUDA support. Choose a GPU with sufficient VRAM (Video RAM) to accommodate the model and batch size.
- CPU: While not as fast as GPUs for training deep neural networks, CPUs can still be used for training smaller models or when GPU resources are limited.
- Memory: Ensure that your system has enough RAM to handle the dataset and model parameters.

Software:

- TensorFlow: Install TensorFlow and ensure that you have the appropriate version compatible with your GPU drivers (if using GPU). Check the TensorFlow documentation for compatibility.
- CUDA and cuDNN: If training on a GPU, make sure you have CUDA and cuDNN installed and configured correctly. These are essential for GPU acceleration in TensorFlow.
- Python Environment: Set up a Python environment with necessary packages like NumPy, Pandas, and Matplotlib, OpenCV.

4.2 RESULT ANALYSIS

The training and testing phases of our modified ResNet50 model yielded promising outcomes, underscoring its effectiveness in real-time wildlife intrusion detection. During the training phase, we employed a batch size of 32 and utilized the sparse categorical cross-entropy loss function coupled with the Adam optimizer, featuring a learning rate of 0.001. This configuration facilitated the convergence of the model's parameters towards optimal values, enabling efficient learning from the provided dataset. Throughout the training process, the model exhibited notable proficiency in learning intricate features from the dataset, comprising images of six distinct animal species commonly found in wildlife habitats. The augmentation of the base ResNet50 architecture with additional dense and dropout layers enhanced its ability to discriminate between various animal species, thus laying a robust foundation for subsequent testing. Explanation of Optimizer used in training is given below.

Adam Optimizer:

The Adam optimizer is a popular optimization algorithm commonly used in training deep learning models. It stands for "Adaptive Moment Estimation" and combines ideas from two other popular optimization methods: RMSprop and Momentum. Adam maintains adaptive learning rates for each parameter by computing individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Here's a brief overview of the algorithm and its formula:

- **Initialization:** Adam maintains two moving averages, namely the first moment (the mean) m and the second moment (the uncentered variance) v of the gradients.
- **Parameter updates:** Given a gradient g_t (the gradient of the loss function with respect to the parameters at time step t), the parameters θ are updated as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \quad (4.1)$$

Where, η is the learning rate

ϵ is a small constant to prevent division by zero (typically around 10^{-8})

t is the time step

Upon completion of training, we evaluated the model's performance using a separate testing dataset consisting of 230 images. Employing a batch size of 32 ensured efficient processing of the test samples, facilitating rapid assessment of the model's accuracy and loss metrics. The results obtained from testing revealed a remarkable accuracy of 96.19%, indicating the model's high capability to correctly classify wildlife intrusions in real-time scenarios. Furthermore, the calculated loss value of 0.1522% underscored the model's proficiency in minimizing errors during the classification process. This low loss value signifies the efficacy of the model in accurately predicting the class labels of the test samples, thereby validating its generalization capability beyond the training dataset.

The high accuracy achieved during testing demonstrates the robustness and reliability of our modified ResNet50 model in real-world applications. By successfully identifying wildlife intrusions with a high degree of accuracy, the model proves its efficacy in enhancing wildlife monitoring and conservation efforts in rural areas adjacent to forested regions. The impressive performance metrics attained in this study highlight the transformative potential of deep learning techniques in addressing human-wildlife conflicts. The utilization of advanced methodologies such as convolutional neural networks, coupled with transfer learning approaches, enables the development of highly efficient models for real-time wildlife intrusion detection. Moreover, the deployment of our trained model for real-time animal detection using live camera feeds further validates its practical utility in wildlife management scenarios. By facilitating prompt identification of wildlife intrusions, our model empowers local communities and wildlife authorities to take timely and appropriate actions to mitigate potential conflicts and ensure the safety of both humans and animals.

In conclusion, the results obtained from the training and testing of our modified ResNet50 model underscore its efficacy and reliability in real-time wildlife intrusion detection. With a remarkable accuracy of 96.19% and a minimal loss value of 0.1522%, the model demonstrates its potential to significantly enhance wildlife monitoring and conservation efforts, thereby fostering harmonious coexistence between humans and wildlife in rural areas.

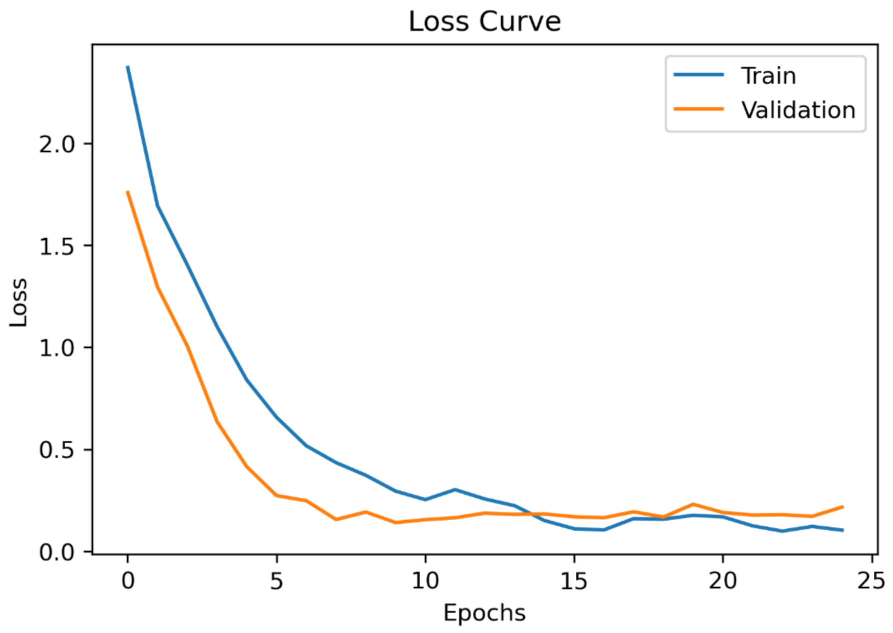

Fig.4.1 Training and validation Loss curve

As shown in fig 4.1 the loss curve appears to show a successful training process. It depicts a steady decrease in both the training loss and the validation loss over a number of epochs. This indicates that the model is learning and improving its performance on both the training data and the validation data.

Here’s a more detailed breakdown of what the curve suggests:

- **Training Loss:** The training loss curve starts at a relatively high value and steadily decreases as the number of epochs increases. This means that the model is effectively reducing the difference between its predictions and the actual values in the training data.
- **Validation Loss:** The validation loss curve also shows a decrease as the number of epochs increases. This is a good sign, as it suggests that the model is generalizing well and is not simply overfitting to the training data.

Ideally, the validation loss should continue to decrease along with the training loss, but at a slower rate. If the validation loss starts to increase while the training loss continues to decrease, it could be a sign of overfitting. Overall, the loss curve you provided suggests that the model is performing well and generalizing effectively to unseen data.

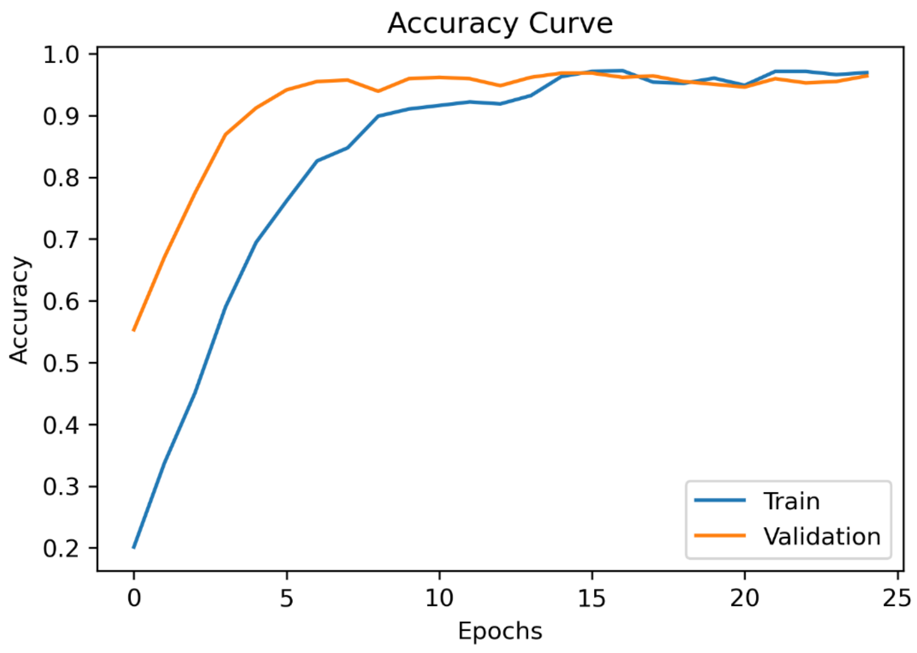

Fig.4.2 Training and validation Accuracy curve

As shown in fig 4.2 the graph shows the accuracy of a machine learning model over time, which are commonly referred to as epoch. The x-axis represents the number of epochs, and the y-axis represents the accuracy. There are two curves plotted on the graph, one for the training accuracy and one for the validation accuracy.

- **Training Accuracy:** The training accuracy curve, labelled “Train” in the graph, starts at a bit over 0.3 and increases steadily to almost 1.0 by the end of the training process. This indicates that the model is performing well on the training data, and it’s able to correctly classify most of the training instances by the end of training.

➢ **Validation Accuracy:** The validation accuracy curve, labelled “Validation” in the graph, also increases over time. However, it increases at a slower rate than the training accuracy, and it ends up at a slightly lower value than the training accuracy. This is a good sign, as it suggests that the model is generalizing well and is not simply overfitting to the training data.

In conclusion, the accuracy graph you provided suggests that the machine learning model is performing well and generalizing effectivel y to unseen data. The fact that the validation accuracy is close to the training accuracy and continues to increase throughout the training process indicates that the model is not overfitting to the training data.

Table.4.1 Accuracies Comparison

Model	**Accuracy**
ResNet-50	60%
CNN	90%
YOLOv5	93%
VGG-16	95%
Modified Resnet-50 (Proposed Method)	96.19%

CHAPTER 5
CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

This research addresses a critical challenge in wildlife management: developing a robust and real-time wildlife intrusion detection system. Conventional techniques are labor-intensive and unreliable. The suggested method makes use of convolutional neural network (CNN) architecture and transfer learning. The base model was a pre-trained ResNet-50 model with extra dense and dropout layers for animal recognition. Images of six animal species that are frequently found in wildlife settings were included in the dataset. After extensive training, the model was able to extract complex features and achieve excellent accuracy in animal species discrimination. With a test loss of 0.1522 and an astounding accuracy of 96.19%, it proved that it could efficiently learn and generalize. A learning rate of 0.001, a batch size of 32, and the Adam optimizer are some of the contributing aspects to this success. The training results are quite promising overall, showing the possibility of a transfer of learning-modified ResNet-50 for real-time detection of animal invasions.

5.2 FUTURE SCOPE

The goal of future developments for this transfer learning-based animal invasion detection system built on a modified ResNet-50 architecture could be to make it more resilient to real-world conditions. Even while the current model achieves great accuracy, by including data with differences in variables like clarity, weather, and camera angles that were not observed during training, these issues could be solved. Furthermore, methods for controlling possible class imbalances—in which certain animal classes have a lower number of training photos than others—may enhance the precision of detection. Maintaining efficiency would also require ongoing performance monitoring of the deployed system and progressive retraining with new data. Finally, by adding their photos to the training set, other animal species might be added to the system. This wildlife intrusion detection system has the potential to be an even more useful tool for managing wildlife, promoting conservation efforts, and preserving the coexistence of humans and wildlife by taking these factors into account and considering other improvements.

BIBLIOGRAPHY

[1] P. Muragod and V. R. K, “Animal Intrusion Detection Using various Deep Learning Models,” in *2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, 2022, pp. 1–12. doi: 10.1109/NKCon56289.2022.10126526.

[2] K. Akhil Joseph Xavier and G. K. Shyam, “Intrusion Detection System Using Deep Convolutional Neural Network and Twilio,” in *Advances in Cognitive Science and Communications*, A. Kumar, S. Mozar, and J. Haase, Eds., Singapore: Springer Nature Singapore, 2023, pp. 279–289.

[3] K. Bhumika, G. Radhika, and C. H. Ellaji, “Detection of animal intrusion using CNN and image processing,” *World J. Adv. Res. Rev.*, vol. 16, no. 3, pp. 767–774, 2022.

[4] G. Ravi, S. Afroz, K. Yamuna, and S. Afsha, “CNN Based Wildlife Intrusion Detection and Alert System,” *Int. Trans. Electr. Eng. Comput. Sci.*, vol. 2, no. 1, pp. 30–36, 2023, doi: 10.62760/iteecs.2.1.2023.40.

[5] K. Palai, “Wild Animal Intrusion Detection System using YOLO”.

[6] R. Saraswathi, G. Shobarani, A. Subramani, and D. Tamilarasan, “Applying Deep Learning and Transfer Learning Techniques for Improving Animal Intrusion Detection in Agriculture Farms,” in *2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, 2023, pp. 1–6. doi: 10.1109/ICDSAAI59313.2023.10452442.

[7] R. T. Ajaykarthik, R. Arunbabu, K. Aditya, G. Dineshkumar, and others, “Intelligent Image-Based Defense Against Incursion from The Wild Animals,” *Int. Res. J. Adv. Eng. Hub*, vol. 2, no. 03, pp. 355–363, 2024.

[8] J. J. Daniel Raj, C. N. Sangeetha, S. Ghorai, S. Das, Manish, and S. Ahmed, “Wild Animals Intrusion Detection for Safe Commuting in Forest Corridors using AI Techniques,” in *2023 3rd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, 2023, pp. 1–4. doi: 10.1109/ICIPTM57143.2023.10117831.

[9] E. Chandralekha, M. A. A, R. V, and M. K. Srinivasan, “Animal Intrusion Detection System using SIFT Features and Transfer Learning with MobileNetV2,” in *2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2023, pp. 1339–1344. doi: 10.1109/ICIMIA60377.2023.10426584.

[10] M. Begum H., D. A. Janeera, and A. Kumar. A.G, “Internet of Things based Wild Animal Infringement Identification, Diversion and Alert System,” in *2020 International Conference on Inventive Computation Technologies (ICICT)*, 2020, pp. 801–805. doi: 10.1109/ICICT48043.2020.9112433.

[11] L. Arulmurugan, S. Pradeep, and B. K. SP, “Animal Intrusion Detection Using ESP32 Cam and Open CV”.

[12] N. M, M. R, N. B, and R. R, “DenseNet201 for Animal detection and repellent system,” in *2022 International Conference on Electronic Systems and Intelligent Computing (ICESIC)*, 2022, pp. 213–218. doi: 10.1109/ICESIC53714.2022.9783582.

[13] W. Xue, T. Jiang, and J. Shi, “Animal intrusion detection based on convolutional neural network,” in *2017 17th International Symposium on Communications and Information Technologies (ISCIT)*, 2017, pp. 1–5. doi: 10.1109/ISCIT.2017.8261234.

[14] M. Hu and F. You, “Research on animal image classification based on transfer learning,” in *Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering*, in EITCE ’20. New York, NY, USA: Association for Computing Machinery, 2021, pp. 756–761. doi: 10.1145/3443467.3443849.

[15] T. Vidhyalatha, Y. Sreeram, and E. Purushotham, “Animal Intrusion Detection using Deep Learning and Transfer Learning Approaches,” *Int. J. Hum. Comput. \& Intell.*, vol. 1, no. 4, pp. 51–60, 2022.

[16] J. J. Kandathil, A. Ashraf, J. Babu, J. Thomas, and J. Francis, “Real Time Recording and Monitoring of Wild Animal Movements,” in *2022 IEEE 19th India Council International Conference (INDICON)*, 2022, pp. 1–6. doi: 10.1109/INDICON56171.2022.10039699.

[17] V. Bharati, “A Deep Neural Network Machine Vision Application for Preventing Wildlife-Human Conflicts,” in *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, 2021, pp. 1–6. doi: 10.1109/AIMV53313.2021.9671013.

[18] S. M. S, T. Sheela, and T. Muthumanickam, “Development of Animal-Detection System using Modified CNN Algorithm,” in *2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, 2022, pp. 105–109. doi: 10.1109/ICAISS55157.2022.10011014.

[19] D. Meena, H. K. P, C. N. V. Jahnavi, P. L. Manasa, and J. Sheela, “Efficient Wildlife Intrusion Detection System using Hybrid Algorithm,” in *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2022, pp. 536–542. doi: 10.1109/ICIRCA54612.2022.9985684.

[20] S. Radhakrishnan and R. Ramanathan, “A Support Vector Machine with Gabor Features for Animal Intrusion Detection in Agriculture Fields,” *Procedia Comput. Sci.*, vol. 143, pp. 493–501, 2018, doi: https://doi.org/10.1016/j.procs.2018.10.422.

[21] https://almablog-media.s3.ap-south-1.amazonaws.com/9_d0e24ad09a.png

[22] https://www.researchgate.net/publication/353745513/figure/fig1/AS:1053974799081472@1628298582631/The-process-of-convolution-of-kernel-over-the-image.png

[23] https://media.licdn.com/dms/image/C5612AQH5AQoL7ImHFA/article-cover_image shrink_423_752/0/1616494208154?e=1719446400&v=beta&t=FdwG9PGSuk_hXWygNwRDXAXaGqbp95PySgwdfAFfcRg

[24] https://www.researchgate.net/profile/Dilbag-Singh 12/publication/340979571/figure/fig10/AS:960341680861217@1605974707803/Rectified-linear-

unit-ReLU-activation-function.png

[25] https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcT0LV7cdYGY58pKh2nKa8-QIOfRYRuDzZ8aFDLzH-eUFKptMWkV

[26] https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQPVs81j0I0FP3bqQ187syoaojzdntNcrcGRuqNdAEDD8OFoYRy

[27] https://sds-platform-private.s3-us-east-2.amazonaws.com/uploads/73_blog_image_1.png

[28] https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcSrhbWCYgUScdiuHdUUS50sA4tHI5E_e7zncGWdj9XgEP4bEfee

[29] https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcR2VKJbYfrNE0kFrGZ7a5B2wgbovrik-jP2FGHjKBeQPo3wISBK

[30] https://carpentries-incubator.github.io/deep-learning-intro/fig/neural_network_sketch_dropout.png

[31] https://cdn.analyticsvidhya.com/wp-content/uploads/2021/04/Screenshot-from-2021-04-01-17-26-04.png

APPENDIX:

```
# python librairies installation
# display, transform, read, split ...
import numpy as np
import cv2 as cv
import os
import splitfolders
import matplotlib.pyplot as plt
# tensorflow
import tensorflow.keras as keras
import tensorflow as tf
# image processing
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
# model / neural network
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
"""
Step 2 - Data preprocessing
To use your data (images), you have to pre-process them.
Create Keras data generators
"""
import splitfolders
# split data in a new folder named data-split
splitfolders.ratio("C:/Users/sreer/Downloads/Page1",
output="C:/Users/sreer/Downloads/Animal Split/", seed=1337, ratio=(0.7, 0.2, 0.1),
group_prefix=None, move=False)
datagen = ImageDataGenerator()
# define classes name
class_names = ['Calm','Bear','Fox','Hyena','Lion','Tiger','Wolf']
# training data
train_generator = datagen.flow_from_directory(
    directory="C:/Users/sreer/Downloads/Animal Split/train",
    classes = class_names,
    target_size=(224, 224),
    batch_size=32,
    class_mode="sparse",)
# validation data
valid_generator = datagen.flow_from_directory(
    directory="C:/Users/sreer/Downloads/Animal Split/validation",
    classes = class_names,
    target_size=(224, 224),
    batch_size=32,
    class_mode="sparse",)
# test data
test_generator = datagen.flow_from_directory(
    directory="C:/Users/sreer/Downloads/Animal Split/test",
    classes = class_names,
    target_size=(224, 224),
    batch_size=32,
    class_mode="sparse",)
"""
Step 3 - Build the model
The first step is to build the model, using **ResNet50**.
"""
# ResNet50 model
resnet_50 = ResNet50(include_top=False, weights='imagenet', input_shape=(224,224,3))
for layer in resnet_50.layers:
    layer.trainable = False
# build the entire model
x = resnet_50.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.5)(x)
```

```
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64, activation='relu')(x)
x = layers.Dropout(0.5)(x)
predictions = layers.Dense(7, activation='softmax')(x)
model = Model(inputs = resnet_50.input, outputs = predictions)
"""
Step 4 - Train the model
**Adam** optimizer is used to train the model over **10 epochs**. It is enough by using
Transfer Learning.
The loss is calculated with the **sparse_categorical_crossentropy** function.
"""
# define training function
def trainModel(model, epochs, optimizer):
    batch_size = 32
    model.compile(optimizer=optimizer, loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
    return model.fit(train_generator, validation_data=valid_generator, epochs=epochs,
batch_size=batch_size)
optimizer = keras.optimizers.Adam(learning_rate=0.001)
# launch the training
model_history = trainModel(model = model, epochs = 20, optimizer = optimizer)
# Display **loss** curves:
loss_train_curve = model_history.history["loss"]
loss_val_curve = model_history.history["val_loss"]
plt.plot(loss_train_curve, label = "Train")
plt.plot(loss_val_curve, label = "Validation")
plt.legend(loc = 'upper right')
plt.title("Loss")
plt.show()
# Display **accuracy** curves:
acc_train_curve = model_history.history["accuracy"]
acc_val_curve = model_history.history["val_accuracy"]
plt.plot(acc_train_curve, label = "Train")
plt.plot(acc_val_curve, label = "Validation")
plt.legend(loc = 'lower right')
plt.title("Accuracy")
plt.show()
# Save the model
model.save('D:/Projects/Major/models from spyder/forestmodel.h5')
"""
Step 5 - Evaluate the model
The model is evaluated on test data.
"""
from tensorflow.keras.models import load_model
# Load your saved model
modelnew = load_model('D:/Projects/Major/models from spyder/forestmodel.h5')
test_loss, test_acc = model.evaluate(test_generator)
print("The test loss is: ", test_loss)
print("The best accuracy is: ", test_acc*100)
"""
Step 6 - Deploy the model
The model is deployed on live camera feed.
"""
import cv2
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input
# Define the classes
classes = ['Calm', 'Bear', 'Fox', 'Hyena', 'Lion', 'Tiger', 'Wolf']
# Function to preprocess the image
def preprocess_img(img):
    img = cv2.resize(img, (224, 224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img
# Accessing live camera feed
```

```
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Preprocess the frame
    processed_frame = preprocess_img(frame)
    # Predict the class
    prediction = modelnew.predict(processed_frame)
    predicted_class = classes[np.argmax(prediction)]
    # Display the result
    cv2.putText(frame, predicted_class, (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 2,
(255, 0, 0), 4)
    cv2.imshow('Animal Classification', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```