IoT Based Wild Animal Identification System Using Deep Convolution neural Network For Farm Protection

A dissertation submitted to the University of Hyderabad
in partial fulfillment of the requirements

for the award of the degree of
Master of Technology
in
Artificial Intelligence
by

Amit Biswas
(19MCMI19)

Supervisor:
Prof. Abhishek Thakur

School of Computer and Information Science
University Of Hyderabad
Hyderabad 500046 (India)

1 August 2021

This Thesis is Dedicated to

My family and My Supervisor

Certificate

School of Computer and Information Science

University of Hyderabad

This is to certify that the thesis entitled “IoT Based Wild Animal Identification System Using Deep Convolution neural Network For Farm Protection” submitted by Amit Biswas (19MCMI19), in partial fulfillment of the requirements for the award of Master of Technology in Artificial Intelligence is a bonafide work carried out by him under my supervision and guidance.

This dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. Abhishek Kumar Thakur
Assistant Professor and Supervisor
IDRBT
Hyderabad

Dr. Chakravarthy Bhagvati
Professor and Dean
School of CIS
University of Hyderabad

Declaration

I, Amit Biswas hereby declare that this dissertation entitled "**IOT Based Wild Animal Identification System Using Deep Convolution neural Network For Farm Protection**" submitted by me under the guidance and supervision of **Prof. Abhishek Thakur** is a bonafide work carried out by myself. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma. I hereby agree that my dissertation can be deposited in Shodganga/INFLIBNET.

A report of plagiarism statistics from the University Librarian is enclosed.

Amit Biswas

Date: (19MCMI19)

// **Counter Signed** //

Date: Prof. Abhishek Thakur

Supervisor

Acknowledgement

First and foremost, my utmost gratitude to Dr. Abhishek Kumar Thakur, Assistant professor, my project supervisor, for his supervision, advice, and guidance from the very early stage of this research as well as giving me extraordinary experiences throughout the work. His vision, encouragement and support in various ways made this work possible. I feel blessed to have got an opportunity to work under his guidance.

I am also thankful to Prof. Chakravarthy Bhagwati,Professor and Dean, School of CIS, University Of Hyderabad. My thanks and appreciations also goes to my fellow colleagues for their direct and indirect support in completing my dissertation work.

I am also thankful to Prof. D. Janakiram, Director of IDRBT, Hyderabad. Last but not the least I would like to express my sincere regards to my family for their encouragement and support during my project.

Amit Biswas
University of Hyderabad
1 August 2021

Abstract

Internet Of Things started playing a significant role in our day-to-day life. In the last few decades, iot stand out to be one of the most emerging technology to get into every sector starting from Home, Office, Public Place, Farmland, etc. This rising interest has drawn the attention of more researchers in this field. Especially, in agriculture, implementing iot could be a game-changer for the growth of our economy as 17-18 percent of our GDP directly comes from Agriculture Sectors. But we often have seen that the production of our agriculture farm is not at par to compete at the global level. Crops often get damaged due to several reasons like bad weather conditions, crop disease, poor farming strategy, different wild animals attacks, etc. Farmer has to suffer huge financial losses due to those reasons. To cope up with those problems iot turned out to be very useful in various ways like smart farming, crop diseases detection, weather monitoring, etc. This paper will give an iot-based solution to monitor and protect Agricultural farms from wild animals and unauthorized human intruders. The modules we will be using PIR sensor, ESP32 cam module, RaspberryPi. Initially, we have experimented with a PIR sensor to detect motion and capturing the range. Next, we will start training our deep learning model with a dataset for detecting the type of animal species, for that matter a pre-trained convolution neural network(CNN) will be used in Rashberry pie. To make it energy efficient, surveillance camera only become activated when it detects some motion in its range.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the last few years world is going through a huge technological shift in terms of communication and Automation as we have started adopting cutting-edge technologies like the Internet Of Things and Artificial Intelligence. So far we have had internet only on servers, personal computers (PC), Portable Digital Device(PDA), etc. But now the internet has started extending its boundary towards every tiny object or thing that surrounds us. Every object or thing is getting more and more intelligent and started communicating with each other through the internet around the world autonomously without human intervention. IOT and AI started expanding their footprint in every sector like Business, Enterprise, Healthcare, Traffic monitoring, etc. The agricultural sector is no exception either.

Especially in Agricultural farms, these technologies started playing a vital role in the growth of those sectors, for example, weather monitoring, Crop management, Cattle management, Soil quality, etc. However, there is one more major issue in Agriculture that needs to be addressed is the wild animal attack which is quite common in India. Every year these sectors suffer a significant loss for this.

In our proposed work, we have introduced an IOT based security system to detect and recognize animals and inform the owner as fast as possible using the fog computing technique. Later, this system can be incorporated with an automated repelling system depending on the detected animal also. Here we have built a client-server model using two types of nodes. The first node is the edge node which is made by ESP32 cam module in combination with PIR sensor and this node is responsible for capturing image frame and sending it to another node which is central node which is built by Raspberry pi 4 on which we have installed Apache server where those image data will be processed and notification of detected animal will be sent to the owner through email immediately. And edge node will communicate with the central node through HTTP protocol as a client-server model.

1.1 Background And Motivation

Automation in Agricultural farms has a significant effect on the production of Agricultural products. IoT has already been used in many different ways in Agricultural to monitor real-time data like temperature, humidity, soil moisture, leaf moisture, light, rain, etc from the ground through a different sensor. Many states of the art Microcontroller chip like Arduino Uno, Arduino nano, ESP8266, ESP32, etc are used in combination with those sensors to sense data from the environment. Those Microcontrollers have been being equipped with new state-of-the-art communication technologies, for example, wireless radio supporting IEEE 802.11 (Wi-Fi), IEEE 802.15.4 (ZigBee, 6LoWPAN), Bluetooth Low Energy (BLE), etc. to connect themselves to the cloud for processing of data. Another important area where iot can play a significant role is for the protection of Agriculture farms from an especially animal attack which is quite prevalent in India as we see that a huge amount of our agricultural product gets destroyed from animal attack in every year. Here Internet Of Things has been used in combination with Artificial Intelligence(AI) specially Convolution Neural Network(CNN) to detect an animal intrusion or any unauthorized human intruder. In combination with a microcontroller, many types of camera modules have been used to capture image frames and sending them to the cloud using various protocols like MQTT or HTTP. And in the cloud itself, the image data get processed and the type of that image detected. Depending on that detected animal different precaution has been taken like ultrasound frequency, predator sound[1], etc to keep the animal away from the farm. however, there are many challenges in this solution that can be improved like coverage area, the latency of the entire system, scalability, etc.

1.2 Objective

This thesis gives a solution to the challenges of some preexisting Agricultural farm protection systems in terms of cost-efficiency, energy efficiency, latency, etc. Here we capture image data from the environment and send it to a local server for animal recognition. Then information regarding the detected animal will be sent to the owner and precautions can be taken accordingly. Here we are using the fog computing paradigm instead of cloud computing, which means data won't have to transmit outside of the network. Fog computing technique significantly improves response time as well as bandwidth conservation as data isn't going outside of the network.

Chapter 2

Literature Survey

2.1 Literature Review

Several methods have already been proposed to protect the farm from an animal attack. Some systems used PIR sensors for detection of animals, ultrasound sound frequency which helps to ward off an animal, loud noise has also been used. Some have deployed the CNN model in the cloud for animal recognition. Some used IP cameras for the verification of intrusion after detection through the PIR sensor.

N S Gogul Dev, KS Sreenesh, PK Binu, "IoT Based Automated Crop Protection System"[2], introduced a method by which animal intrusion can be detected through PIR sensor which is connected to PI, and raspberry pi immediately turn on the IP camera to capture an image and send it to the server for image recognition and detected type of animal is then sent back to the raspberry pi. Pi transmit the ultrasound frequency depending on the detected animal.

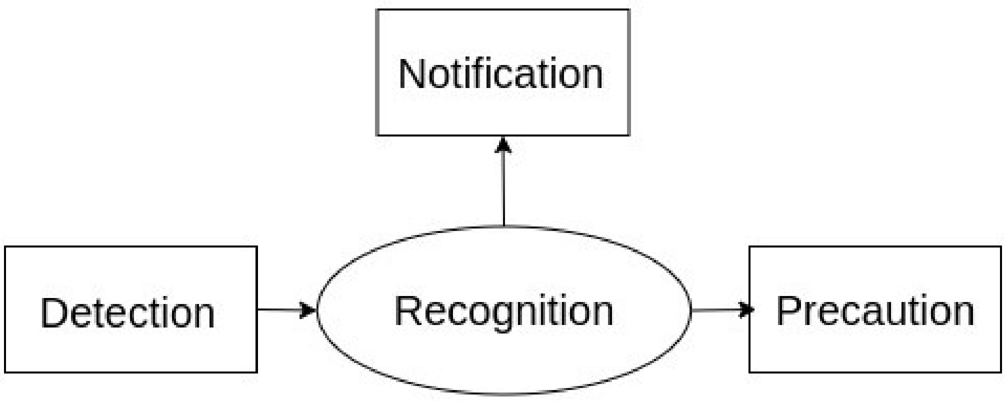
Khirod Chandra Sahoo, Umesh Chandra Pati, "IoT based intrusion detection system using PIR sensor"[3], implemented an intrusion detection system by PIR sensor. They have used ZigBee here as a wireless sensor network (WSN). PIR sensor is used here for detecting any type of movement of animals, people, or any object. If the sensor detects any motion, an alert message will be sent to the user using the GSM module. An ESP8266 is used to sends sensor data to ThingSpeak(public server). And when the user gets the text alert, he can use an IP camera to check the farm live remotely.

Hung Nguyen, Sarah J. Maclagan, Tu Dinh Nguyen, Thin Nguyen, Paul Flemons, Kylie Andrews, Euan G. Ritchie, Dinh Phung, "Animal Recognition and Identification with Deep Convolutional Neural Networks for Automated Wildlife Monitoring"[4], introduced a wildlife monitoring system using state of the art CNN Architecture for animal recognition. Here three different model has been explored, VGG16, ResNet50, Light AlexNet to check their accuracy.

Stefano Giordano, Ilias Seitanidis, Mike Ojo, Davide Adami, Fabio Vignoli, "IoT solutions for crop protection against wild animal attacks"[5], introduced a weather monitoring system that provides real-time and historic data from the farm and a repeller system with the help of state-of-the-art Cortex ARM M0+ microprocessor in combination with PIR sensor which is responsible for ultrasound frequency production to scare the animal away. They have adopted different wireless technology like 6LoWPAN, WiFi, Zigbee, etc.

2.2 Inferences Drawn from Literature Review

Many have proposed solutions regarding farm protection from wild animals and unauthorized human intrusion. Among them, the most effective are those which are capable of creating automated responses because this is one of those systems that need an immediate response with different precaution systems like loud noise, predator sound, ultrasound frequency, etc. This type of system can be divided into four major parts. The first part is detection using edge sensors, the second part is recognition for identifying the animal, the third part is called protection, the fourth one is notification for alerting the owner.



The effectiveness of such a system is largely depending on how effectively we detect any animal activity around the farm and how fast we recognize the animal, cause depending on that automated precaution can be taken.

Chapter 3

Problem Formulation and Proposed Work

3.1 Introduction

We have seen many proposed solutions regarding farm protection systems. In recent years those systems improved significantly as we have adopted automation techniques using IoT and computer vision. But when it comes to installation of such system in a remote area such as Agricultural farm we have to consider the different unfavourable situation. And considering this, we have to optimize the system.

3.2 Problem Statement

When we attempt to build such an automated farm protection system, it would be ideal to take into consideration a few major concerns. First and foremost it should be able to take action immediately without human intervention and in that regard response time is very crucial.

Another issue is area coverage. Some farms are quite large that it becomes a really hectic task to install such a device that will give full coverage to the entire firm. In that case, one solution could be to use an IoT gateway where all the sensor nodes will send image data using MQTT or HTTP and the gateway will send those image data to the cloud through the GSM module for processing of those images for animal detection. And depending on that detected animal different types of automated precaution systems can be implemented like ultrasound frequency, predator sound, loud noise, etc. But here also image data have to go through several network hops to reach the cloud for processing and again result has to come to the gateway for automated action. This could create a high latency.

Energy consumption is another major issue as all the edge node which is mostly combined of motion sensor, microcontroller, camera module, etc. will spread throughout the farm would be battery dependent only. So, we should use a different method to reduce battery consumption also. We should maintain the cost of the system as low as possible so that it becomes affordable for the farmer.

3.3 Proposed Work

The functions of our system can be described by the following steps.

3.3.1 Motion Detection From Edge Node

We have built an edge node by using the esp32 cam module and PIR sensor. PIR sensor is connected to esp32 through a GPIO(13) pin of esp32. Normally esp32 stays in deep sleep mode. Once PIR detects the motion, it triggers the esp32 GPIO pin. As soon as its trigger, esp32 comes out from deep sleep mode and captures an image(width=800, height=600) of the area, and sends it to the central node through its WiFi(802.11b/g/n/e/i) module. Here we are using an HW 6F22 battery to supply power to the entire node.

3.3.2 WiFi-based Wireless Sensor Network

Our edge node is capable of communicating with the central node using a WiFi module embedded in esp32. Here we are using a client-server architecture. esp32 is getting connected to the central node as a client using the HTTP protocol for sending an image. As a server, we are using raspberry pi, on which we have installed an apache server through which we are receiving images. Here we are using WiFi instead of other communication technology because we need a considerable high bandwidth to send an image.

3.3.3 Animal Identification At Central Node

Here we have used raspberry pi as the central node on which we have installed the apache server. We have written PHP code to receive images in a temporary folder from which those images are used for recognition. Tensorflow Lite version has been installed on this apache server on which we have deployed our CNN model. The model had been converted to a lite version by TFLiteConverter for use in raspberry pi. The received image in the server will first be pre-processed because images captured from esp32 cam are generally of size 800*600, we converted it into 224*224. After that, it is processed through a binary classification model to check if there exist any animals in the frame or not. If it detects any animal the image is then given to a multi-class classifier to classify the animal.

3.3.4 Alert Generation

Depending on the type of animal detected, many different precautions can be taken. We are using the smtplib python library to give notifications to the owner through the mail. It is normally a mail transfer protocol that can send emails to any valid email id on the internet. Here we have used SMTP connection in TLS mode for security reasons. TLS is basically meant transfer layer security which can encrypt all the SMTP commands.

Chapter 4

Methodology and Implementation

4.1 Basic Idea

Basic idea is to create an animal recognition system that recognizes the animal as close as where it gets detected which means we bringing data processing from the cloud to the local server or IoT gateway to get the fast result so that automated precaution can be taken faster. An email notification will also be sent to the owner if any animal intrusion is detected.

4.2 System Algorithm

This system is comprised of two types of nodes. The first type is edge node where any type of motion gets detected and photo captured accordingly and the image is then sent to another type of node, which is called central node made by Raspberry pi 4 for the processing of those image data for detecting animals in those images.

1. In the edge node, the PIR sensor detects any type of animal movement around the farm. If it detects, it turns on the camera in active mode.

2. Real-time photo is captured and sent to the central node for image recognition.

3. RPI of central node use Tensorflow lite to detect and classify the animal using CNN model. And the owner is then notified by email.

4. Later on depending on classified animal automated precautions can be taken from central node like ultrasound frequency, loud noise, etc.

4.3 Architecture

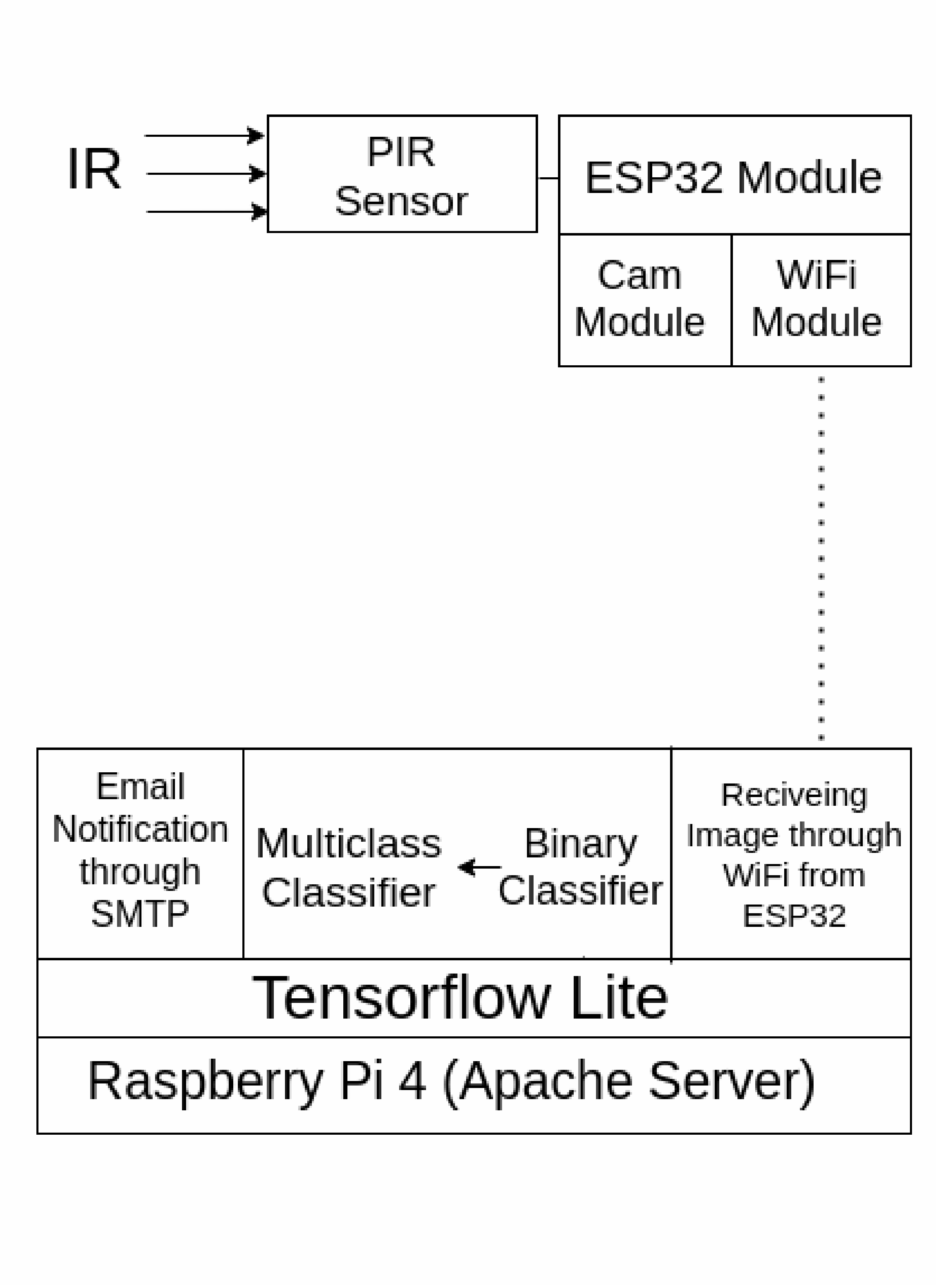


Figure 4.1: Model Architecture

4.4 Tools/Hardware/Software to be used

4.4.1 PIR Sensor

PIR sensor is one of the most commonly used motion sensors as it gives a wide range of coverage. It can give a range of around 10-meter distance. Horizontally and vertically it can give around 110 degree and 90-degree coverage respectively. The way it functions is quite complicated than others also.

Most important feature by which it detects someone's presence is called infrared(IR). But before that, we need to understand what infrared is. Every living and non-living object emits a certain amount of IR radiation. That radiation can't be seen in open eyes. But this electronic device can detect.

It generally has two slots for the detection of IR energy. Both slots are kept side by side. When there is no change in environment, both slots will detect the same amount of IR radiation but When something crosses from this range, it will first intercept the first slot causing positive differential change and if it leaves the area it will detect negative differential change. In both cases, a motion will be detected. It basically detects the change of IR radiation in their range. When this detection happens our camera gets activated.

4.4.2 ESP32 CAM

ESP32-cam is a low-cost microcontroller board for development purposes with WiFi, an SD card slot, and an OV2640 camera module attached with it[6]. It has several GPIO pins, 3 ground pins, and 2 power pins(3.3v and 5v). We have used GPIO 13 to connect the PIR sensor with it. And when GPIO 13 pin trigger, we activate the camera module attached with esp32 to capture one frame and send it to Raspberry pi for processing through WiFi module. Here we have used Arduino board instead of FTDI programmer to upload the code from Arduino IDE to esp32 module as the esp32 module doesn't have the option to directly upload the code. Here we have used GPIO 1 and GPIO 3 pins of esp32 and connected with TX and RX pins of Arduino for uploading code.

4.4.3 Raspberry Pi

Raspberry pi is an affordable mini-computer that is capable of doing an interesting project. In our project, we have used Raspberry Pi 4 with 1.5GHz clock speed, 4GB RAM, one micoSD card slot for booting OS. We have installed a lite OS for our RPI, called 'Raspbian'. Here, we are using RPI as a central node where all computation regarding animal classification will be done.

4.4.4 Battery

Here we have used an HW 6F22 battery of 9V for our testing. As its voltage is higher than required, we have connected resistors to reduce it to 5V.

4.4.5 TensorFlow Lite

TensorFlow Lite is basically a light version of a deep learning framework that is designed specifically for IoT, mobile, embedded devices, etc. We first train the TensorFlow model. Then converted it using TensorFlow lite converter to TensorFlow lite file format (.tflite). And it can now be deployed to the different embedded devices. The main difference between TensorFlow and TensorFlow Lite is that TensorFlow can be used for both operation training and inference but TensorFlow lite can be used only for inference on low computing devices like mobile, IoT, etc.

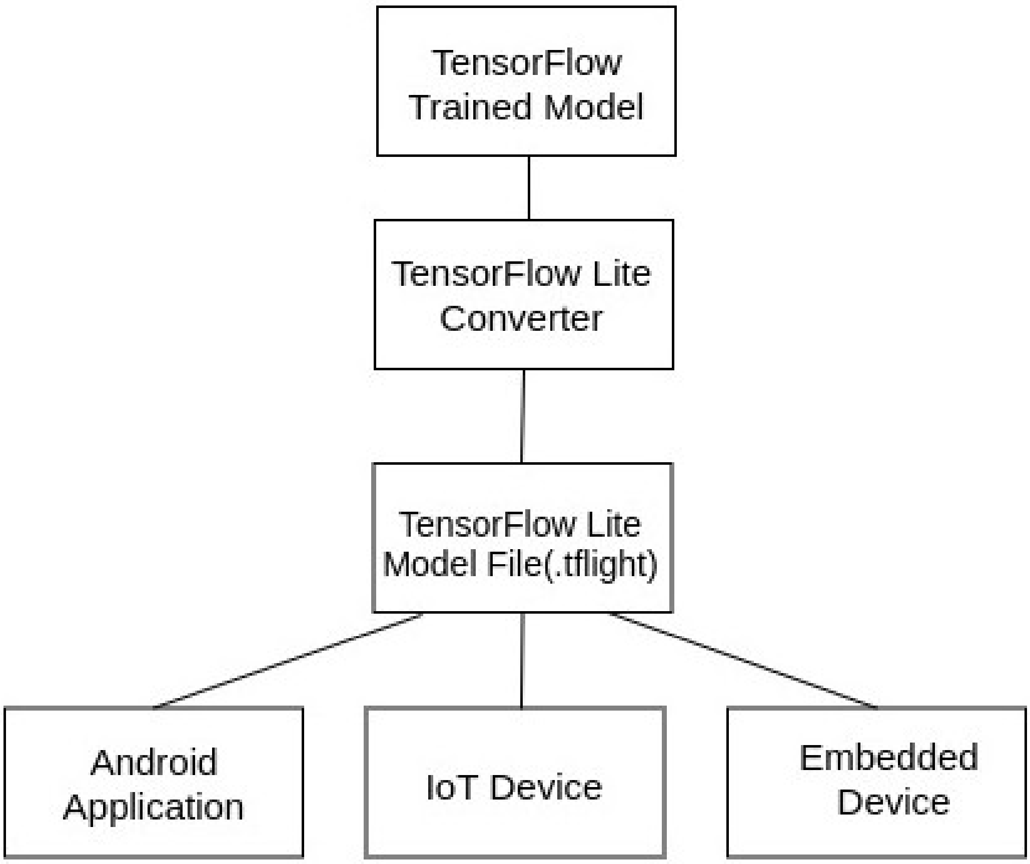


Figure 4.2: Architecture Of TensorFlow Lite

4.5 Working Circuit Of Sensor Node

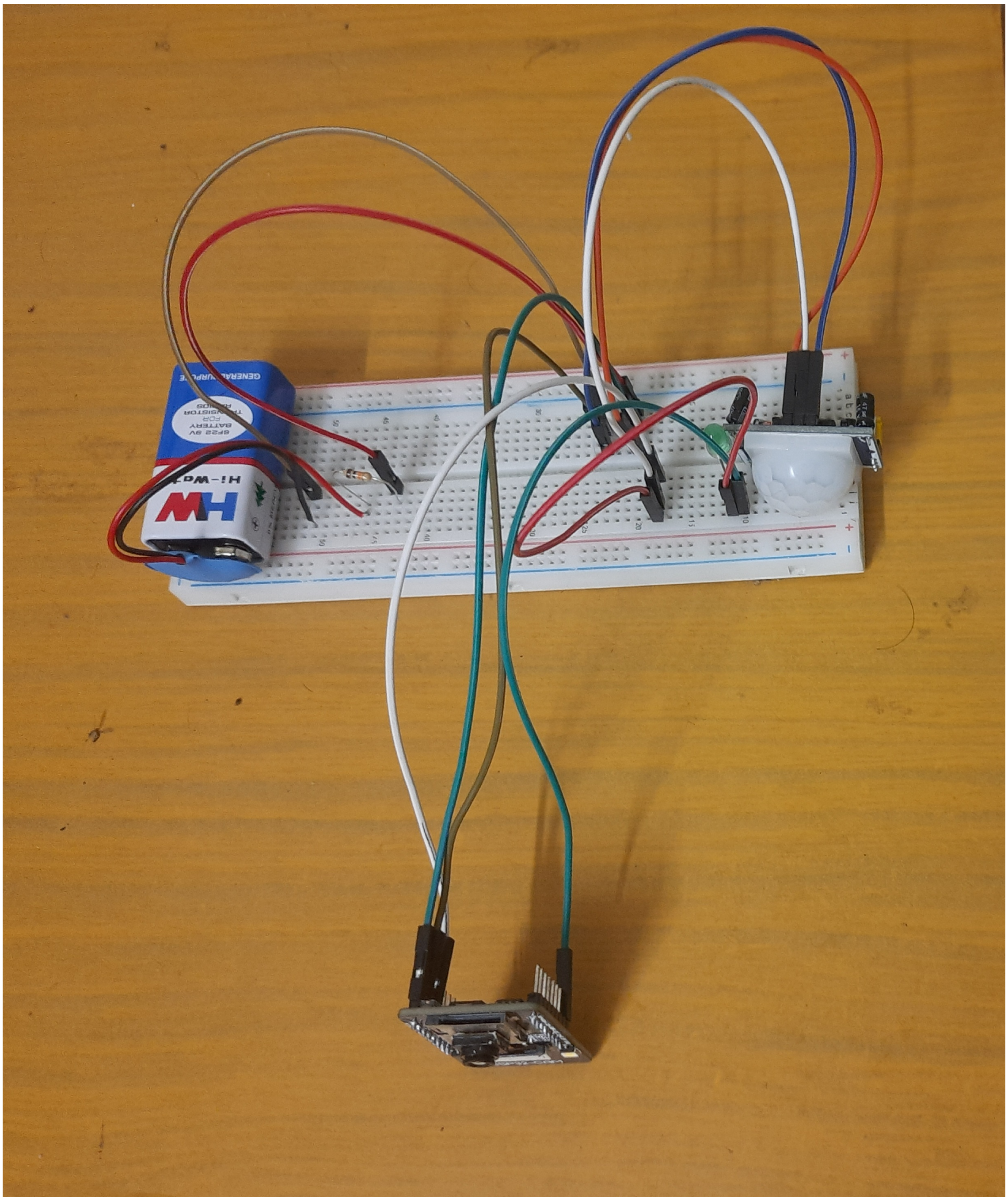


Figure 4.3: Circuit Design Of Edge Node

4.6 Evaluation

Here we have adopted a fog computing paradigm in our system that means our data need not go to the cloud. Data can be processed as close as to the data source. We are using raspberry pi as a fog node where we are processing the image data. As it improves response time significantly, it becomes very useful for real-time applications. As our data staying inside our local network bandwidth requirement becomes very less. This system will also be very useful in a remote area where internet access is very slow.

Cost is one of the most crucial parameters for our system, the cause is intended to deploy on an Agricultural farm. It should be affordable to the farmer. We have used some cheapest devices for our project. The edge node is built by one esp32 module, one PIR sensor, one battery for the power supply. We can use any 5V battery. The total cost to build an edge node could be around one thousand. We can use one raspberry pi as a central node for all data processing and several edge node can deployed depending on the area of the farm. The total cost could be close to six thousand.

As we are not letting the data come out from the network, data is getting processed locally not in the cloud, data privacy will be reserved and data would be much more protected. The system would be scalable as many edge nodes can be built. All nodes can communicate with the central node using a client-server architecture.

As we are using WiFi[7] to send image data from the edge to the central node, it consumes a lot of energy from the battery. To mitigate this issue, we have used a different power mode of esp32. The Esp32 cam module has different power modes, for example, active mode, modem sleep mode, light sleep mode, deep sleep mode, and hibernation mode. Here we made use of the deep sleep mode of esp32. In this mode, CPU, WiFi/BT baseband, and radio are powered off. Only ULP co-processor, RTC-memory, and RTC-peripheral are powered on. So, we can write a program for our need for a ULP co-processor and can store it into RTC memory for accessing the peripheral device. During this mode, the ULP co-processor can access some of the esp32 pins, which are called RTC-GPIO pins. Our microcontroller always stays in a deep sleep mood until it is triggered by the PIR sensor through GPIO pin 13.

Chapter 5

Animal Detection Using Convolution Neural Network(CNN)

5.1 Introduction

Automatic animal classification is one of the crucial parts of our project. How much accurate our system would become is solely dependent on how accurately we would be able to classify animals. Several machine learning algorithms can be used in that regard, but when it's about image classification, deep learning has algorithm has been proven to be best. There are several deep learning algorithms available, for example, ANNs, RNNs, CNN, LSTMs. But among them, CNN has given the most promising result recently. Here in our system, we will use CNN for our classification model. We are using a transfer learning technique that means we are going to use a state-of-the-art CNN model, VGG16, and reuse it for our purpose.

For the purpose of our system, we are creating two CNN models. The first one is for binary classification to check if there exists any animal or human being in the image or not because sometimes we might capture some image by mistake where there is no living object inside the frame. If it detects any living being on the frame the image will be sent to the second model to identify the type of animal. The second model is for multi-class classification where it will be able to classify 7 different types of wild animals including human beings.

5.2 Working Procedure Of CNN Model

Convolution Neural Network basically has four types of layers in it, convolution, relu layer, pooling, fully connected layer.

In convolution operation, we choose different features or filters and for each feature, we do a convolution operation on the image. here we consider an image to be a matrix of RGB value(0 to 255). we divide the value of each element of the matrix by 255 to make it in the range of 0 to 1. For each filter, we get a different feature map.

Next, this feature map is given to the relu layer. Here all the negative values of the feature map are converted to zero. The next pooling layer is used to shrink the image map into small size. We can do these three operation several times to reduce the image more. After that, we flatten the matrix and feed it to a fully connected network.

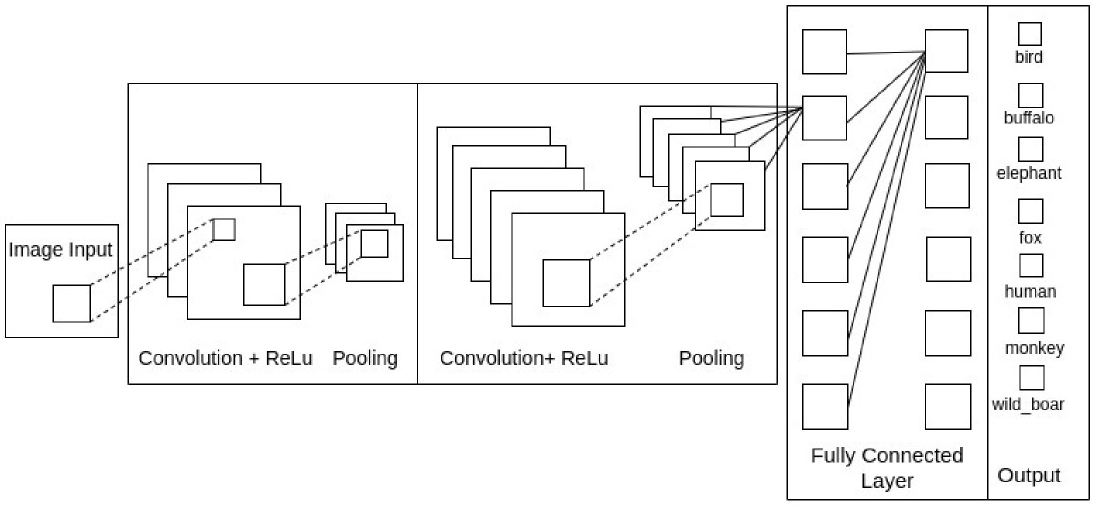


Figure 5.1: Architecture Of Convolution Neural Network

5.3 Transfer Learning Using VGG16

Transfer Learning is basically a way of gaining knowledge from one task and use that knowledge to solve another task. For example, if you learn to ride a bicycle, you can use that knowledge to ride a bike also. The main motivation for transfer learning is the complexity of a deep learning model. Generally, a standard deep learning model which is capable of solving a complex problem is basically trained with a huge dataset and it needs huge computation power to train it. So, transfer learning actually helps us utilizing knowledge gained from the previous task and applying it to new ones. If you have a huge data for a task m1, you may use its knowledge like features, weights for task m2, which might have very little data. In the case of computer vision problems, knowledge of certain features, such as corners, edges, shapes, etc can be transferred among different tasks.

In our project, we are using a pretrained CNN model, VGG16[8]. It is one of the best CNN models now for image recognition and it also won ILSVR(Imagenet) competition in 2014. They trained this model with Imagenet dataset with around 14 million image data which has 1000 classes. We have made two models using transfer learning from VGG16[9]. The first binary classification model was for filtering out all the images where no living being present. The second multi-class classification model is for recognizing animal after it gets filtered out from the first binary classification model.

5.4 Dataset Description

We have used African Wildlife, Oregon Wildlife dataset from Kaggle for our project. For the human dataset, we have used the Standford dataset for recognizing 40 human actions[10]. Dataset is divided into two parts. The first one is training data and the second one is validation data. 80 percent of the total dataset has been used for training and the remaining 20 percent has been used for validation purposes for both models. For the binary classifier, we have used a total of 5209(balanced) images as training data and a total of 1059(balanced) as testing data.

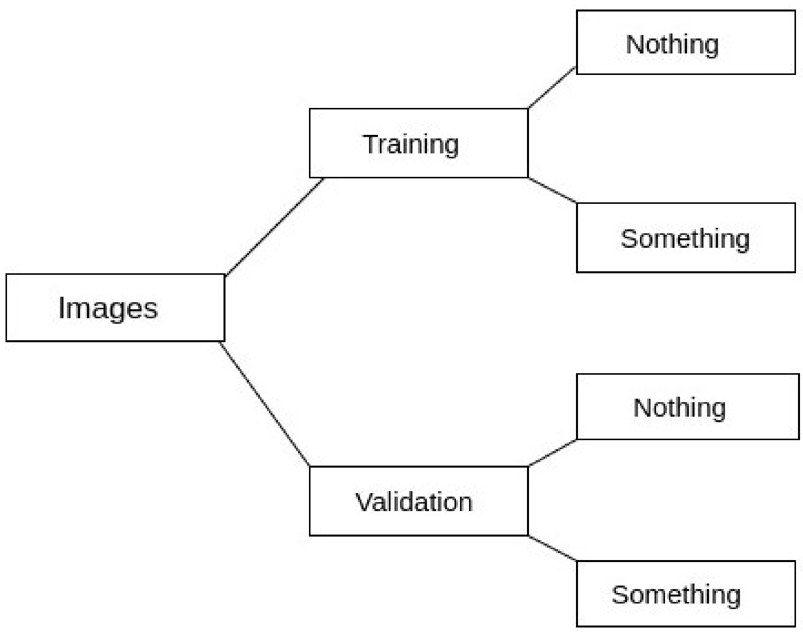


Figure 5.2: Dataset Of Binary Classifier

In the case of a multiclass classifier, we have given a total of 7000(balanced) images as training data for 7 classes and a total of 1400(balanced) as testing data. As our dataset isn't very huge, we aren't using the fine-tuning approach for our project. We are transferring the weights of features from the pre-trained model in our project by using transfer learning.

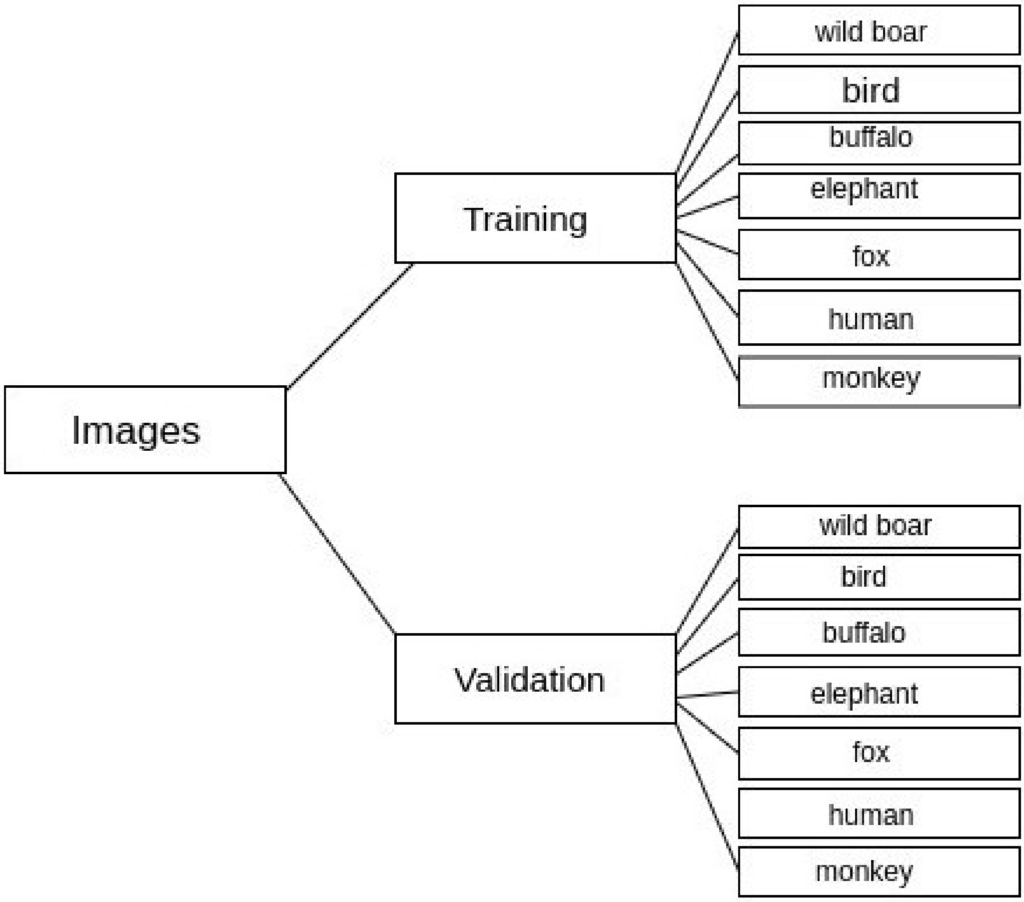


Figure 5.3: Dataset Of Multiclass Classifier

5.5 Model Description

```
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0

block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792

block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928

block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0

block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856

block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584

block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0

block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168

block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080

block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080

block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0

block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0

block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808

block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0

flatten_2 (Flatten)          (None, 25088)             0

dense_2 (Dense)              (None, 7)                 175623
=================================================================
Total params: 14,890,311
Trainable params: 14,890,311
Non-trainable params: 0
_________________________________________________________________
Found 6034 images belonging to 7 classes.
Found 1209 images belonging to 7 classes.
95
19
```

Figure 5.4: Model Summary Of Multiclass Classifier

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 50178
=================================================================
Total params: 14,764,866
Trainable params: 14,764,866
Non-trainable params: 0
_________________________________________________________________
Found 5209 images belonging to 2 classes.
Found 1059 images belonging to 2 classes.
82
17
```

Figure 5.5: Model Summery Of Multiclass Classifier

Chapter 6

Experiment and Result

We have trained our model in google colab which provided Tesla K80 , compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM, 12.69GB ram, and 68.35GB disk. Here we have taken both the fine-tuning and transfer learning approaches and compared both results.

6.1 Wild animal detector

In our binary classification model which is used here as a wild animal detector, we got 97.36 percent accuracy on batch size 64 by using the transfer learning technique. It was taking around 126 seconds for each epoch and 2 seconds for each step. But when we trained from the scratch using fine-tuning we got a subtle drop in accuracy. We got an accuracy of 93.86 percent using the fine-tuning approach. It was taking around 144 seconds for each epoch and 2 seconds for each step. We have tested our model by tuning different hyperparameters, for example, batch size, epoch, optimizer, activation function, etc. We have experimented with many different batch sizes like 16, 32, 64, 128, etc. The model was size around 60MB. Then we have used TFLiteConverter to reduce its size and deployed it on raspberry pi.

6.2 Wild animal Identifier

Here we have built a multi-class classifier for wild animal identification. It has seven classes like wild boar, monkey, human, bird, elephant, buffalo, and fox. Both the fine-tuning and transfer learning approach has been taken. In the case of the transfer learning technique, we got 90.16 percent accuracy. But for fine-tuning, we got 79.82 percent accuracy. Both have given the best result with batch size 64.

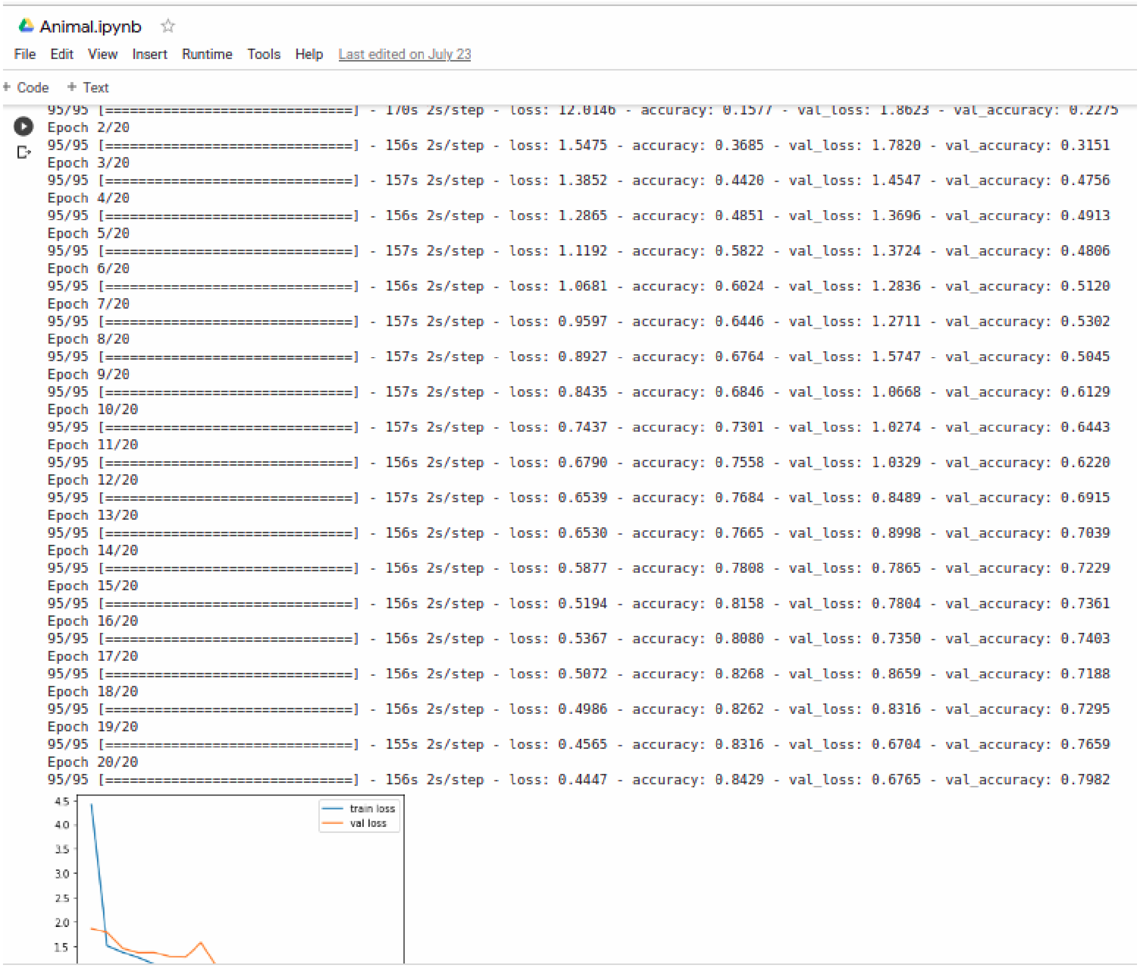


Figure 6.1: Training Process at google colab

Accuracy Table			
Wild Animal Detector		Wild Animal Identifier	
Fine Tuning	Transfer Learning	Fine Tuning	Transfer Learning
93.86	97.36	79.82	90.16

Table 6.1: Accuracy Of CNN Model

```
[2] import tensorflow as tf
    from keras.layers import Input, Lambda, Dense, Flatten
    from keras.models import Model
    from keras.applications.vgg16 import VGG16
    from keras.applications.vgg16 import preprocess_input
    from keras.preprocessing import image
    from keras.preprocessing.image import ImageDataGenerator
    from keras.models import Sequential
    import numpy as np
    from glob import glob
    import matplotlib.pyplot as plt
    from keras.preprocessing.image import ImageDataGenerator
    from keras.models import load_model
```

```
# re-size all the images to this
IMAGE_SIZE = [224, 224]
train_path = '/content/Datasets_B/Train' #All folders will be there according to number of classes.
valid_path = '/content/Datasets_B/Test' #All folders will be there according to number of classes.

# add preprocessing layer to the front of VGG
vgg = VGG16(input_shape = IMAGE_SIZE + [3], weights='imagenet', include_top=False)

#don't train existing weights
#for layer in vgg.layers:
  #layer.trainable = False

#useful for getting number of classes
folders_Train = glob('/content/Datasets_B/Train/*')
folders_Test = glob('/content/Datasets_B/Test/*')
print("Total number of folder in Test:",len(folders_Train))
print("Total number of folder in Test:",len(folders_Test))
```

```
[4] # our layers - you can add more if you want
    x = Flatten()(vgg.output)

    # x = Dense(1000, activation='relu')(x)
    prediction = Dense(len(folders_Train), activation='softmax')(x)

    # create a model object
    model = Model(inputs=vgg.input, outputs=prediction)

    # view the structure of the model
    model.summary()
```

Figure 6.2: Code Snippet(1)

```
# create a model object
[4] model = Model(inputs=vgg.input, outputs=prediction)

    # view the structure of the model
    model.summary()

    # tell the model what cost and optimization method to use
    model.compile(
      loss='categorical_crossentropy',
      optimizer='adam',
      metrics=['accuracy']
    )

    train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
    test_datagen = ImageDataGenerator(rescale = 1./255)

    training_set = train_datagen.flow_from_directory('/content/Datasets_B/Train', target_size = (224, 224), batch_size = 64,
                                                     class_mode = 'categorical')
    test_set = test_datagen.flow_from_directory('/content/Datasets_B/Test', target_size = (224, 224), batch_size = 64,
                                                class_mode = 'categorical')

    print(len(training_set))
    print(len(test_set))

[ ] # fit the model
    r = model.fit_generator(
      training_set,
      validation_data=test_set,
      epochs=20,
      steps_per_epoch=len(training_set),
      validation_steps=len(test_set)
    )
    # loss
    plt.plot(r.history['loss'], label='train loss')
    plt.plot(r.history['val_loss'], label='val loss')
    plt.legend()
    plt.show()
    plt.savefig('LossVal_loss')

    # accuracies
    plt.plot(r.history['accuracy'], label='train accuracy')
    plt.plot(r.history['val_accuracy'], label='val acc')
    plt.legend()
    plt.show()
    plt.savefig('AccVal_acc')
```

Figure 6.3: Code Snippet(2)

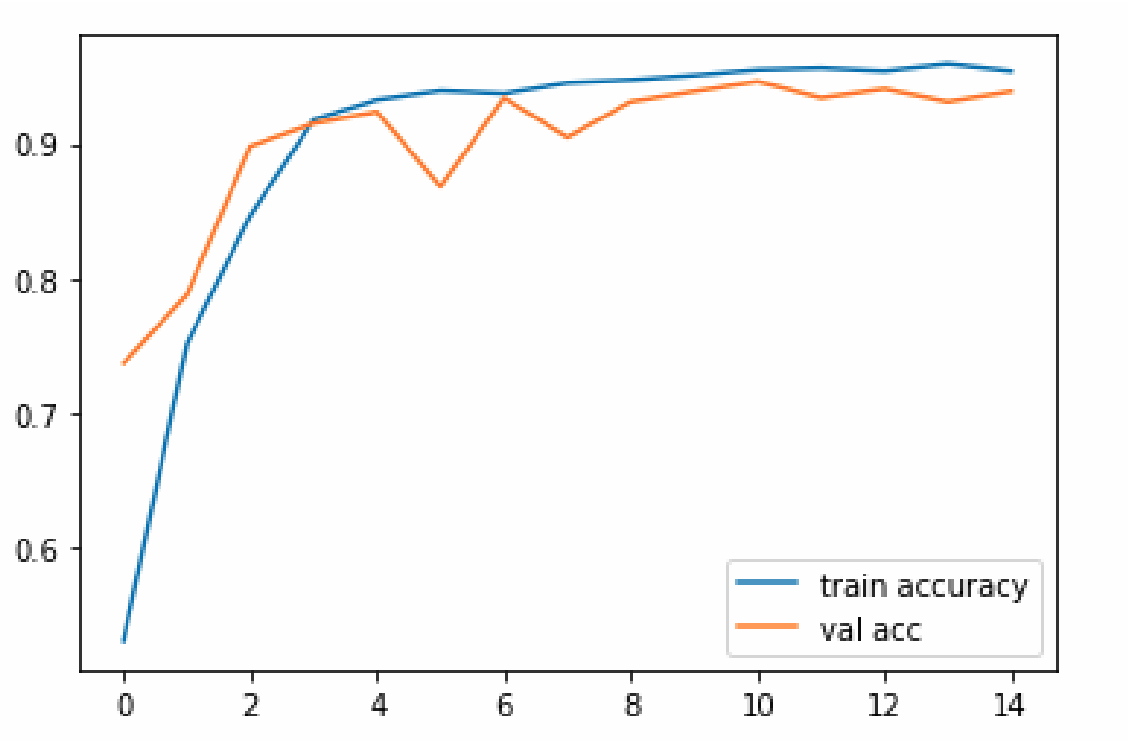


Figure 6.4: Training/Validation Accuracy Using Fine Tuning Approach(Binary Classifier)

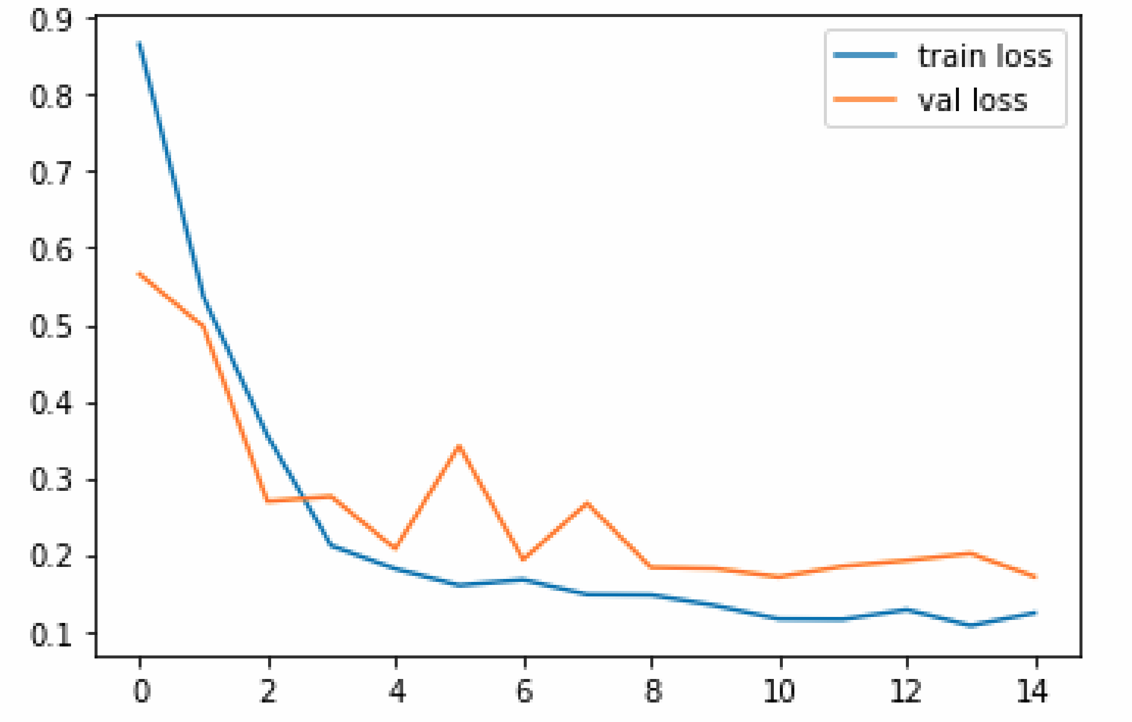


Figure 6.5: Training/Validation loss Using Fine Tuning Approach(Binary Classifier)

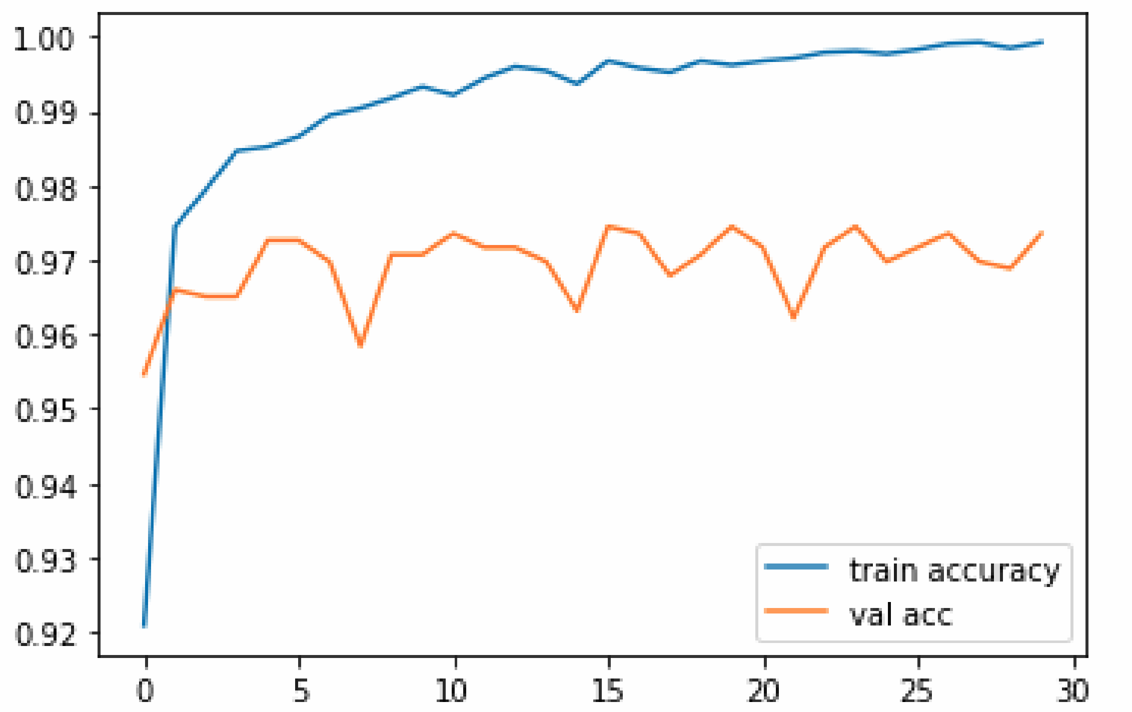


Figure 6.6: Training/Validation Accuracy Using Transfer Learning Approach(Binary Classifier)

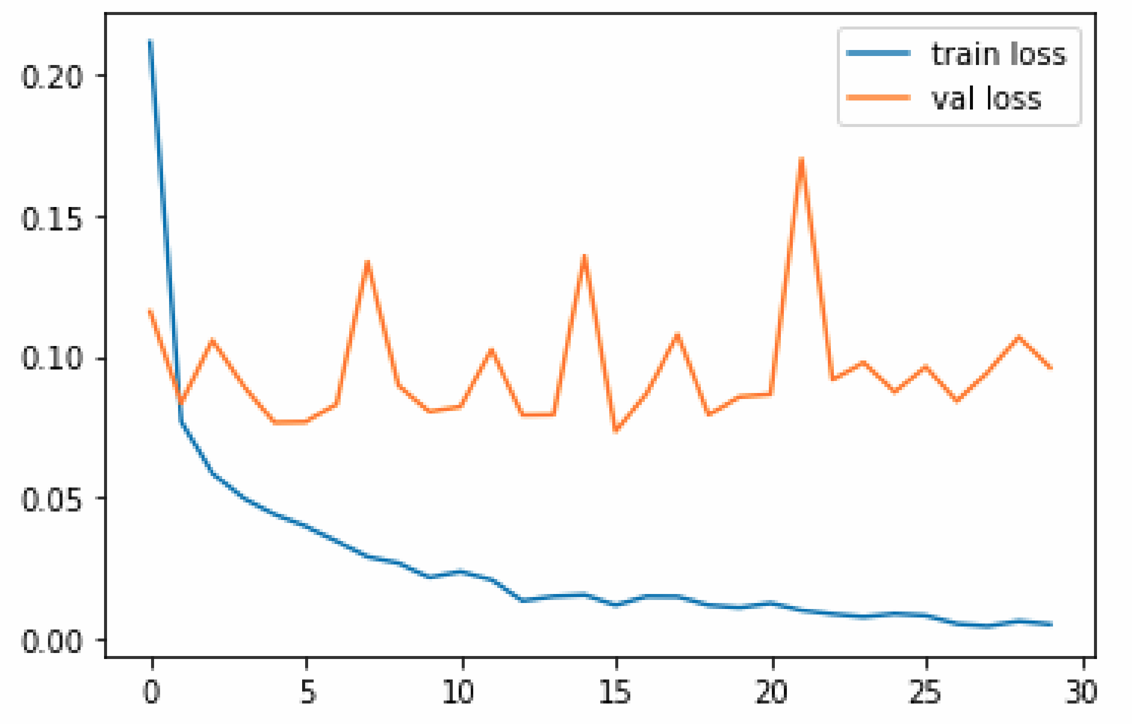


Figure 6.7: Training/Validation loss Using Transfer Learning Approach(Binary Classifier)

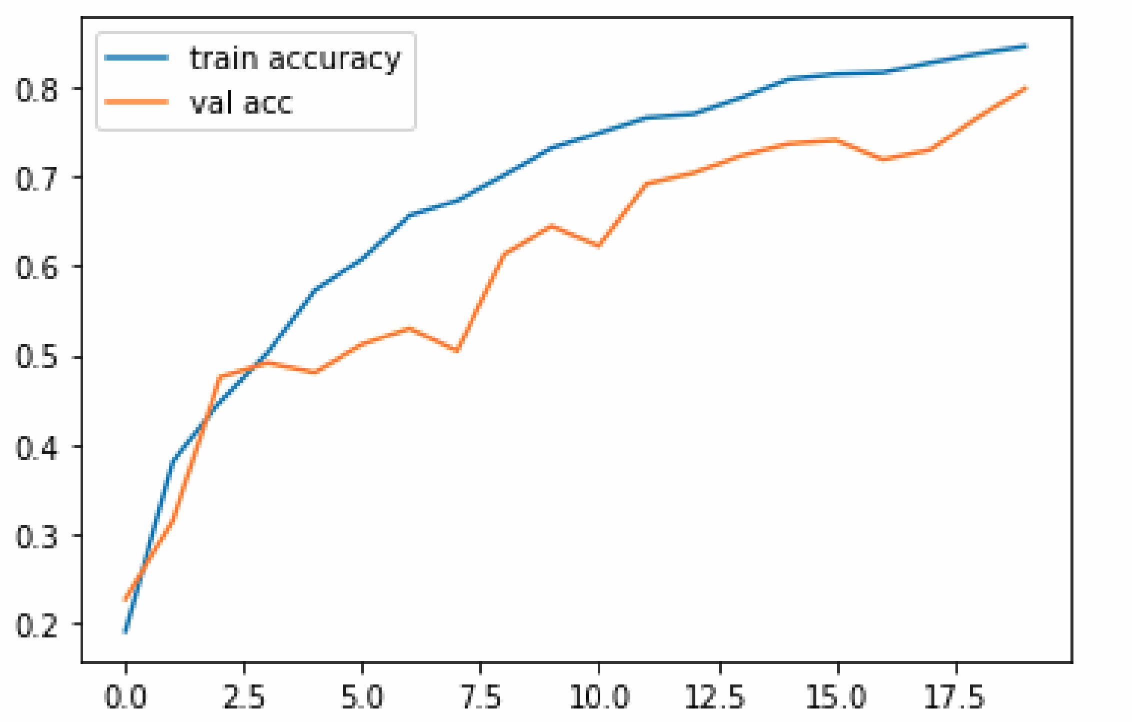


Figure 6.8: Training/Validation Accuracy Using Fine Tuning Approach(Multi-class classifier)

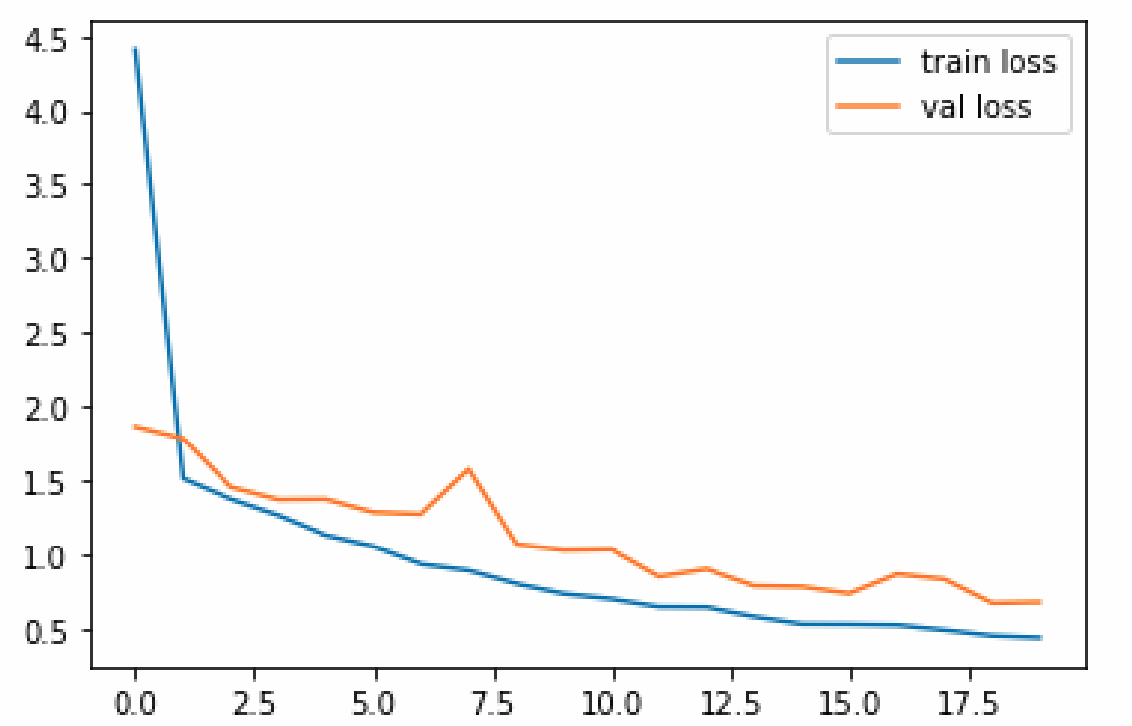


Figure 6.9: Training/Validation loss Using Fine Tuning Approach(Multi-class classifier)

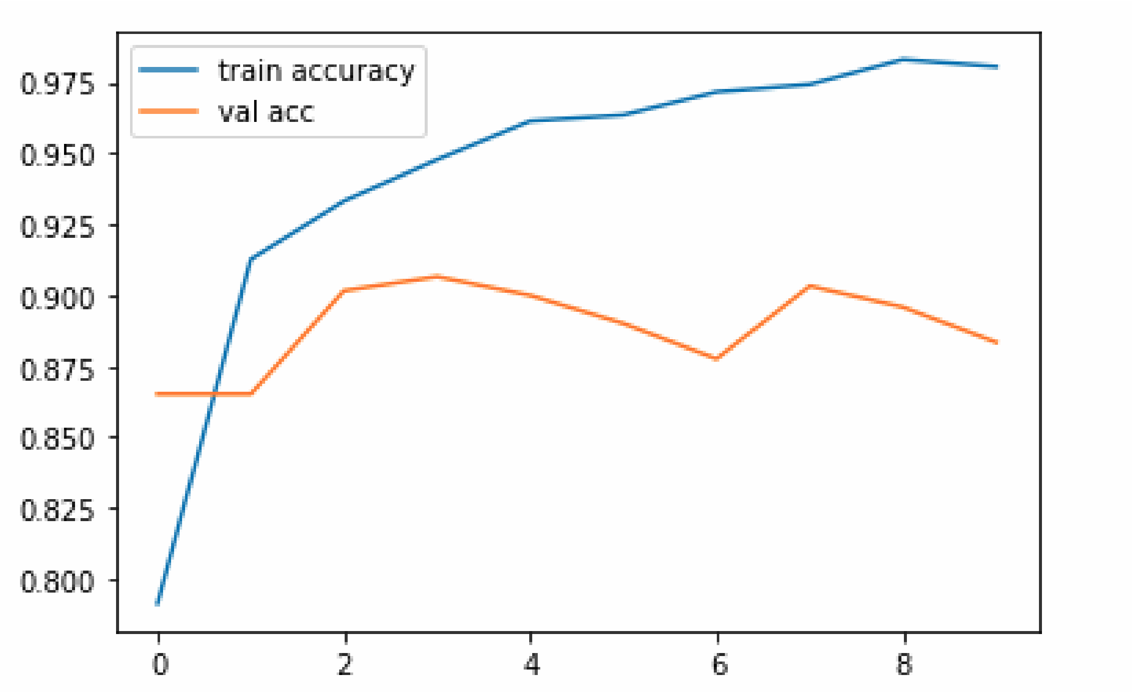


Figure 6.10: Training/Validation Accuracy Using Transfer Learning Approach(Multi-class classifier)

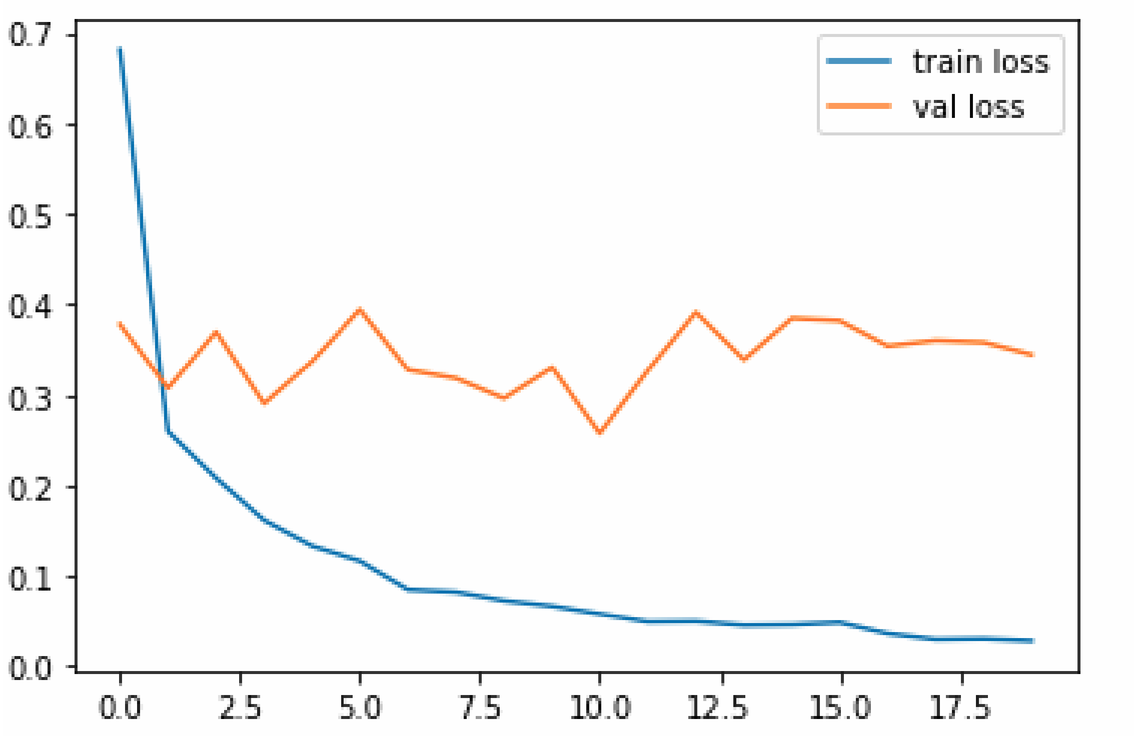


Figure 6.11: Training/Validation loss Using Transfer Learning Approach(Multi-class classifier)

Chapter 7

Conclusion And Future Scope

This paper proposed a design and implementation of a system that can protect the agricultural farm from an animal attack. This system can recognize animals around the farm by capturing the image and processing it locally in the raspberry pi server. We have used esp32 cam and PIR sensor to detect any intrusion. And for recognition purposes, we have trained a convolution neural network model and deployed it on a raspberry pi server. Our model was able to recognize an image where animals exist with 97.36 percentage accuracy. And it was able to recognize animals from 7 classes with 90.16 percent accuracy.

Later, we can connect an automated protection system depending on the recognized animal. As we can identify the animal in raspberry pi itself, we can use a different system like ultrasound frequency from raspberry pi to ward off the animal from the garden. We can use predator sound for the same purpose also.

Bibliography

[1] Khant Aaditya Arunkumar et al. "Wireless Speakers using WiFi and IoT". In: *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*. IEEE. 2019, pp. 1–6.

[2] NS Gogul Dev, KS Sreenesh, and PK Binu. "IoT Based Automated Crop Protection System". In: *2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. Vol. 1. IEEE. 2019, pp. 1333–1337.

[3] Khirod Chandra Sahoo and Umesh Chandra Pati. "IoT based intrusion detection system using PIR sensor". In: *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. IEEE. 2017, pp. 1641–1645.

[4] Hung Nguyen et al. "Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring". In: *2017 IEEE international conference on data science and advanced Analytics (DSAA)*. IEEE. 2017, pp. 40–49.

[5] Stefano Giordano et al. "IoT solutions for crop protection against wild animal attacks". In: *2018 IEEE international conference on Environmental Engineering (EE)*. IEEE. 2018, pp. 1–5.

[6] Alexander Maier, Andrew Sharp, and Yuriy Vagapov. "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things". In: *2017 Internet Technologies and Applications (ITA)*. IEEE. 2017, pp. 143–148.

[7] S Sruthy and Sudhish N George. "WiFi enabled home security surveillance system using Raspberry Pi and IoT module". In: *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*. IEEE. 2017, pp. 1–6.

[8] Shuying Liu and Weihong Deng. "Very deep convolutional neural network based image classification using small training sample size". In: *2015 3rd IAPR Asian conference on pattern recognition (ACPR)*. IEEE. 2015, pp. 730–734.

[9] Amartya Ranjan Saikia et al. "Comparative assessment of CNN architectures for classification of breast FNAC images". In: *Tissue and Cell* 57 (2019), pp. 8–14.

[10] Bangpeng Yao et al. "Human action recognition by learning bases of action attributes and parts". In: *2011 International conference on computer vision*. IEEE. 2011, pp. 1331–1338.

Report20

by Amit Biswas

Submission date: 26-Jul-2021 08:40PM (UTC+0530)
Submission ID: 1624311034
File name: Amit_Biswas.pdf (1.99M)
Word count: 4825
Character count: 23594

Report20

ORIGINALITY REPORT

3%	1%	2%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Khirod Chandra Sahoo, Umesh Chandra Pati. "IoT based intrusion detection system using PIR sensor", 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017 Publication	1%
2	Stefano Giordano, Ilias Seitanidis, Mike Ojo, Davide Adami, Fabio Vignoli. "IoT solutions for crop protection against wild animal attacks", 2018 IEEE International Conference on Environmental Engineering (EE), 2018 Publication	<1%
3	www.mitpressjournals.org Internet Source	<1%
4	circuitdigest.com Internet Source	<1%
5	randomnerdtutorials.com Internet Source	<1%
6	Submitted to University of Westminster Student Paper	<1%

Exclude quotes	On	Exclude matches	< 15 words
Exclude bibliography	On		