Safeguarding Agriculture: using Deep Learning to prevent Animal Invasions

by

Moulika K
Roll No. 204G1A0559

Bhavana C
Roll No. 204G1A0523

Anusha U
Roll No. 204G1A0514

Chandana T
Roll No. 204G1A0526

Under the guidance of

Mr. C. Lakshminatha Reddy **M.Tech, (Ph.D)**

Department of Computer Science and Engineering

Srinivasa Ramanujan Institute of Technology

Autonomous

Rotarypuram Village, B K Samudram Mandal, Ananthapuramu – 515701.

2023-2024

Contents

- Abstract
- Introduction
- Literature Survey
- Existing System
- Proposed System
- Planning
- Design
- Implementation
- Research Paper
- References

Abstract

Crop damage caused by animal attacks is one of the major threats in reducing the crop yield. Due to the expansion of cultivated land into previous wildlife habitat, crop raiding is becoming one of the most antagonizing human-wildlife conflicts. Traditional methods followed by farmers are not that effective and it is not feasible to hire guards to keep an eye on crops and prevent wild animals. Since safety of both human and animal is equally vital, it is important to protect the crops from damage caused by animal as well as divert the animal without any harm. By Using transfer learning models to detect the animals in the crops.

This project develops an algorithm to provide software to detect the animals in wildlife.

Keywords: Convolutional Neural Networks (CNN),VGG16 and VGG19.

Introduction

- In today's interconnected world, agriculture plays a vital role in ensuring food security and economic stability. However, one of the significant challenges faced by farmers worldwide is the invasion of their crops by pests and animals.
- These invasions not only result in substantial economic losses but also threaten food production and ecological balance. Traditional methods of pest control and animal deterrence often fall short in effectively addressing these challenges. Hence, there is a growing need for innovative solutions that can mitigate the risks posed by animal invasions.
- Deep learning, a subset of artificial intelligence, has emerged as a promising technology capable of revolutionizing various industries, including agriculture. By leveraging deep learning models, we can develop robust systems to monitor, detect, and prevent animal invasions in agricultural landscapes. This presentation aims to explore the potential of deep learning in safeguarding agriculture by providing early detection and proactive management of animal threats, ultimately ensuring the sustainability and resilience of our food systems.

Literature survey

- Wenling Xue [1] In this paper, a wireless sensor network based on UWB technology is used to deploy intrusion detection. By analyzing the characteristics of Ultra wide band (UWB) signals, convolutional neural network is used to learn the characteristics of UWB signals automatically. And finally the SVM or Softmax classifier is used to classify human beings from animals. Several experiments are tested in corn field and the experimental results show that the method proposed in this paper can detect human and animal intrusion very effectively and improve the accuracy of detection by nearly 16% compared to the traditional manual extraction.
- Sabeenian1, B. Mythili [2] Since safety of both human and animal is equally vital, it is important to protect the crops from damage caused by animal as well as divert the animal without any harm. Thus, in order to overcome above problems and to reach our aim, use machine learning to detect animals, entering into our farm by using deep neural network concept, a division in computer vision.

Literature survey

- Bala krishna [3] This paper presents the development of the Internet of Things and Machine learning technique-based solutions to overcome major threat to the productivity of the crops, which affects food security and reduces the profit to the farmers. Machine learning algorithms like Region-based Convolutional Neural Network and Single Shot Detection technology plays an important role to detect the object in the images and classify the animals.

- Ravoor [4] This research presents a novel end-to-end design of a distributed cross-camera tracking system (a "Digital Border") based on computer vision for the purpose of detecting animal trespass using deep learning networks. In addition to sending out notifications when an animal incursion is detected, the system provides useful data like the intruders' species and number, their approximate location, and their last known direction of movement.

Literature survey

- Krishnamurthy B, Divya M [5] In this project explains how agriculture provides for people's food needs and generates a variety of raw resources for industry. However, a significant loss of crops is anticipated due to animal interference in agricultural areas. A crop's vulnerability to wild animals exists. As a result, it's critical to keep an eye out for any animal presence nearby. After then, a number of devices should be activated to drive away the dangerous animals. We suggest a strategy to keep wild animals away from farms. The primary application of operational amplifier circuits is the detection of animal incursion from the exterior of farms. The purpose of the suggested monitoring programe is to give early notice of potential wild animal entry and harm. The Solar Electric Fence system is a cutting-edge substitute for traditional fencing techniques to safeguard your land and crops.

Existing System

The conflict between humans and animals is seen across the country in a variety of forms, including monkey menace in the urban areas, crop raiding by wild pigs and so on. Providing effective solutions for human-animals conflict is now one of the most significant challenges all over the world.

In this project we use machine learning methods. there are many applications of Machine learning, out of which Image Classification is one. To classify images, here we are using SVM Algorithm. However, there is a certain error between the actual output and the ideal output and the result is not accurate

Disadvantage:

- Low Accuracy
- High time complexity
- Slow process

Proposed System

We propose this application that can be considered a useful system .Since it helps to reduce the limitations obtained from existing methods. we are extend this project to deep learning and transfer learning models. First we collected some animal images then perform data pre-processing and image pre-processing steps. Convert all the images into array and reshape into one particular shape. Then split the data train and test and tried some different CNN architecture and Transfer learning models also.

The proposed system contains the following facilities over the present system:

1. By using transfer learning models we can correctly detect the animals.

2. And send email notification to the person when an animal entered into the filed.

Advantages :

- Accuracy is good.
- Low time complexity.
- Timely Notification

Proposed System

The objectives of this proposed system are

- Enhance Effectiveness: To enhance the effectiveness of safeguarding agriculture by leveraging deep learning and transfer learning models. By utilizing these advanced technologies, we aim to overcome the limitations of existing methods and provide a more robust solution for preventing animal invasions in agricultural fields.
- Data Collection and Pre-processing: Another objective is to collect a diverse dataset of animal images and perform comprehensive data pre-processing and image pre-processing steps. This involves converting the images into arrays, reshaping them into a standardized format, and ensuring data quality for training our models effectively.
- Model Evaluation and Selection: We seek to explore various convolutional neural network (CNN) architectures, VGG-16, and VGG-19 models. The objective here is to evaluate the performance of these models through rigorous training and testing procedures, ultimately selecting the most suitable model based on its accuracy and efficiency.

Proposed System

- Deployment and Notification: The final objective is to deploy the selected model into a practical system that can actively monitor agricultural fields. Once deployed, the system will use the chosen model to detect any animal intrusion and promptly notify farmers via email, enabling timely intervention to protect their crops.

By achieving these objectives, we aim to create a useful system that not only addresses the challenges of animal invasions in agriculture but also provides a reliable solution for farmers to safeguard their crops effectively.

Planning

Objective

- The main objective of the project is to safe guard the agricultural field from wild animals and also to protect them by driving them away instead of killing. The project also aims to protect human lives from animal attacks. We are using an integrative approach in the field of Deep Learning to provide a monitoring and repelling system for crop protection against animal attacks.

Scope

- The project encompasses experimenting with various CNN architectures and transfer learning models,VGG-16, and VGG-19. Through training and testing on both training and testing datasets, we assess the performance of these models to identify the most effective one for the task at hand. Regarding performance, the chosen final model exhibits high accuracy in detecting and categorizing animals in agricultural fields. This capability facilitates timely intervention, minimizing crop damage and losses. Furthermore, the system integrates a mail notification feature to promptly alert farmers when animals are detected in their fields, enabling swift action to be taken.

Planning

Functional Requirements

Image Collection and Preprocessing:

- Collect a diverse dataset of animal images relevant to agricultural contexts.
- Preprocess images to standardize format and quality for model training

Data Preprocessing:

- Convert images into arrays and reshape them into a standardized shape suitable for input into deep learning models.
- Perform data augmentation techniques to increase dataset diversity and robustness.

Model Training:

- Implement various CNN architectures and transfer learning models (VGG-16, VGG-19).
- Train models on the preprocessed image dataset using appropriate optimization algorithms.

Planning

Model Evaluation:

1. Evaluate the performance of trained models on both training and testing datasets.
2. Measure metrics such as accuracy to assess model effectiveness.

Model Selection and Tuning:

1. Select the most effective model based on evaluation metrics.
2. Fine-tune hyperparameters and architecture if necessary to optimize model performance.

Deployment:

1. Deploy the selected model into a production environment for real-time inference.
2. Integrate a notification system to alert farmers when animals are detected in their fields.

Planning

Non-Functional Requirements

Performance:

- The system should be able to process images and make predictions efficiently, with minimal latency, to ensure timely detection of animals in agricultural fields.

Scalability:

- The system should be able to handle a growing amount of data and user requests as the user base expands or the dataset size increases.

Reliability:

- The system should be highly reliable, with minimal downtime or errors, to ensure uninterrupted operation and timely notifications to farmers.

Planning

System Requirements:

- **Hardware Requirements:**

  RAM : 4GB

  Hard Disk : 500GB

  Processor: Intel i3

- **Software Requirements:**

  Operating System : Windows 10

  Programming Language : Python

  Libraries used : Pandas, NumPy

Planning

Methodology

➢The proposed methodology involves a combination of image processing techniques and deep learning algorithms, particularly Convolutional Neural Networks (CNNs), VGG16 and VGG19 to detect animals in agricultural fields. The following steps outline the approach.

System Modules:

- **Data pre-processing** : Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. When creating a deep learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way.
- **Normalizing numeric data**: Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges

Planning

- **Partitioning dataset** : Now we need to split our dataset into two sets a Training set and a Test set. We will train our deep learning models on our training set and then we will test the models on our test set to check how accurately it can predict. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a deep learning model which performs well with the training set and also with the test dataset. Here, we can define these data sets as:

Fig 1 : partitioning dataset for training and testing

- **Algorithms** : Aimed at safeguarding agriculture by preventing animal invasions using deep learning models, particularly CNN, VGG16, and VGG19 can be utilized.

Planning

Work Schedule

S.NO	Activity	Duration
1	Domain Selection	1 week
2	Abstract	1 week
3	Literature Survey	2 weeks
4	Planning	1 week
5	Design	2 week
6	Software Implementation	2 week
7	Testing	3 week
8	Execution	3 week

Design

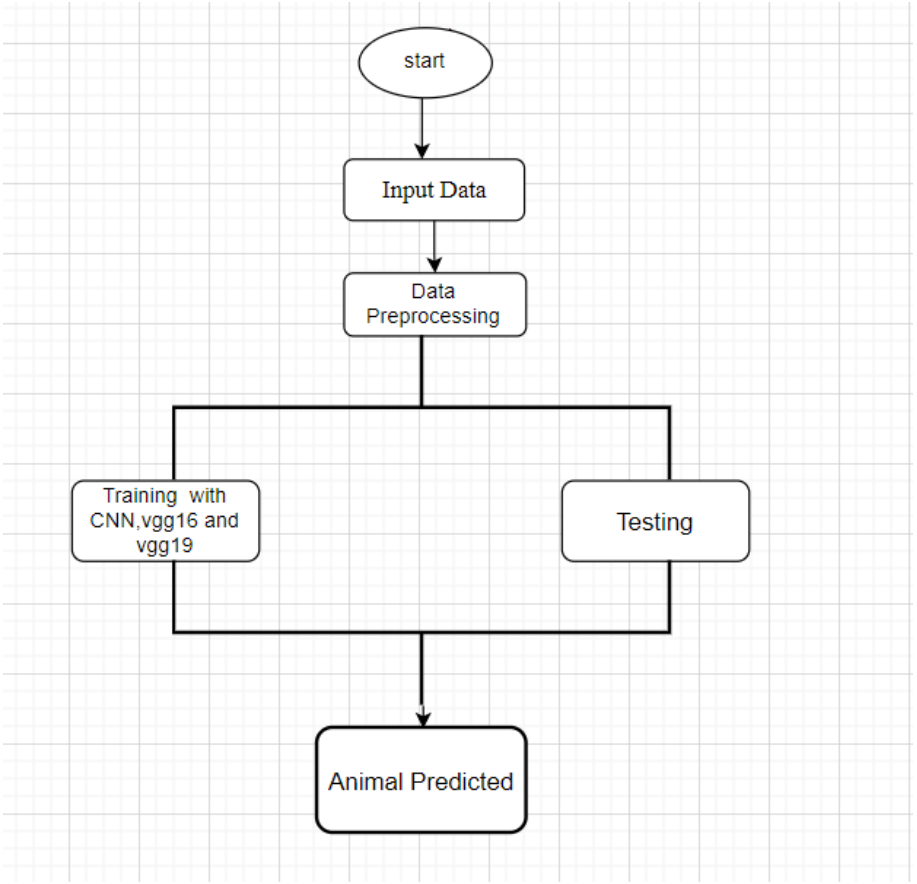


Fig 2: Architecture of proposed system

Design – Use Case Daigrams

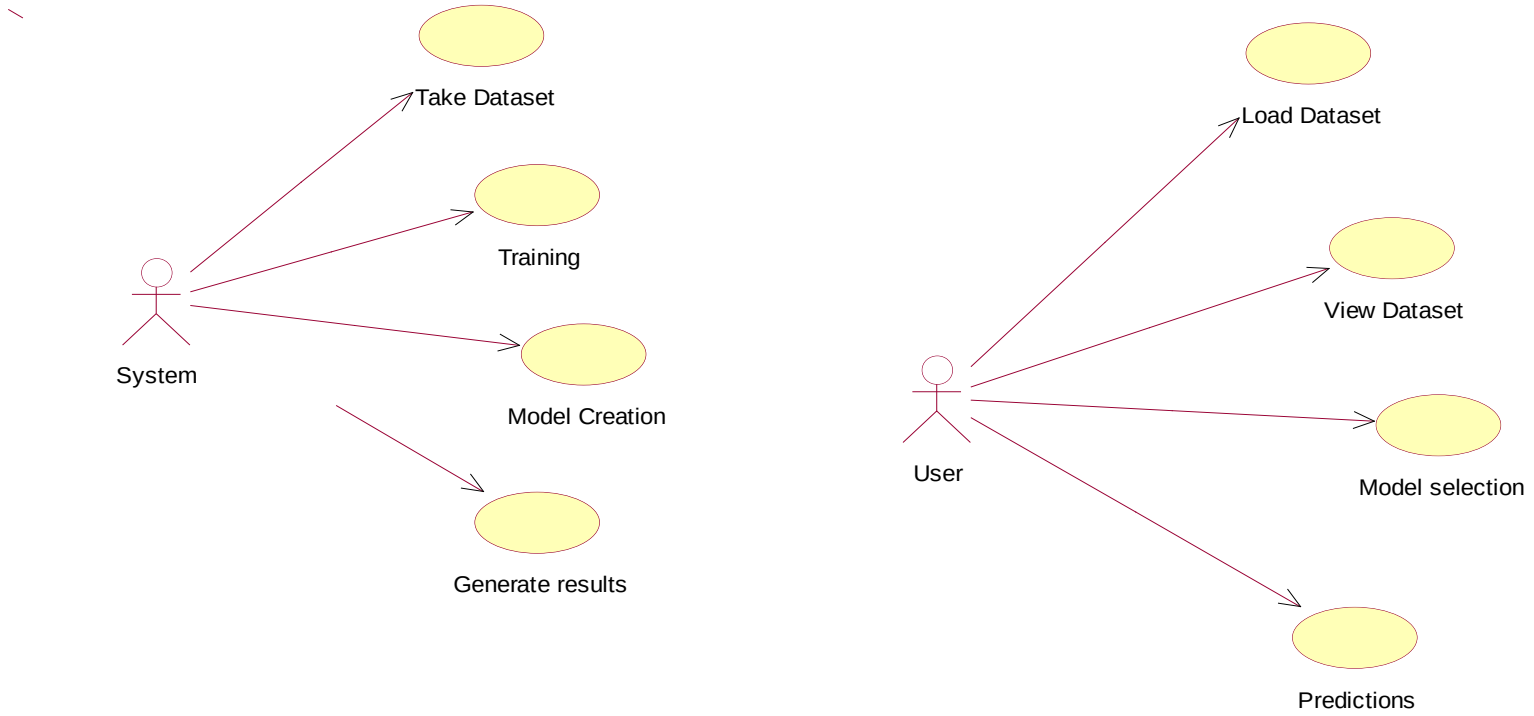


Fig 3: Use Case Daigrams

Implementation

Importing Libraries

```
[ ]  #list of useful imports that  I will use
     %matplotlib inline
     import os
     import tqdm
     import matplotlib.pyplot as plt
     import pandas as pd
     import cv2
     import numpy as np
     from glob import glob
     import seaborn as sns
     import random
     from keras.preprocessing import image
     import tensorflow as tf
     from tensorflow.keras.models import Model

     #from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
     from keras.models import Sequential
     from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, GlobalMaxPooling2D,Input
     #from keras.optimizers import RMSprop
     from keras.preprocessing.image import ImageDataGenerator

     from sklearn.model_selection import train_test_split
```

Implementation

Length of dataset

```
Images = []
import os
for dirname, _, filenames in os.walk(data):
    for filename in filenames:
        img = os.path.join(dirname, filename)
        Images.append(img)
```

```
Images[:10]
```

```
['Animal_data/Horse/images (1).jpg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.24.52 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.24.50 AM (1).jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.24.33 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.25.19 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.24.49 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.25.01 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.25.10 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.25.17 AM.jpeg',
 'Animal_data/Horse/WhatsApp Image 2022-07-06 at 12.24.58 AM (1).jpeg']
```

```
len(Images)
```

```
999
```

Class Label Count

```
# Shuffle two lists with same order
# Using zip() + * operator + shuffle()
temp = list(zip(Images, Class_label))
random.shuffle(temp)
Images, Class_label = zip(*temp)
```

```
data = pd.DataFrame(list(zip(Images, Class_label)), columns =['Image_path', 'Class_label'])
```

```
data.head(5)
```

	Image_path	Class_label
0	Animal_data/Cow/WhatsApp Image 2022-07-05 at 1...	Cow
1	Animal_data/Monkey/WhatsApp Image 2022-07-06 a...	Monkey
2	Animal_data/Tiger/WhatsApp Image 2022-07-06 at...	Tiger
3	Animal_data/Rabbit/WhatsApp Image 2022-07-06 a...	Rabbit
4	Animal_data/Monkey/WhatsApp Image 2022-07-06 a...	Monkey

Next steps: View recommended plots

```
data.Class_label.value_counts()
```

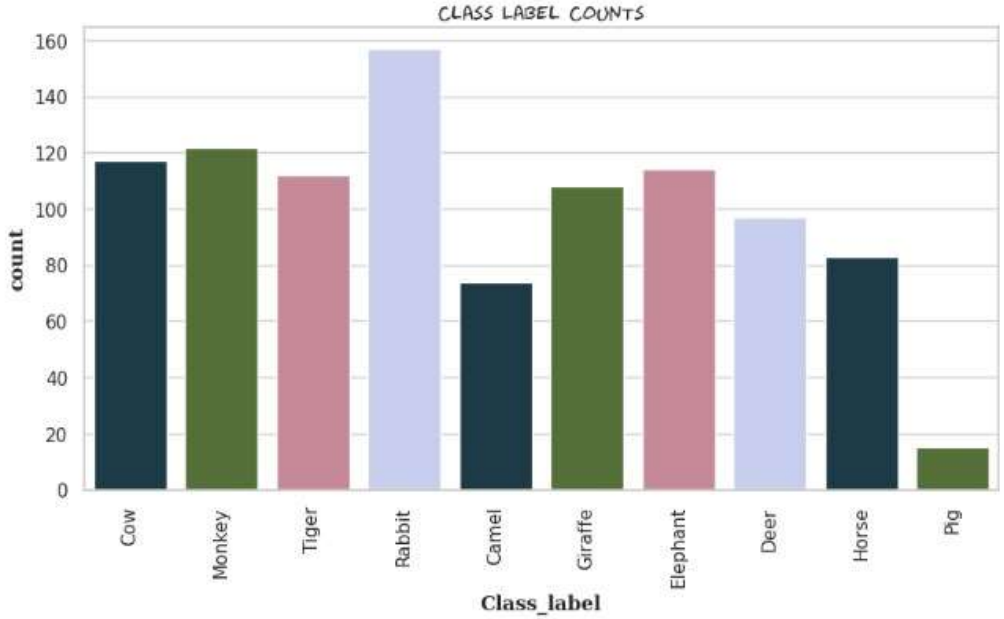
```
Rabbit      157
Monkey      122
Cow         117
Elephant    114
Tiger       112
Giraffe     108
Deer         97
Horse        83
Camel        74
Pig          15
Name: Class_label, dtype: int64
```

Implementation for CNN

Graph of class Label Count

```
sns.set(style="whitegrid")
plt.figure(figsize=(10, 5))
ax = sns.countplot(x="Class_label", data=data, palette=sns.color_palette("cubehelix", 4))
plt.xticks(rotation=90)
plt.title("Class Label Counts", {"fontname":"fantasy", "fontweight":"bold", "fontsize":"medium"})
plt.ylabel("count", {"fontname": "serif", "fontweight":"bold"})
plt.xlabel("Class_label", {"fontname": "serif", "fontweight":"bold"})
```

```
<ipython-input-17-ff59cf945e57>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  ax = sns.countplot(x="Class_label", data=data, palette=sns.color_palette("cubehelix", 4))
<ipython-input-17-ff59cf945e57>:3: UserWarning:
The palette list has fewer values (4) than needed (10) and will cycle, which may produce an uninterpretable plot.
  ax = sns.countplot(x="Class_label", data=data, palette=sns.color_palette("cubehelix", 4))
Text(0.5, 0, 'Class_label')
```



Implementation

Image resize

Resize the images

```
[22] def resize_images(img):
       file = Image.open(img)
       img = file.convert('RGB')
       img_bgr= img.resize((224, 224))
       img_bgr = np.array(img_bgr)
       return img_bgr
```

```
[23] from PIL import Image
```

```
[24] #save resized images into images.
     images = [resize_images(img) for img in data['Image_path']]
```

```
[25] # print number of classes in our dataset
     num_classes = len(np.unique(data['Class_label']))
```

```
[26] num_classes

     10
```

```
[27] # save the class into class_names
     class_names = list(data['Class_label'])
```

```
     # Print the shape of the image
     images[10].shape

     (224, 224, 3)
```

Splitting data for training and testing

Spliting data into training and testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(images, y, test_size=0.3,stratify = y,random_state=42)
```

```
X_train.shape

(1400, 224, 224, 3)
```

```
X_test.shape

(600, 224, 224, 3)
```

Implementation for CNN

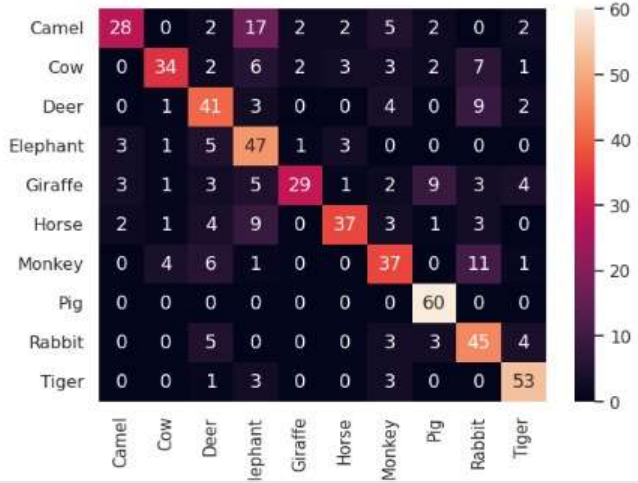
```
batch_size = None

model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu', batch_input_shape = (batch_size,224, 224, 3)))


model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.2))


model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.3))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.4))



model.add(GlobalMaxPooling2D())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))
model.summary()
```

Implementation for CNN

Convolution matrix

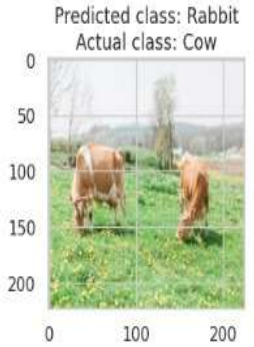
```
# Import necessary libraries
from sklearn.metrics import confusion_matrix
import seaborn as sns
import pandas as pd
import numpy as np

# Assuming your model is named 'model' and X_test, y_test are your test data
# Use the predict method to get probabilities for each class
y_pred_probs = model.predict(X_test)

# Use np.argmax to get the predicted class
y_pred_classes = np.argmax(y_pred_probs, axis=1)

# Get the true class labels
y_true_classes = np.argmax(y_test, axis=1)

# Get class names
class_names = enc.classes_

# Create a DataFrame for confusion matrix
df_heatmap = pd.DataFrame(confusion_matrix(y_true_classes, y_pred_classes), columns=class_names, index=class_names)

# Plot the confusion matrix
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")
```

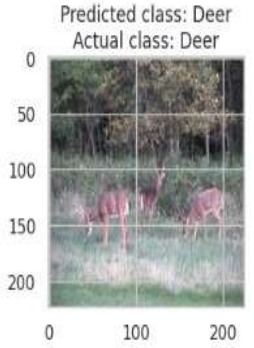
```
19/19 [==============================] - 1s 35ms/step
```



Predicted class using cnn

```
# print images with actual and predicted class labels
for i in range(20):
    plt.figure(figsize=(15, 15))
    plt.subplot(4, 5, i + 1)
    pred_probabilities = model.predict(np.array([X_test[i]]))[0]
    pred_class = np.argmax(pred_probabilities)
    actual_class = np.argmax(y_test[i])

    plt.title("Predicted class: {}\nActual class: {}".format(enc.classes_[pred_class], enc.classes_[actual_class]))
    plt.imshow(X_test[i])
```





Implementation for CNN

Accuracy for CNN

```
[ ]  # print the test accuracy
     score_1 = model.evaluate(X_test, y_test, verbose=0)
     print('Test accuracy:', score_1[1])
```

```
Test accuracy: 0.6850000023841858
```

Implementation for VGG16

Importing Libraries for VGG16 model

VGG_16 Model

```
# import the vgg16 model
from keras.applications.vgg16 import VGG16
```

```
vgg=VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 1s 0us/step
```

```
vgg.trainable=False
```

```
# Set the vgg16 model
model_1=Sequential()
model_1.add(vgg)
model_1.add(Flatten())
model_1.add(Dense(128, activation='relu'))
model_1.add(Dropout(0.2))
model_1.add(Dense(10, activation='softmax'))
```

Implementation for VGG16

Accuracy for VGG16

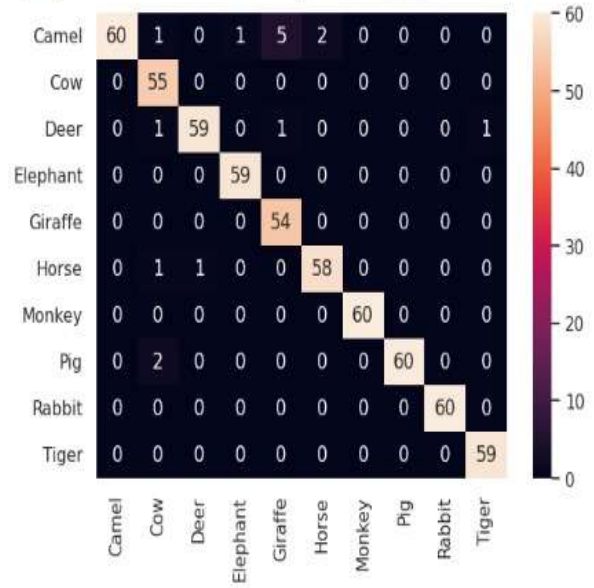
```
#Fit the data or train the model
History_1 = model_1.fit(X_train, y_train, epochs = 10, validation_data = (X_test,y_test),batch_size = 128)
```

```
Epoch 1/10
11/11 [==============================] - 71s 5s/step - loss: 70.1956 - accuracy: 0.5143 - val_loss: 1.0632 - val_accuracy: 0.9167
Epoch 2/10
11/11 [==============================] - 8s 767ms/step - loss: 1.0786 - accuracy: 0.9264 - val_loss: 0.3270 - val_accuracy: 0.9767
Epoch 3/10
11/11 [==============================] - 8s 781ms/step - loss: 0.4618 - accuracy: 0.9586 - val_loss: 0.4375 - val_accuracy: 0.9633
Epoch 4/10
11/11 [==============================] - 11s 1s/step - loss: 0.3261 - accuracy: 0.9621 - val_loss: 0.3989 - val_accuracy: 0.9650
Epoch 5/10
11/11 [==============================] - 11s 1s/step - loss: 0.4410 - accuracy: 0.9636 - val_loss: 0.3728 - val_accuracy: 0.9733
Epoch 6/10
11/11 [==============================] - 11s 1s/step - loss: 0.3061 - accuracy: 0.9693 - val_loss: 0.2987 - val_accuracy: 0.9733
Epoch 7/10
11/11 [==============================] - 11s 1s/step - loss: 0.1327 - accuracy: 0.9864 - val_loss: 0.9801 - val_accuracy: 0.9317
Epoch 8/10
11/11 [==============================] - 8s 793ms/step - loss: 0.7188 - accuracy: 0.9679 - val_loss: 0.4327 - val_accuracy: 0.9767
Epoch 9/10
11/11 [==============================] - 8s 795ms/step - loss: 0.7273 - accuracy: 0.9686 - val_loss: 0.3415 - val_accuracy: 0.9833
Epoch 10/10
11/11 [==============================] - 9s 798ms/step - loss: 0.1931 - accuracy: 0.9821 - val_loss: 0.5435 - val_accuracy: 0.9733
```

Convolution matrix for vgg16

```
#plot confusion matrix
from sklearn.metrics import confusion_matrix
predicted_classes = np.argmax(model_1.predict(X_test), axis=1)

class_names = enc.classes_
df_heatmap = pd.DataFrame(confusion_matrix(predicted_classes, np.argmax(y_test, axis=1)), columns=class_names, index=class_names)
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")
```

```
19/19 [==============================] - 3s 132ms/step
```



Implementation for VGG19

Importing Libraries for vgg19

VGG_19 Model

```
7] # import the vgg16 model
   from keras.applications.vgg19 import VGG19
```

```
8] vgg_19 = VGG19(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 1s 0us/step
```

```
9] vgg_19.trainable=False
```

```
0] # Set the vgg16 model
   model_2=Sequential()
   model_2.add(vgg_19)
   model_2.add(Flatten())
   model_2.add(Dense(128, activation='relu'))
   model_2.add(Dropout(0.2))
   model_2.add(Dense(10, activation='softmax'))
```

Implementation for VGG19

Accuracy for vgg19

```
1] #Compile the model
   from tensorflow.keras.optimizers import Adam,RMSprop
   opt = Adam(lr=0.001)
   optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08)
   model_2.compile(optimizer = optimizer, loss = "categorical_crossentropy", metrics = ["accuracy"])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
```

```
   #Fit the data or train the model
   History_2 = model_2.fit(X_train, y_train, epochs = 10, validation_data = (X_test,y_test),batch_size = 128)
```

```
Epoch 1/10
11/11 [==============================] - 12s 1s/step - loss: 33.9371 - accuracy: 0.5850 - val_loss: 0.6958 - val_accuracy: 0.9317
Epoch 2/10
11/11 [==============================] - 11s 1s/step - loss: 0.6160 - accuracy: 0.9471 - val_loss: 0.4063 - val_accuracy: 0.9700
Epoch 3/10
11/11 [==============================] - 11s 988ms/step - loss: 0.2394 - accuracy: 0.9743 - val_loss: 0.2249 - val_accuracy: 0.9817
Epoch 4/10
11/11 [==============================] - 12s 1s/step - loss: 0.0796 - accuracy: 0.9871 - val_loss: 0.3118 - val_accuracy: 0.9750
Epoch 5/10
11/11 [==============================] - 12s 1s/step - loss: 0.3203 - accuracy: 0.9743 - val_loss: 0.3102 - val_accuracy: 0.9750
Epoch 6/10
11/11 [==============================] - 12s 1s/step - loss: 0.3135 - accuracy: 0.9764 - val_loss: 0.3162 - val_accuracy: 0.9767
Epoch 7/10
11/11 [==============================] - 12s 1s/step - loss: 0.1613 - accuracy: 0.9850 - val_loss: 1.0430 - val_accuracy: 0.9483
Epoch 8/10
11/11 [==============================] - 12s 1s/step - loss: 0.8334 - accuracy: 0.9550 - val_loss: 1.4463 - val_accuracy: 0.9583
Epoch 9/10
11/11 [==============================] - 13s 1s/step - loss: 0.2819 - accuracy: 0.9871 - val_loss: 0.5887 - val_accuracy: 0.9717
Epoch 10/10
11/11 [==============================] - 12s 1s/step - loss: 0.3022 - accuracy: 0.9807 - val_loss: 0.4022 - val_accuracy: 0.9800
```

Implementation for VGG19

Convolution matrix for vgg19

```
#plot confusion matrix
from sklearn.metrics import confusion_matrix
class_names = enc.classes_
y_pred = model_2.predict(X_test)
predicted_classes = np.argmax(y_pred, axis=1)
true_classes = np.argmax(y_test, axis=1)

df_heatmap = pd.DataFrame(confusion_matrix(true_classes, predicted_classes), columns=class_names, index=class_names)
heatmap = sns.heatmap(df_heatmap, annot=True, fmt="d")
```

```
19/19 [==============================] - 3s 160ms/step
```

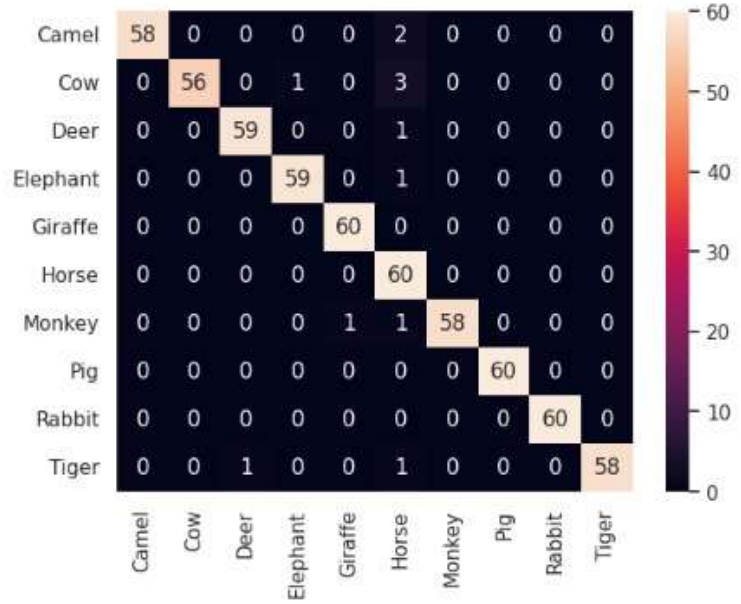


Image testing

Test image and mail notification

```
[82] # bounding box testing

     img='/content/drive/MyDrive/Animal_project/Test/download (2).jpg'

     file = Image.open(img)
     img = file.convert('RGB')
     img_bgr= img.resize((224, 224))
     img_bgr = np.array(img_bgr)
     img_bgr_reshape = img_bgr.reshape(1,224,224,3)
```

```
cnn = tf.keras.models.load_model('/content/drive/MyDrive/Animal_project (1)/Animal_data_cnn.h5')
vgg_16 = tf.keras.models.load_model('/content/drive/MyDrive/Animal_project (1)/Animal_data_vgg-16.h5')
vgg_19 = tf.keras.models.load_model('/content/drive/MyDrive/Animal_project (1)/Animal_data_vgg-19.h5')
```

Implementation

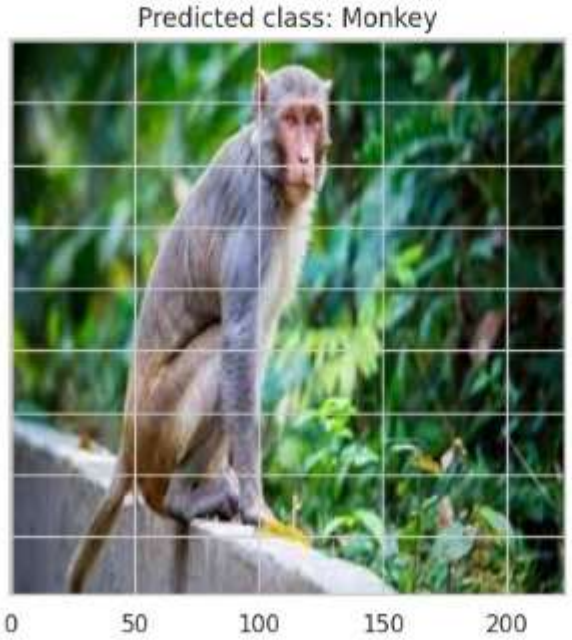
- Here we are using VGG19 model for predicting the class labels .Because, when compare to CNN and VGG16 models VGG19 have better accuracy i.e.,98% so we are using this model for predicting the output.

```
# Make predictions
predictions = vgg_19.predict(img_bgr_reshape)

# Find the index of the maximum probability
predicted_class_index = np.argmax(predictions)

# Get the predicted class label
predicted_class = enc.classes_[predicted_class_index]

# Display the prediction result
plt.title("Predicted class: {}".format(predicted_class))
plt.imshow(img_bgr)

import smtplib

# creates SMTP session
s = smtplib.SMTP('smtp.gmail.com', 587)

# start TLS for security
s.starttls()

# Authentication
s.login("tvidhyalatha2@gmail.com", "jshskjoprioogylx")

# message to be sent
TEXT = "{} came into your field".format(predicted_class)
SUBJECT = "Animal Intrusion Notification"


message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)


# sending the mail
s.sendmail("tvidhyalatha2@gmail.com", 'moulikakalimisetty9@gmail.com' ,message)

# terminating the session
s.quit()
```

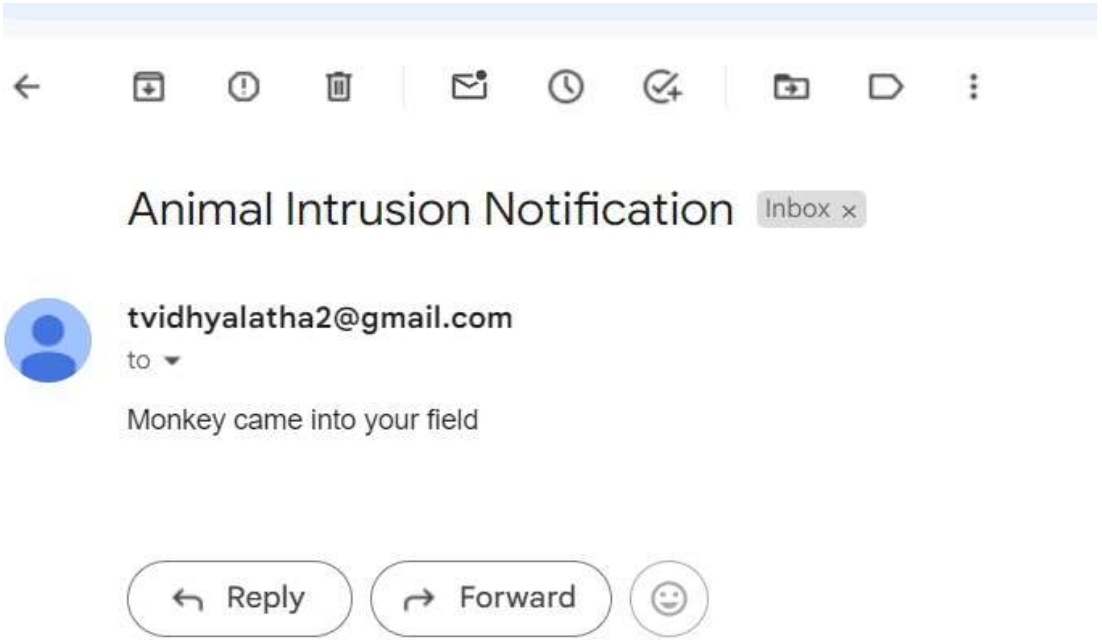
Implementation - Results

Animal Dectection

```
1/1 [==============================] - 0s 153ms/step
(221,
 b'2.0.0 closing connection o9-20020a056a00214900b006e4e616e520sm5020185pfk.72 - gsmtp')
```

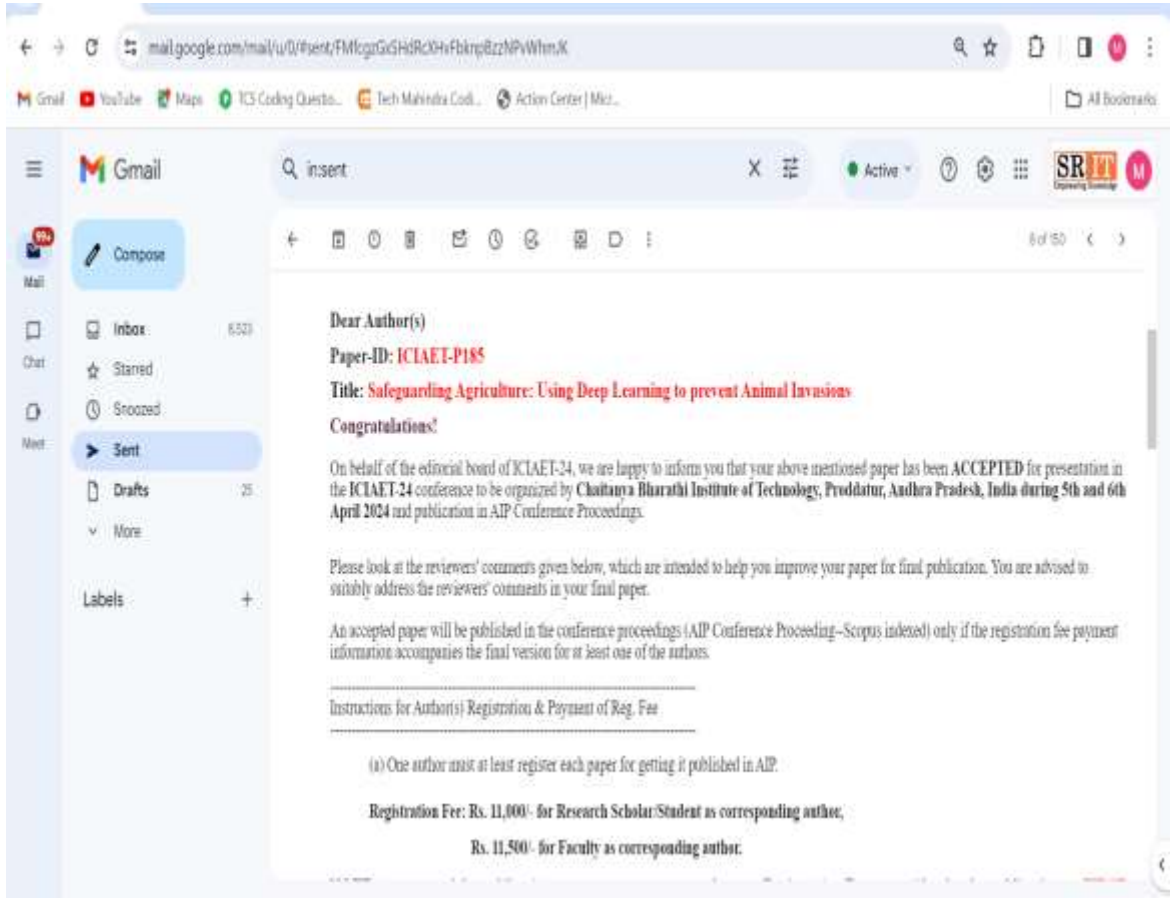


Mail Notification



Paper

Fig 4 : Acceptance of paper from CBIT and Payment proof



Gmail

Compose

Inbox

Starred

Snoozed

Sent

Drafts

More

Labels

Dear Author(s)

Paper-ID: ICIAET-P185

Title: Safeguarding Agriculture: Using Deep Learning to prevent Animal Invasions

Congratulations!

On behalf of the editorial board of ICIAET-24, we are happy to inform you that your above mentioned paper has been **ACCEPTED** for presentation in the **ICIAET-24** conference to be organized by **Chaitanya Bharathi Institute of Technology, Proddatur, Andhra Pradesh, India during 5th and 6th April 2024** and publication in AIP Conference Proceedings.

Please look at the reviewers' comments given below, which are intended to help you improve your paper for final publication. You are advised to suitably address the reviewers' comments in your final paper.

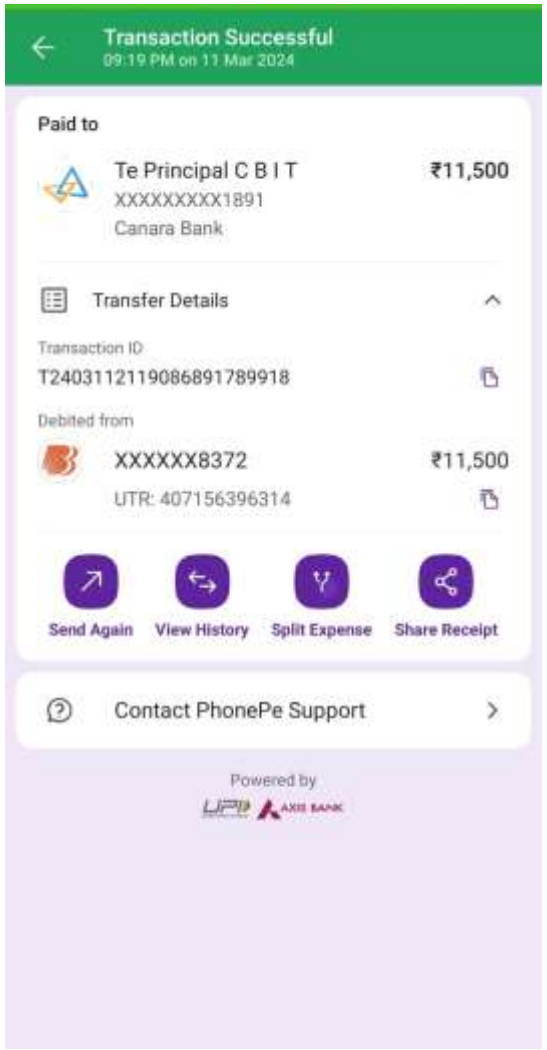
An accepted paper will be published in the conference proceedings (AIP Conference Proceeding–Scopus indexed) only if the registration fee payment information accompanies the final version for at least one of the authors.

Instructions for Author(s) Registration & Payment of Reg. Fee

(a) One author must at least register each paper for getting it published in AIP.

Registration Fee: Rs. 11,000/- for Research Scholar/Student as corresponding author,

Rs. 11,500/- for Faculty as corresponding author.



Transaction Successful

09:19 PM on 11 Mar 2024

Paid to

Te Principal C B I T ₹11,500

XXXXXXXX1891

Canara Bank

Transfer Details

Transaction ID

T240311211908689178991 8

Debited from

XXXXXX8372 ₹11,500

UTR: 407156396314

Send Again View History Split Expense Share Receipt

Contact PhonePe Support

Powered by

Conclusion

- In the current day, the issue of wild animals consuming crops has grown to be quiet important. Put differently, every farmer should utilize the crop productivity that he or she should be conscious of and mindful of the reality that animals are sentient beings who require protection from potential harm. It needs to be addressed immediately and effectively. As a result, this initiative has a great deal of social significance because it will free farmers from the needless labor associated with field protection, assist them in safeguarding their farms and preventing them form suffering large financial damages.

References

[1]. Xue, Wenling, Ting Jiang, and Jiong Shi. Animal intrusion detection based on convolutional neural network. 2017 17th international symposium on communications and information technologies (ISCIT). IEEE, 2017.

[2]. Sabeenian, R., N. Deivanai, and B. Mythili. Wild animals intrusion detection using deep learning techniques. *Int. J. Pharm. Res* 12.4 (2020): 1053-1058.

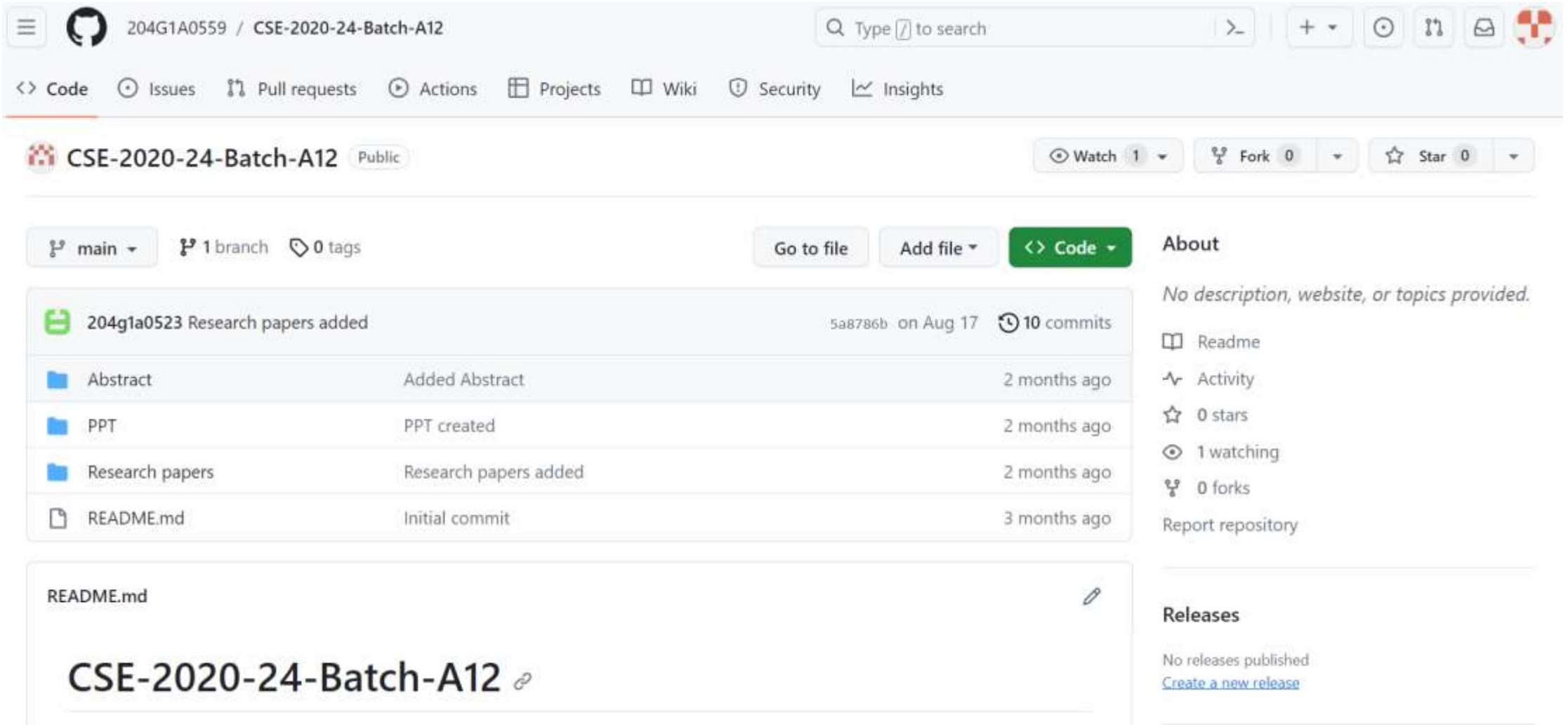
[3]. Balakrishna, K., et al. Application of IOT and machine learning in crop protection against animal intrusion. Global Transitions Proceedings 2.2 (2021): 169-174.

[4]. Ravoor, Prashanth C., T. S. B. Sudarshan, and Krishnan Rangarajan. Digital Borders: Design of an Animal Intrusion Detection System Based on Deep Learning. *International Conference on Computer Vision and Image Processing*. Singapore: Springer Singapore, 2020.

[5]. Krishnamurthy B, Divya M. Solar Fencing Unit and Alarm for Animal Entry Prevention. International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137, Volume – 02, Issue – 05, May – 2017, PP – 128-135.

Git Hub Dashboards of each student

Github link: https://github.com/204G1A0559/CSE-2020-24-Batch-A12



Thank You!!!