

A Project Report on

**CROP PROTECTION USING DEEP LEARNING TECHNIQUES**

*Submitted by*

Deivanai N

Gowsalya P

Soujanya S

Sripriya S

*Submitted to*

SMARTBRIDGE

*in partial fulfillment for the completion of one month internship*

*in*

Python in Machine Learning

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

**SONA COLLEGE OF TECHNOLOGY**

SALEM-636005.

(BATCH 2016-20)

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter No:** | **Heading No:** | **Topic:** | **Page No:** |
| -- | -- | **ABSTRACT** | 3 |
| 1. |  | **INTRODUCTION** | 3 |
|  | 1.a | Computer Vision | 3 |
|  | 1.b | Machine Learning | 4 |
|  | 1.b.1 | *Supervised Learning* | 5 |
|  | 1.c | Deep Learning | 5 |
|  | 1.c.1 | *Convolutional Neural Networks* | 6 |
| 2. |  | **PREVIOUS WORKS AND APPROACH TO THE PROBLEM** | 6 |
|  | 2.a | The Problem Prevailing | 6 |
|  | 2.b | Previous Approaches on this Problem | 7 |
|  | 2.c | Approaching the Problem Using Deep Learning | 7 |
| 3. |  | **METHODOLOGY** | 7 |
|  | 3.a | Tools Used | 7 |
|  | 3.b | Data Collection | 8 |
|  | 3.c | Libraries Used | 8 |
|  | 3.d | Creation of the model | 9 |
|  | 3.d.1 | *Process involved in Convolutional Neural Network* | 9 |
|  | 3.d.2 | *Training the dataset using Artificial neural Network* | 10 |
|  | 3.d.3 | *Prediction of animal entering the field* | 11 |
| 4. |  | **CONCLUSION** | 12 |
| 5. |  | **SUMMARY** | 13 |
| 6. |  | **REFERNCES** | 13 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No:** | **Figure Name:** | **Page No:** |
| 1.1 | Computer Vision of an image with cars | 3 |
| 1.2 | Types of Machine Learning and their uses in different fields | 4 |
| 1.3 | Process involved in Supervised Learning | 5 |
| 1.4 | *The layers in Convolutional Neural Network in different era* | 6 |
| 2.1 | Raspberry pi with Camera module interfacing | 7 |
| 3.1 | Full Connection | 9 |

Project Report on

**CROP PROTECTION USING DEEP LEARNING TECHNIQUES**

**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 conflicts antagonizing human-wildlife relationships. Farmers in India face serious threats from pests, natural calamities & damage by animals resulting in lower yields. 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.*

*Thus, in order to overcome above problems and to reach our aim, we use machine to detect animals, entering into our farm by using deep neural network concept, a division in computer vision. In this project, we will monitor the entire farm at regular intervals through a camera which will be recording the surrounding throughout the day. With the help of a machine learning model, we detect the entry of animals and we play appropriate sounds to drive the animal away. This report specifies various libraries and concepts of convolutional neural networks used to create the model.*

**1.INTRODUCTION**

***1.a COMPUTER VISION:***

Computer vision is an [interdisciplinary field](https://en.wikipedia.org/wiki/Interdisciplinarity) that deals with how computers can be made to gain high-level understanding from [digital images](https://en.wikipedia.org/wiki/Digital_image) or [videos](https://en.wikipedia.org/wiki/Video). It is like imparting human intelligence and instincts to a computer. In reality though, it is a difficult task to enable computers to recognize images of different objects .From the perspective of [engineering](https://en.wikipedia.org/wiki/Engineering), it seeks to automate tasks that the [human visual system](https://en.wikipedia.org/wiki/Human_visual_system) can do. "Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding."

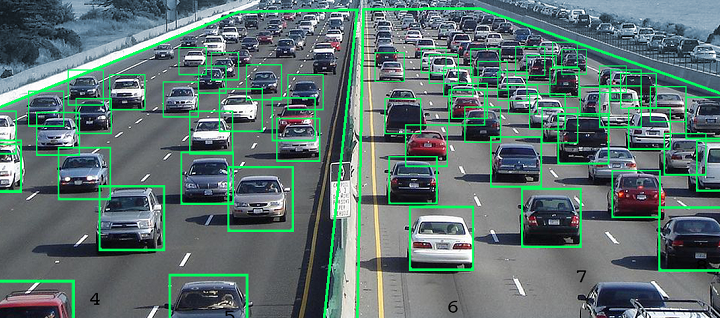


Fig 1.1 Computer Vision of an image with cars

***1.b MACHINE LEARNING:***

Machine learning is an approach that helps analysts to develop the models and algorithms that can find patterns and to apply them in a practical manner. It is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. It focuses on the development of computer programs that can access data and use it learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Types of Machine Learning are:

1. Supervised Learning
2. Unsupervised Learning
3. Semi supervised Learning
4. Reinforcement Learning

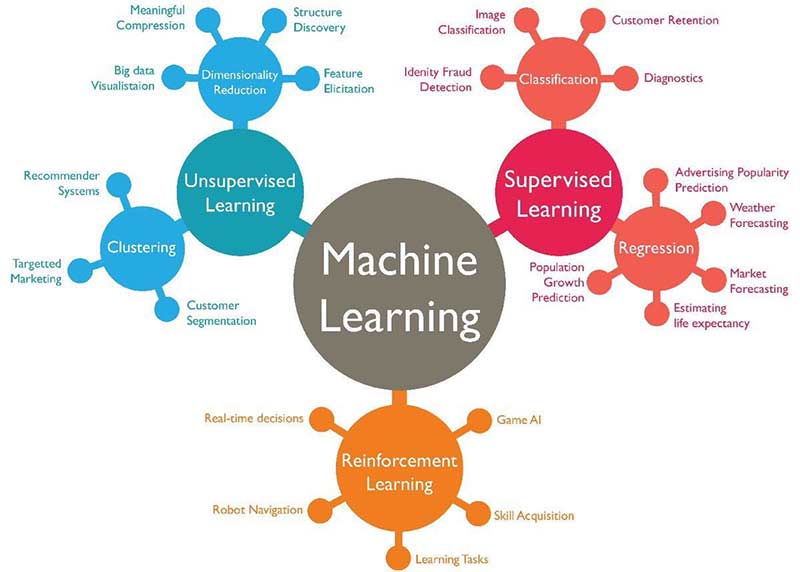


Fig 1.2 Types of Machine Learning and their uses in different fields

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

*1.b.1 SUPERVISED LEARNING:*

In the project, we use labelled data and hence we use Supervised Learning Concept.

Supervised machine learning algorithms can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

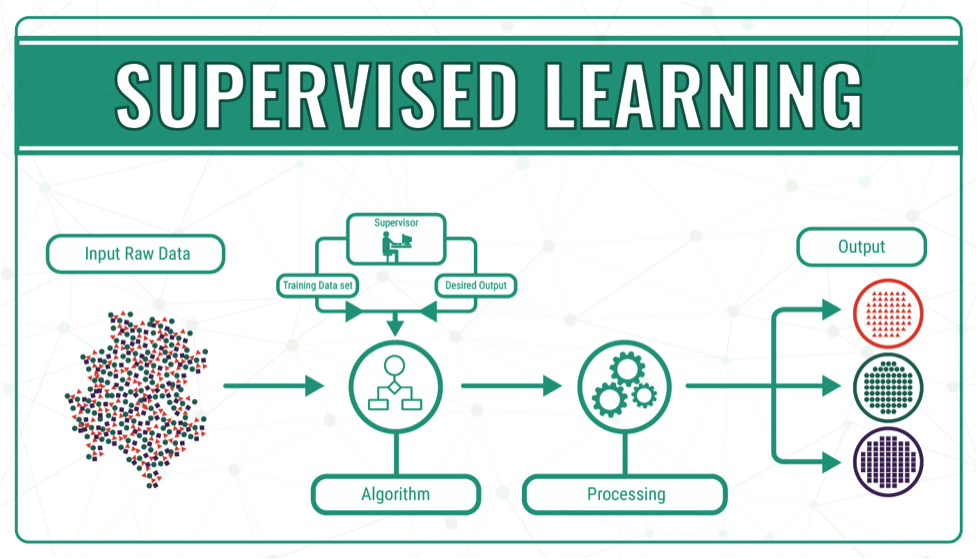


Fig 1.3 Process involved in Supervised Learning

***1.c DEEP LEARNING:***

Deep learning focuses on the important point about scale, that as we construct larger neural networks and train them with more and more data, their performance continues to increase. This is generally different to other machine learning techniques that reach a plateau in performance. In addition to scalability, another often cited benefit of deep learning models is their ability to perform automatic feature extraction from raw data, also called feature learning. *DEEP LEARNING AS A SCALABLE LEARNING ACROSS DOMAINS-*Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are images of pixel data, documents of text data or files of audio data.

The most popular techniques are:

1. Multilayer Perceptron Networks.
2. Convolutional Neural Networks.
3. Long Short-Term Memory Recurrent Neural Networks.

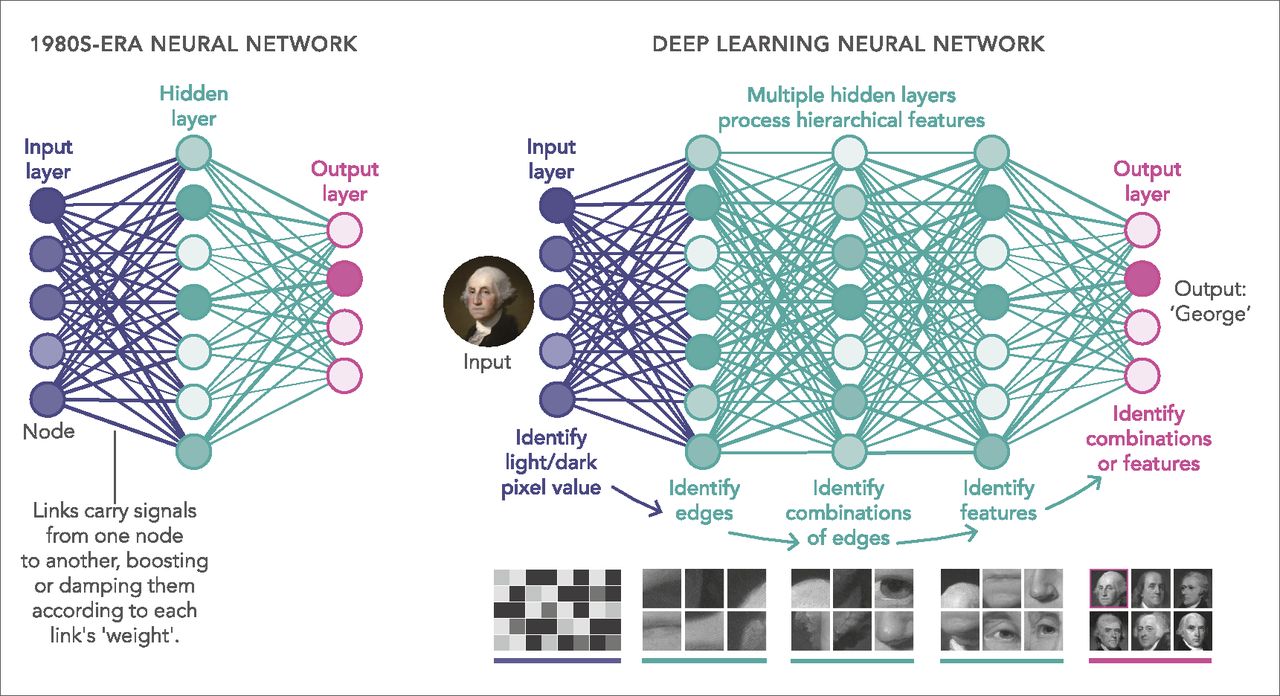
*1.c.1 CONVOLUTIONAL NEURAL NETWORKS(CNN):*

In the project, we use Convolutional Neural Network technique.

CNNs are powerful image processing, Artificial Intelligence that use Deep learning to perform both generative and Descriptive tasks, often using machine vision that includes image and video recognition, along with recommender systems and Natural Language Processing(NLP).

Traditional Neural Network are not ideal for Image processing and must be fed images in reduced-resolution pieces. CNN have their neurons arranged more like those of frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of Traditional neural network.

A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.



*Fig 1.4 The layers in Convolutional Neural Network in different era*

**2. PREVIOUS WORKS AND APPROACH TO THE PROBLEM:**

***2.a THE PROBLEM PREVAILING:***

A nation-wide study of the human-wildlife conflict around wildlife reserves across the country has highlighted the need for a comprehensive evaluation of the current mitigation strategies as despite widespread use of protection measures for crops and livestock, many households continued to experience losses.

The three-year study long conducted around 11 reserves has found that 71 percent of the households surveyed had suffered crop loss, and 17 percent livestock loss. Besides, three per cent of the households had members who had been either killed or injured because of animal attack. The survey had covered 5,196 households living in 2,855 villages at different distances from the boundaries of the reserves.

***2.b PREVIOUS APPROACHES ON THIS PROBLEM:***

Manual ways such as constructing different kinds of fences and using natural repellents are effective but they are not cost efficient. It is also not possible to increase man power for this problem.

So, initially projects were taken up to drive away the animals automatically by using hardware components like controllers and sensors. One such approach is - *Camera interfaced to the raspberry pi module*. Camera is used to captures an image of wild animal and send captured image to the Raspberry pi module. When image taken by the raspberry pi, it is compared with database image. After comparing images if the wild animal is detected then it gives commands to GSM module. GSM used to send the message to the owner of the farm. To get the output in the form audio, connect raspberry pi to the speaker. Disadvantage: The problem here is there is more of hardware component which is not cost efficient and its maintenance is also hard.

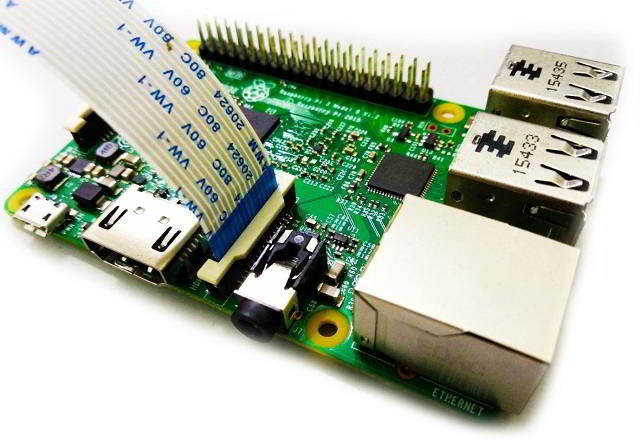


Fig 2.1 Raspberry pi with Camera module interfacing

In recent times, researches are taken to solve this problem using Artificial Intelligence. One of the already proposed idea was WCoHOG.

WCoHOG is a Histogram oriented gradients-based feature vector. LIBLINEAR classifier is used in order to get better accuracy for high dimensional data. The experiments were conducted on two benchmark datasets called Wild-Animal and CamaraTrap dataset. But it is a complex process. This feature vector supplied to the classifier to detect the animal in a particular window. In sliding window technique different sliding windows used in order to identify animals in different sizes of animal and zoom level of the camera. In the CoHOG method, gradient directions are used to calculate feature vector and the magnitude is ignored.

***2.c APPROACHING THE PROBLEM USING DEEP LEARNING:***

As the above problem stated is still prevailing despite of all the methods taken, we approached the problem using deep learning to drive away the animals automatically. In our project, we used packages like *keras* and *playsound* to do the pre-processing steps involved and to create an appropriate output depending on the detected output from the model.

**3.METHODOLOGY:**

***3.a TOOLS USED:***

*Personal Computer* - This is the vital component of the model. Using jupyter notebook, we run the code that is created to detect animals.

*Jupyter notebook* - The Jupyter Notebook is an open-source web application that allows data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

Camera - Secondary camera used to record the field area live 24/7 and this is fed as an input to the model to detect animals.

Speaker - To amplify the sound that is received from the model to the field area to drive away animals.

***3.b DATA COLLECTION:***

The dataset used as input for this project is image dataset. We have three classes namely elephant, cow and monkey as we take these animals as threat to farms in South India. We divided the dataset to two types, one for testing and another for training.

The training dataset includes:

Monkey – 318 images

Elephant – 449 images

Cow – 413 images

The testing dataset includes:

Monkey – 82 images

Elephant – 84 images

Cow – 84 images

The input to the model is received from the camera which records 24/7 and the received video is converted to images for further processing with the training dataset.

***3.c LIBRARIES USED:***

The libraries imported from the above packages that we used in the program are:

***1.Sequential***: Imported from *keras.models* package. The sequential API allows you to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs. The Sequential model API is great for developing deep learning models. Models are defined by creating instances of layers and connecting them directly to each other in pairs, and then defining a model that specifies the layers to act as the **input** and **output** to the model, via the parameters inputs and output respectively.

***2.Dense***: Imported *keras.layers* package implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True).

***3.Convo2D****:* Imported from keras.layers package. This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activationis not None, it is applied to the outputs as well. When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers, does not include the batch axis), e.g. input\_shape= (128, 128, 3) for 128x128 RGB pictures in data\_format="channels\_last".

***4.MaxPooling****:*Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling. ***5.Flatten****:* As the name of this step implies, we are literally going to flatten our pooled feature map into a column. What happens after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further.

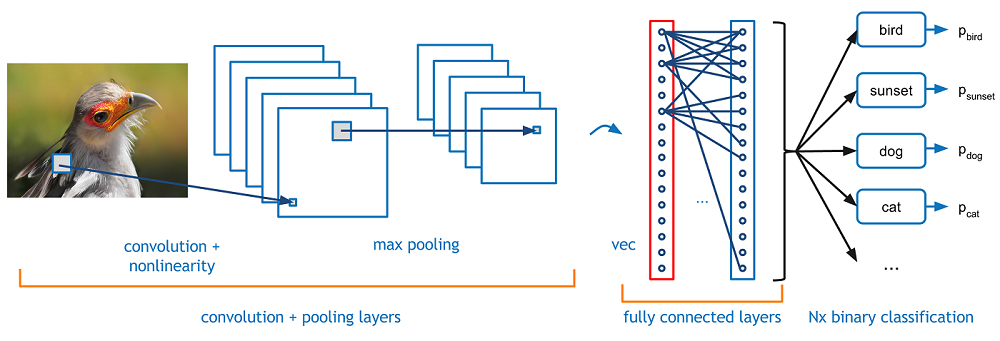


Fig 3.1 Full Connection

***6.ImageDataGenerator:*** For this library, we have many arguments to be passed but in our project we use:

*keras.preprocessing.image.ImageDataGenerator(shear\_range=0.0, zoom\_range=0.0, horizontal\_flip=****False/True****, rescale=****None****)*

🡪shear\_range: Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees)

🡪horizontal\_flip: Boolean. Randomly flip inputs horizontally.

🡪rescale: rescaling factor. Defaults to None. If None or 0, no rescaling is applied, otherwise we multiply the data by the value provided (after applying all other transformations).

🡪zoom\_range: Float or [lower, upper]. Range for random zoom. If a float, [lower, upper] = [1-zoom\_range, 1+zoom\_range].

***7.Resize*:** It is imported from skimage.transform. Resize operation resizes an image by a given specific output image shape. The image shape is two-dimentional- one along each axis. Note that when down-sampling an image, resize and rescale should perform Gaussian smoothing to avoid aliasing artifacts.

***3.d CREATION OF THE MODEL:***

*3.d.1 PROCESS INVOLVED IN CONVOLUTIONAL NEURAL NETWORK:*

The steps involved in CNN process are:

1. Initially, we import the libraries ***Sequential, Dense, Convo2D, MaxPooling and Flatten*** libraries from *keras.models and keras.layers* packages.
2. Initializing of the model, by calling the constructor – Sequential
3. To add convolution layer to the Neural Network we use the ‘add’ function along with ‘Convo2D’

In our project we used

*cnn\_model.add(Conv2D(32,3,3,input\_shape=(64,64,3)))*

32 denotes the number of feature detectors to be applied.

3,3 is the shape of the feature detector

*input\_shape* defines the size of the image and no of channels (colour images-3,grey scale images-1). As this is the first layer, we used *input\_shape* parameter. If we want to perform convolution more than once, this parameter is not necessary from the second cycles.

1. Next we add the pooling layer by using *add* function along with *MaxPooling2D*

*cnn\_model.add(MaxPooling2D(pool\_size=(2,2)))*

The pool size represents the filter used in the convoluted image.

1. Again, another set of Convolution and MaxPooling can be performed to increase the efficiency of the model.

In the project, we performed the Convolution and MaxPooling process thrice.

1. After MaxPooling, we flatten the shape of the image into a single dimension/array.

The output from the flatten function is given as the input layer to our future ANN model.

This step is important because the ANN model accepts input only in single dimension.

*3.d.2 TRAINING THE DATASET USING ARTIFICIAL NEURAL NETWORK:*

The steps involved in training process is:

1. Now, to establish full connection we already have our input layer from the Convolution Network, next we have to add hidden layers for processing the image by the hidden layer neurons. Increase in hidden layers increases the accuracy of the output.

In our project, we add only one hidden layer by using the *Dense* library *cnn\_model.add(Dense(init="random\_uniform",activation="relu",output\_dim=250))*

🡪Init is random uniform here, which is one of the weight optimization strategy

🡪The activation function used here is relu to omit the negative values.

🡪And the output dimension is the number of outputs from the hidden layer

1. After the hidden layer, we add the final output layer.

*cnn\_model.add(Dense(init="random\_uniform”, activation="softmax", output\_dim=3))*

Here the activation function used is ‘softmax’ for categorical data as our output consists of more than two classes (output dimension as 3). ‘Sigmoid’ activation function is used if our output is binary (consisting of two classes only)

1. After adding all layers, we compile the model.

*cnn\_model.compile(optimizer="adam",loss="categorical\_crossentropy",metrics=["accuracy"])*

🡪Optimizer - *adam* which is one of the weight optimization strategy

🡪Loss- *categorical\_crossentropy* to calculate the error

🡪Metrics – *accuracy* to calculate the model performance

1. In order to avoid over fitting of the various images, we need to transform the image. So, we use *ImageDataGenerator* to apply pre-processing steps to the images. For both training and testing dataset, we use pre-processing features like

🡪Rescaling – 1. /255 is used to apply feature scaling to get values in range of 0 to 1

🡪Shear\_range – 0.2 performs geometrical operation to move images in some particular angle

🡪Zoom range and horizontal flip. By using all these pre-processing features, we obtain different images with variations and hence avoid overfitting.

1. Next, we call the ‘flow\_from\_directory’ function which helps to import data and apply transformations simultaneously

The arguments that has to be passed are:

🡪The file path where the training/testing dataset is present

🡪Target size – size of the image (we used 64x64 dimension)

🡪Batch size – as we use mini batch gradient descent to update the weights, we give batch size i.e. the number of rows after which the weights are updated

🡪Class mode - categorical as we have more than two classes (cow, elephant and monkey)

We perform this function for both training and testing datasets.

1. Use of class\_indices to know the index of each class. This will be helpful when we check the prediction of the testing data.

*{cow : ‘0’, elephant: ‘1’, monkey: ‘2’)*

1. Call the *fit.generator* function to train and test the model simultaneously, to provide validation accuracy.

The arguments passed are,

*x\_train, steps\_per\_epoch = number of images in training dataset/batch\_size, epoch=number of epoch i.e. number of times the model should be trained, validation\_data= x\_test, validation\_steps=number of images in testing dataset/batch\_size.*

The epoch will run and finally we get the validation accuracy and loss as output from this function which denotes how well our model is trained (depending on validation accuracy) and the error produced by the model. More number of epoch will increase the accuracy of the output. In this project we use 40 epochs and received a validation accuracy of 82% and loss of 0.738.

1. Saving the model as ‘mymodel.h5’ using *model.save* function.

Thus, the training and testing dataset undergoes the above processes and the images from this model is compared with the live images from the camera to predict whether the animal is entering the fields are not. So, for doing this we again have to load this model to compare it with our frames from camera.

*3.d.3 PREDICTION OF ANIMAL ENTERING THE FIELD:*

The steps involved are:

1. To get the frames from the live camera, we need to first load the model in order to compare the frames with the trained model.

For that first we import load\_model, numpy and cv2(OpenCV). Next, we load the model that we saved earlier(‘mymodel.h5’). We then compile the model by giving the optimizer as *adam* loss as *categorical\_crossentropy* and metrics as *accuracy* similar to the way we compiled the full connection.

1. To transform the image, we resize the image into particular dimension.

In the project, we create a function *detect(frame)* to resize the image so that we can call the function at appropriate places. We use try and except method, wherein in try, I resize my image into 64x64 dimension and am expanding the dimension. If the ranges of the image are greater than one, then we rescale it to range 0 to1 and we predict the output using ‘predict’ function. Then, to know the class in which the image belongs, we use predict\_classes.

Shape not found is printed if there is an Attribute Error i.e. if my testing data/frame doesn’t match the training set we get this indication.

1. To know whether the prediction is right or not, initially we give a testing image to find whether it belongs to the appropriate class or not

Example if my class indices are *{cow : ‘0’, elephant: ‘1’, monkey: ‘2’}*

Then when I give a cow image, my output will be [0] and if my test image is a goat, then it should return ‘shape not found’

For this, we have to read the image using *imread* and call the detect function.

1. To detect the animals in the live video, we have to divide it into frames. For this, we read the video from the folder and then we create a directory to save all the frames that is created. We name the frames and then we read each frame using *imread* and we call the *detect* function to find whether there is an animal in that particular frame.

Then we give the appropriate sound according to the animal detected by importing playsound and by using else-if case by giving the class indices as condition. We used crackers and high pitch sounds to drive the animal away. After every frame is read from the live video the space and all the windows are released at once.

***Output:*** We get an indication of the frame creation from the video then the probability of the particular frame with the training classes i.e. the probability of how well the frame is matched with the classes and finally the class index of the animal detected along with the sound to drive them away. An amplifier is used to amplify the sound produced from the model.

**4.CONCLUSION:**

The problem of damaging crops by wild animals has become a major social problem in the current time. It requires urgent attention and an effective solution. Thus, this project carries a great social relevance as it aims to address this problem. Thus, this project is built using methods for acquiring, processing, analysing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information.

We are using an integrative approach in the field of deep learning to overcome this.

The goal of this work is to provide a repelling and monitoring system for crop protection against animal attacks.

In our future work, we will extend the current functionalities of our model like increasing the dataset so as to achieve high accuracy and investigate the chance of incorporating the features of our model to other sectors.

**5.SUMMARY**

First, we introduce the Convolution Neural Network concept to extract features from the images by creating feature map and then flattening into single dimension. This output from flattening is given as input to the Artificial Neural Network to establish full connection. Then, we compile the model to get the accuracy of the model. ImageDataGenerator will generate different images of training and testing dataset in different directions/angles. Then by using fit\_generator we run the model equal to the number of epochs given to get the validation accuracy and loss of the model. WE then save and load the model later to predict the output of the testing image. Then, the live video is converted into frames, each frame is compared with the trained model and predicts the entry of the animal until the completion of the video or the interruption from the keyboard to quit the detection process. If the animal is detected then we play an alarming sound to drive away the animal which is the ultimate motive of the project.

**6.REFERENCES:**

1. <https://www.kaggle.com/alessiocorrado99/animals10>

2. [https://keras.io](https://keras.io/)

3. [www.superdatascience.com](http://www.superdatascience.com)

4. [www.slideshare.net](http://www.slideshare.net)