Insect detection in chili and cotton crops using convolution neural network

Anilkumar Dr.Ponmaniraj

Department of computer science and engineering ,Saveetha School of Engineering SIMATS ,CHENNAI,

Tamilnadu .

Email of student : [na9918588@gmail.com](mailto:na9918588@gmail.com)

Email of guide :

Abstract :-

Aim:- To conduct a comparative analysis of machine learning and algorithmic approaches for insect

detection in agricultural corps ,with the goal of identifying the most effective and efficient methods

for early pest detection and management in the context of precision agriculture

Materials and methods:-

Researchers employ various classifiers, including SVM, Naïve Bayes, KNN, and CNN [1], for the

Purpose of conducting classification tasks . image enhancement techniques applied to reduce noise

In the images and sharpen the images for better accuracy

Result and discussion:-

Kasinathan et al. [3], experimented on 5 and 9 insect classes. To detect the highest

prediction accuracy, Rani et al. [4] employed an SVM classifier to classify whiteflies, aphids, and

thrips, and proposed a model for accurately detecting pest-infected regions on leaves with a

remarkable 98% accuracy. By this analysis of machine learning algorithms we get more accuracy

to detect the insects.

Conclusion:-

The main objective of our project is to detect local insects, pest or leaf present on crops using various machine learning algorithms. In this study, we are going to detect ladha, black pest and normal leaf from cotton and chili plant of Andhra Pradesh

Key words:-

**Convolutional neural network**

Machine learning, insect detection ,, svm

Introduction:-

Agriculture, a pivotal economic sector, drives growth and living standards. The agri-food industry is crucial, bolstering exports and responding to domestic needs. In developing nations, export revenues and local demand fuel food processing growth. Storage, equipment upkeep, and workspaces are essential. Pest infestations harm crop quality. . Pests, germs, and weeds cause massive loss to crops and results in a low market for the final products.

Machine vision is now widely used for monitoring crops, soil, grading fruits, detecting plant diseases, and recognizing insect pests. There have been significant advancements in agriculture, including machine learning for identifying and categorizing insects in stored grain conditions.[6] The evaluation of fruit and vegetable quality is accomplished through computer vision-based inspection, which involves four key stages: acquisition, segmentation, feature extraction, and classification [7].

Newer research [8,9] has shown that image processing is a highly effective method for insect detection due to its low computational overhead, rapid detection capabilities, and the ability to distinguish insects with similar colors and shapes. Pest detection in grapevines under various lighting conditions and orientations is achieved through clustering segmentation with descriptors. This method combines contour-based and region-based segmentation to identify individual moths and insects in close proximity [10].

Insect characteristics can vary across geographic regions, impacting detection algorithms. To address this, we conducted a study focused on creating a computer vision algorithm tailored for detecting insects in Andhra Pradesh crops, specifically. Our approach utilizes Artificial Neural Networks and Convolutional Neural Networks, and the implementation takes place on Google Co lab. The project is coded in Python, with support from frameworks like K eras and TensorFlow.

Literature survey:-

Recent years have seen remarkable progress in pest classification and detection through advanced machine learning models [11,12]. These models have effectively classified insect images based on extracted features. Nonetheless, discerning insects with similar traits but in different environmental contexts remains a significant challenge. For example, researchers like Wang [13] and Xie [14] utilized ANN and SVM models for classifying crop insects. Wang's work emphasized the stability of ANN, while SVM excelled in categorizing seven geometrical features.

Yang et al. [15] recently utilized an SVM-based approach to identify insects with different wing proportions, while Xie et al. [14] applied SVM to classify 24 common pest species based on color, texture, and shape, introducing effective feature descriptions. ANN and SVM models are recognized for their strong classification performance. However, existing insect-classification methods require parameter adjustments affecting low-level features [16] and increasing computation time [17]. As a result, the application of varied parameter sets to SVM and ANN algorithms has effectively improved insect classification.

Deep learning models with limited classifiers have faced challenges in image classification, prompting the need for enhancements. Liu et al. [18] systematically assessed methods like VGG, Google Net, Res Net, Over Feat, and their stack sparse autoencoder model to improve deep learning classification. Meanwhile, Nanni et al. [19] investigated handcrafted and learned descriptors, augmenting data to boost CNN performance through a combination of local features, dense sampling, and deep learning techniques applied to augmented images.

This study classifies and detects insects in early-stage corn, soybean, and wheat crops using machine learning and insect pest detection algorithms. Various shape features are utilized with ANN, SVM, KNN, NB, and CNN models. The research evaluates machine learning performance on two datasets, providing insect class information across four datasets. The insect pest detection algorithm efficiently identifies insects in agricultural fields through image processing, employing foreground segmentation and bounding boxes for positioning

The experiments were conducted on a system featuring a 2.3 GHz Intel Core i5 processor and 16 GB of RAM. Our work is implemented using MATLAB 2018a, Python, and utilizes frameworks like Keras and TensorFlow to analyze and classify insect images in the agricultural domain.

The previous studies have shown success in detecting and classifying local insects on crops. However, they may not perform optimally for identifying and categorizing insects on the crops of Andhra Pradesh and Telangana states. Therefore, we have conducted a specific study to classify insects found on crops in these regions.

In our research, we've organized our study into three distinct classes:

1. Ladha: This class represents an insect.

2.Black pest: This class signifies a pest.

3. Leaf: This class indicates a normal leaf without any pest or insect infestation.

To gather data for each class, we collected a set of images from local agricultural fields featuring cotton and chili plants.

**ALGORITHMS:-**

**Artificial neural network (Wikipedia)**

**Artificial neural networks** (**ANNs**), usually simply called **neural networks** (**NNs**), are computing systems inspired by the biological neural networks that constitute animal brains.

An artificial neural network (ANN) consists of interconnected units, or artificial neurons, which mimic the behaviour of biological neurons. These neurons can receive, process, and transmit signals to other connected neurons. Each connection, referred to as an edge, can carry a real number as a signal, and the output of a neuron is determined by a non-linear function applied to the sum of its inputs. Neurons and edges often have associated weights that are adjusted during the learning process, influencing the strength of the signal transmitted through the connection. These weights can be increased or decreased to modify the network's behaviour as it learns.

**Hyperparameter**

A hyperparameter is a constant parameter whose value is set before the learning process begins. The values of parameters are derived via learning. Examples of hyperparameters include learning rate, the number of hidden layers and batch size. The values of some hyperparameters can be dependent on those of other hyperparameters. For example, the size of some layers can depend on the overall number of layers

**Convolutional neural network**

In deep learning, a Convolutional Neural Network (CNN) is a type of Artificial Neural Network (ANN) mainly used for visual data analysis. CNNs, also known as Conv Nets, employ shared-weight convolutional filters to extract features from input data, enabling translation-equivariant responses called feature maps. It's important to note that CNNs are typically equivariant, not fully invariant, to translations. They find extensive applications in image and video analysis, recommendation systems, image classification, segmentation, medical imaging, natural language processing, brain-computer interfaces, and financial time series analysis.

**Materials and Methods:-**

In this study we have three classes. They are

1. Ladha
2. Black pest
3. Leaf

For each and Every class we have collected the images from local agricultural fields of Andhra Pradesh and Telangana. Totally we have collected 1503 images. The below table represent the no of images we have collected for each class.

|  |  |  |  |
| --- | --- | --- | --- |
| **CLASS** | **NO OF IMAGES** | **TRAIN** | **TEST** |
| Ladha | 736 | 686 | 50 |
| Black pest | 512 | 462 | 50 |
| Leaf | 255 | 205 | 50 |
| **TOTAL** | **1503** | **1363** | **150** |

**Table 1.Dataset**

Some of the images are shown below

**Black Pest:**

**Ladha:**

**Leaf:**

**Image preprocessing**

In image preprocessing, enhancement techniques are employed to reduce noise, sharpen images, and enhance their quality, contributing to improved accuracy in insect detection and classification. It's worth noting that the datasets used in this study are already preprocessed and segmented.

**Image Classification Methodology**

Collecting images of insects(dataset)

CNN insect classification

Image pre-processing

Apply image augmentation

Shape feature extraction

**Figer 2. Frame work of insect classification**

Insect classification using ANN

ANN Classifier(fig 2)

A feed-forward multi-layer artificial neural network was designed to classify adult stage whiteflies and thrips in greenhouse environments [20]. Initially, random weights were assigned to the network's linkages, and the activation rates of the output layer were calculated. The model's performance was further enhanced by implementing a backpropagation ANN model to identify Beet armyworm (Spodoptera exigua) among other species [21]. This approach involved iteratively adjusting the network's weights to improve its accuracy in distinguishing the target insect species.

**CNN Classifier(fig 2)**

The CNN model, designed to analyze RGB insect images, is a deep, feed-forward neural network focused on visual feature extraction and computational efficiency [22–25]. It comprises two convolutional layers and two max-pooling layers, followed by a flatten layer, a fully connected layer, and a soft max output layer for insect image classification. Images are rescaled to 64x64 pixels, ensuring speedy processing and reduced computational load and memory usage. The convolution and max-pooling layers use 3x3 and 5x5 filter sizes, respectively. The fully connected layer learns high-level features crucial for the final insect classification.

**Image preprocessing:**

Eliminated unnecessary noise from the dataset

Code:

import cv2

import glob

import os

inputFolder ='leaf'

os.mkdir('leaf Resized')

i = 0

for img in glob.glob(inputFolder+ '/\*.jpg'):

image = cv2.imread(img)

imgResized = cv2.resize(image, (512, 512))

cv2.imwrite("Leaf Resized\image%04i.jpg" %i, imgResized)

i+=1

cv2.destroyAllWindos()

result:

images will be rescaled to 512\*512 pixel images

* + 1. **Artificial neural network**

Code:

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

#import utils

import os

%matplotlib inline

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D

from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D

from tensorflow.keras.models import Model, Sequential

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

#from tensorflow.keras.utils import plot\_model

from IPython.display import SVG, Image

#from livelossplot import PlotLossesTensorFlowKeras()

import tensorflow as tf

base\_dir = '/dataset' #we have data set in dataset folder

train\_dir = os.path.join(base\_dir,'train')

test\_dir = os.path.join(base\_dir,'test')

#data augmentation

img\_size = 64

batch\_size = 64

train\_datagen = ImageDataGenerator(rescale=1/255,

                                  rotation\_range=40,

                                  width\_shift\_range=0.2,

                                  height\_shift\_range=0.2,

                                  shear\_range=0.2,

                                  zoom\_range=0.2,

                                  horizontal\_flip=True,

                                  fill\_mode='nearest')

valid\_datagen = ImageDataGenerator(rescale = 1/255)

train\_generator = train\_datagen.flow\_from\_directory(train\_dir,

                                                    target\_size = (img\_size, img\_size),

                                                    color\_mode='grayscale',

                                                    batch\_size=batch\_size,

                                                    class\_mode='categorical',

                                                    shuffle=True)

validation\_generator = valid\_datagen.flow\_from\_directory(test\_dir,

                                                    target\_size = (img\_size, img\_size),

                                                    color\_mode='grayscale',

                                                    batch\_size=batch\_size,

                                                    class\_mode='categorical',

                                                    shuffle=False)

#ANN Model

model = Sequential()

model.add(Conv2D(64,(3,3),padding='same',input\_shape=(64,64,1)))

model.add(Flatten())

model.add(Dense(2048, input\_dim=8, activation='relu'))

model.add(Dense(1024, input\_dim=8, activation='relu'))

model.add(Dense(1024, input\_dim=8, activation='relu'))

model.add(Dense(128, activation='relu'))

model.add(Dense(4, activation='softmax'))

opt = Adam(lr=0.0005)

model.compile(optimizer=opt,loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Layer (type) Output Shape Param # |
| ================================================================= |
| conv2d\_1 (Conv2D) (None, 64, 64, 64) 640 |
|  |
| flatten\_2 (Flatten) (None, 262144) 0 |
|  |
| dense\_9 (Dense) (None, 2048) 536872960 |
|  |
| dense\_10 (Dense) (None, 1024) 2098176 |
|  |
| dense\_11 (Dense) (None, 1024) 1049600 |
|  |
| dense\_12 (Dense) (None, 128) 131200 |
|  |
| dense\_13 (Dense) (None, 4) 516 |
|  |

Total params: 540,153,092

Trainable params: 540,153,092

Non-trainable params: 0

#model fitting

epochs = 15

steps\_per\_epoch = train\_generator.n//train\_generator.batch\_size

validation\_steps = validation\_generator.n//validation\_generator.batch\_size

checkpoint = ModelCheckpoint('model\_weights.h5',monitor='val\_accuracy',

                            save\_weights\_only=True,mode='max',verbose=1)

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss',factor=0.1,patience=2,min\_lr=0.00001,mode='auto')

callbacks = [checkpoint, reduce\_lr]

history = model.fit(x=train\_generator,

                   steps\_per\_epoch=steps\_per\_epoch,

                   epochs=epochs,

                   validation\_data=validation\_generator,

                   validation\_steps=validation\_steps,

                   callbacks = callbacks )

accuracy: 41.5%

* + 1. **Convolutional neural network (CNN)**

**Code:**

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

#import utils

import os

%matplotlib inline

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D

from tensorflow.keras.layers import BatchNormal ization, Activation, MaxPooling2D

from tensorflow.keras.models import Model, Sequential

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

#from tensorflow.keras.utils import plot\_model

from IPython.display import SVG, Image

#from livelossplot import PlotLossesTensorFlowKeras()

import tensorflow as tf

base\_dir = '/dataset' #insect data set is present in dataset folder

train\_dir = os.path.join(base\_dir,'train')

test\_dir = os.path.join(base\_dir,'test')

#data augmentation

img\_size = 64

batch\_size = 64

datagen\_train = ImageDataGenerator(horizontal\_flip=True)

train\_generator = datagen\_train.flow\_from\_directory(train\_dir,

                                                    target\_size = (img\_size, img\_size),

                                                    color\_mode='grayscale',

                                                    batch\_size=batch\_size,

                                                    class\_mode='categorical',

                                                    shuffle=True)

datagen\_validation = ImageDataGenerator(horizontal\_flip=True)

validation\_generator = datagen\_validation.flow\_from\_directory(test\_dir,

                                                    target\_size = (img\_size, img\_size),

                                                    color\_mode='grayscale',

                                                    batch\_size=batch\_size,

                                                    class\_mode='categorical',

                                                    shuffle=False)

#CNN model

model = Sequential()

#1 - conv

model.add(Conv2D(64,(3,3),padding='same',input\_shape=(64,64,1)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

#2 - conv

model.add(Conv2D(128,(5,5),padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.25))

#flatten

model.add(Flatten())

model.add(Dense(256))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.25))

model.add(Dense(512))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.25))

model.add(Dense(4,activation='softmax'))

opt = Adam(lr=0.0005)

model.compile(optimizer=opt,loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| Layer (type) Output Shape Param # |
|  |
| conv2d\_16 (Conv2D) ( None, 64, 64, 64) 640 |
|  |
| batch\_normalization\_32 (Bat ( None, 64, 64, 64) 256 |
| chNormalization) |
|  |
| activation\_32 (Activation) (None, 64, 64, 64) 0 |
|  |
| max\_pooling2d\_16 (MaxPoolin (None, 32, 32, 64) 0 |
| g2D) |
|  |
| dropout\_32 (Dropout) (None, 32, 32, 64) 0 |
|  |
| conv2d\_17 (Conv2D) (None, 32, 32, 128) 204928 |
|  |
| batch\_normalization\_33 (Bat (None, 32, 32, 128) 512 |
| chNormalization) |
|  |
| activation\_33 (Activation) (None, 32, 32, 128) 0 |
|  |
| max\_pooling2d\_17 (MaxPoolin (None, 16, 16, 128) 0 |
| g2D) |
|  |
| dropout\_33 (Dropout) (None, 16, 16, 128) 0 |
|  |
| flatten\_8 (Flatten) (None, 32768) 0 |
|  |
| dense\_24 (Dense) (None, 256) 8388864 |
|  |
| batch\_normalization\_34 (Bat (None, 256) 1024 |
| chNormalization) |
|  |
| activation\_34 (Activation) (None, 256) 0 |
|  |
| dropout\_34 (Dropout) (None, 256) 0 |
|  |
| dense\_25 (Dense) (None, 512) 131584 |
|  |
| batch\_normalization\_35 (Bat (None, 512) 2048 |
| chNormalization) |
|  |
| activation\_35 (Activation) (None, 512) 0 |
|  |
| dropout\_35 (Dropout) (None, 512) 0 |
|  |
| dense\_26 (Dense) (None, 4) 2052 |
|  |

Total params: 8,731,908

Trainable params: 8,729,988

Non-trainable params: 1,920

**#model fitting**

epochs = 15

steps\_per\_epoch = train\_generator.n//train\_generator.batch\_size

validation\_steps = validation\_generator.n//validation\_generator.batch\_size

checkpoint = ModelCheckpoint('model\_weights.h5',monitor='val\_accuracy',

                            save\_weights\_only=True,mode='max',verbose=1)

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss',factor=0.1,patience=2,min\_lr=0.00001,mode='auto')

callbacks = [checkpoint, reduce\_lr]

history = model.fit(x=train\_generator,

                   steps\_per\_epoch=steps\_per\_epoch,

                   epochs=epochs,

                   validation\_data=validation\_generator,

                   validation\_steps=validation\_steps,

                   callbacks = callbacks )

**accuracy= 92%**

**5.5.3 WEB APP:**

Among ANN and CNN, CNN gives the best accuracy and performance. So the weights obtained from cnn is used in web app to recognize the image.

Web Pages:

home.html

<html>

<head>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<title>Upload Image</title>

</head>

<body>

<div class="col-lg">

<h1><span class="badge badge-danger">Insect Detection</span></h1>

</div>

<div class="col-lg" style="border:thin">

<form action = "/predict" method = "post" enctype="multipart/form-data">

<input type="file" name="file" align="center"/>

<br>

<br>

<br>

<input type = "submit" value="Upload">

</form>

</div>

</body>

</html>

**predict.hmtl**

<!DOCTYPE html>

<html>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

<title>Quality Check</title>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-sm">

<h1>Category <span class="badge badge-secondary">{{product}}</span></h1>

</div></div></div>

<br>

</head>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

</body>

</html>

**“\_\_init\_\_.py” for running web app**

import tensorflow as tf

from tensorflow import keras

from flask import Flask, redirect, url\_for, render\_template, request, flash

from flask import Flask

from tensorflow.keras.preprocessing import image

import numpy as np

import cv2

import os

print(tf.\_\_version\_\_)

model = keras.models.load\_model('model\_weights.h5')

def prepare(filepath):

IMG\_SIZE = 64

img\_array = cv2.imread(filepath, cv2.IMREAD\_GRAYSCALE)

new\_array = cv2.resize(img\_array, (IMG\_SIZE, IMG\_SIZE))

return new\_array.reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

def prediction(img\_path):

img = image.load\_img(img\_path, target\_size=(64, 64,1))

img\_array = image.img\_to\_array(img)

prediction = model.predict([prepare(img\_path)])

return prediction

app = Flask(\_\_name\_\_)

#get\_model()

@app.route("/", methods=['GET', 'POST'])

def home():

return render\_template('home.html')

@app.route("/predict", methods = ['GET','POST'])

def predict():

if request.method == 'POST':

file = request.files['file']

filename = file.filename

file\_path = os.path.join(r'E:\Data set\static', filename) #slashes should be handeled properly

file.save(file\_path)

print(filename)

print(file\_path)

product = prediction(file\_path)

classes = {1:'ladha',2:'leaf',3:'black pest'}

classes\_x=np.argmax(product,axis=1)

product = classes[int(classes\_x)]

return render\_template('predict.html', product = product)

app.run()

**5.6 Results:**

After running the ANN model we have got 41.5% accuracy and for CNN model we have got 92%accuracy

**5.6.1 ANN model:**

Epoch 1/15

20/20 [==============================] - ETA: 0s - loss: 7.1101 - accuracy: 0.4109

Epoch 1: saving model to model\_weights.h5

20/20 [==============================] - 147s 7s/step - loss: 7.1101 - accuracy: 0.4109 - val\_loss: 2.2769 - val\_accuracy: 0.3906 - lr: 5.0000e-04

Epoch 2/15

20/20 [==============================] - ETA: 0s - loss: 1.2544 - accuracy: 0.4683

Epoch 2: saving model to model\_weights.h5

20/20 [==============================] - 149s 7s/step - loss: 1.2544 - accuracy: 0.4683 - val\_loss: 1.3257 - val\_accuracy: 0.3906 - lr: 5.0000e-04

Epoch 3/15

20/20 [==============================] - ETA: 0s - loss: 1.0027 - accuracy: 0.4683

Epoch 3: saving model to model\_weights.h5

20/20 [==============================] - 143s 7s/step - loss: 1.0027 - accuracy: 0.4683 – val loss: 1.2200 - val\_accuracy: 0.3906 - lr: 5.0000e-04

Epoch 4/15

20/20 [==============================] - ETA: 0s - loss: 0.9900 - accuracy: 0.5035

Epoch 4: saving model to model\_weights.h5

20/20 [==============================] - 150s 7s/step - loss: 0.9900 - accuracy: 0.5035 - val\_loss: 1.1859 - val\_accuracy: 0.3906 - lr: 5.0000e-04

Epoch 5/15

20/20 [==============================] - ETA: 0s - loss: 1.0049 - accuracy: 0.4746

Epoch 5: saving model to model\_weights.h5

20/20 [==============================] - 149s 7s/step - loss: 1.0049 - accuracy: 0.4746 - val\_loss: 1.3327 - val\_accuracy: 0.3906 - lr: 5.0000e-04

Epoch 6/15

20/20 [==============================] - ETA: 0s - loss: 0.9781 - accuracy: 0.5113

Epoch 6: saving model to model\_weights.h5

20/20 [==============================] - 145s 7s/step - loss: 0.9781 - accuracy: 0.5113 - val\_loss: 1.1132 - val\_accuracy: 0.3594 - lr: 5.0000e-04

Epoch 7/15

20/20 [==============================] - ETA: 0s - loss: 0.9629 - accuracy: 0.5496

Epoch 7: saving model to model\_weights.h5

20/20 [==============================] - 145s 7s/step - loss: 0.9629 - accuracy: 0.5496 - val\_loss: 1.0250 - val\_accuracy: 0.3906 - lr: 5.0000e-04

Epoch 8/15

20/20 [==============================] - ETA: 0s - loss: 0.9717 - accuracy: 0.5293

Epoch 8: saving model to model\_weights.h5

20/20 [==============================] - 156s 8s/step - loss: 0.9717 - accuracy: 0.5293 - val\_loss: 1.0586 - val\_accuracy: 0.3750 - lr: 5.0000e-04

Epoch 9/15

20/20 [==============================] - ETA: 0s - loss: 0.9557 - accuracy: 0.5504

Epoch 9: saving model to model\_weights.h5

20/20 [==============================] - 147s 7s/step - loss: 0.9557 - accuracy: 0.5504 - val\_loss: 1.3406 - val\_accuracy: 0.3984 - lr: 5.0000e-04

Epoch 10/15

20/20 [==============================] - ETA: 0s - loss: 0.9511 - accuracy: 0.5653

Epoch 10: saving model to model\_weights.h5

20/20 [==============================] - 153s 8s/step - loss: 0.9511 - accuracy: 0.5653 - val\_loss: 1.0832 - val\_accuracy: 0.4219 - lr: 5.0000e-05

Epoch 11/15

20/20 [==============================] - ETA: 0s - loss: 0.9322 - accuracy: 0.5668

Epoch 11: saving model to model\_weights.h5

20/20 [==============================] - 146s 7s/step - loss: 0.9322 - accuracy: 0.5668 - val\_loss: 1.0886 - val\_accuracy: 0.4141 - lr: 5.0000e-05

Epoch 12/15

20/20 [==============================] - ETA: 0s - loss: 0.9186 - accuracy: 0.5809

Epoch 12: saving model to model\_weights.h5

20/20 [==============================] - 158s 8s/step - loss: 0.9186 - accuracy: 0.5809 - val\_loss: 1.0874 - val\_accuracy: 0.4141 - lr: 1.0000e-05

Epoch 13/15

20/20 [==============================] - ETA: 0s - loss: 0.9206 - accuracy: 0.5731

Epoch 13: saving model to model\_weights.h5

20/20 [==============================] - 146s 7s/step - loss: 0.9206 - accuracy: 0.5731 - val\_loss: 1.0846 - val\_accuracy: 0.4141 - lr: 1.0000e-05

Epoch 14/15

20/20 [==============================] - ETA: 0s - loss: 0.9208 - accuracy: 0.5754

Epoch 14: saving model to model\_weights.h5

20/20 [==============================] - 149s 7s/step - loss: 0.9208 - accuracy: 0.5754 - val\_loss: 1.0865 - val\_accuracy: 0.4141 - lr: 1.0000e-05

Epoch 15/15

20/20 [==============================] - ETA: 0s - loss: 0.9175 - accuracy: 0.5801

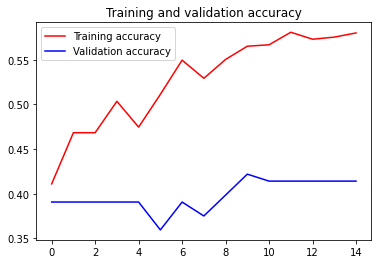
Epoch 15: saving model to model\_weights.h5

20/20 [==============================] - 155s 8s/step - loss: 0.9175 - accuracy: 0.5801 - val\_loss: 1.0817 - val\_accuracy: 0.4141 - lr: 1.0000e-05

**RESULT**

**Graph**

Below graph represents the change in accuracy for each and every epoch



epochs

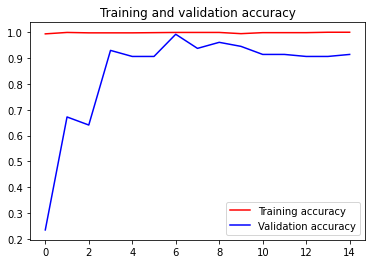
A

c

curacy

Graph 2.1 change in accuracy of ANN model

Below graph represents the change in accuracy for each and every epoch



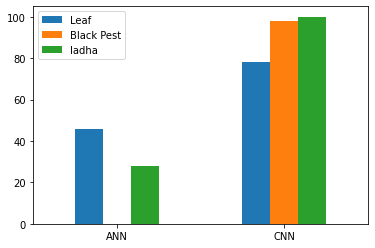
Epochs

accuracy

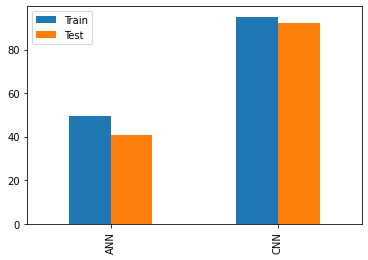
Graph: change in accuracy for every epoch (CNN)model

**ANN VS CNN**

**Classwise accuracy comparision for both ANN and CNN model**



**Graph 2.3classwise accuracy (CNN & ANN model)**

****

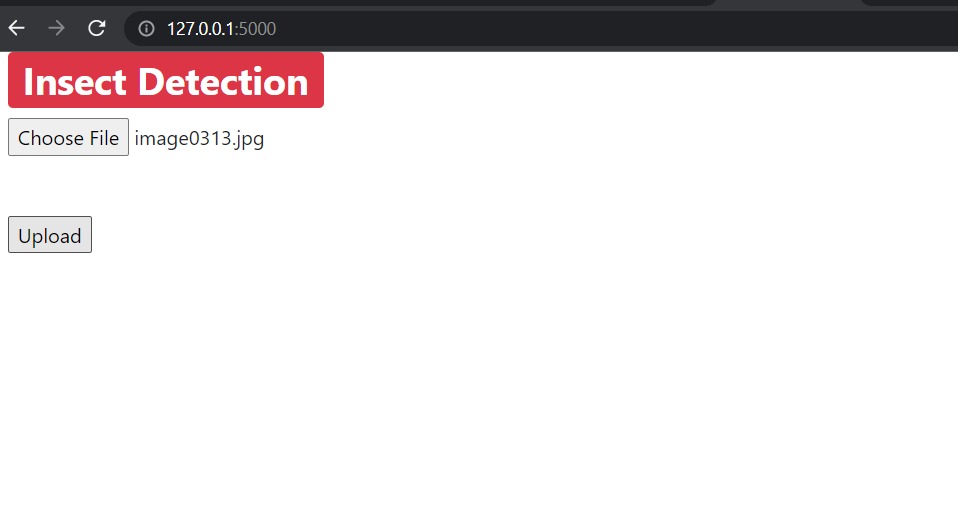
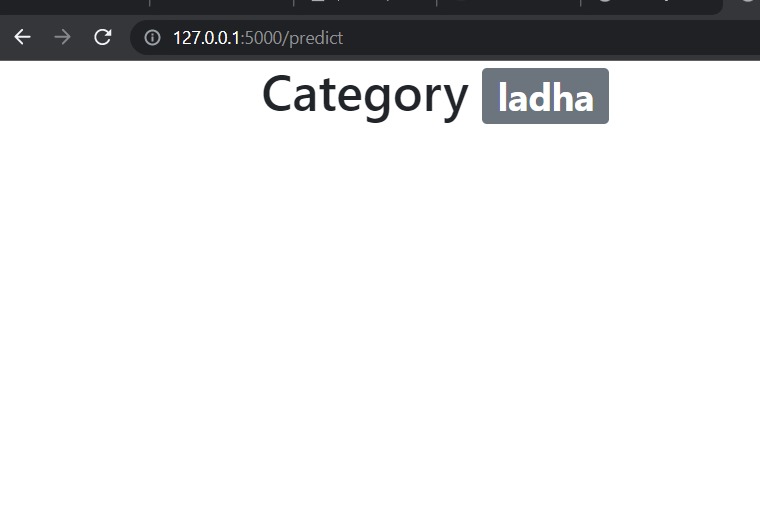
Graph 2.4 accuracy of CNN and ANN model

After comparing the results of CNN and ANN, we are concluded that CNN algorithm is working better for the insect detection, especially for our project. So we used the weights obtained from CNN model to build the web and detect the image.

**Web APP**

In this project we have also build a web app which accepts an image from the user and detect if any insect or pest present in the leaf.

This web app is developed by using Flask web development framework.

Figure 5.3

**Results:**

After running the ANN model we have got 41.5% accuracy and for CNN model we have got 92%accuracy

**Conclusion**

Our project aims to identify Ladha, Black Pest, and Leaf in input images through a web app. Farmers can benefit from this tool for crop protection. We've gathered 1503 images from cotton and chili fields in Andhra Pradesh.

**Future work:**

Our future plans include expanding the project's scope by incorporating additional datasets encompassing various insects and pests from different crops. We'll collect more data and introduce additional classes for detection. Moreover, we aim to implement a mobile camera system for continuous image capture in the field to provide real-time alerts to farmers whenever it detects insects or pests.

**Acknowledgement:**

The authors thanks to the Saveetha school of engineering and SIMTS university for their support in providing the infrastructure needed to complete this work successfully

**References**

[1]Senthil Kumar Swami Durai, Mary Divya Shamili. (2022). Smart farming using Machine Learning and Deep Learning techniques, Decision Analytics Journal, 3, 100041.

Insect classification and detection in field crops using modern machine learning techniques

[2]image enhancement techniques applied to reduce noise in the images and sharpen the images for better accuracy

[3]Thenmozhi Kasinathan, T. Kasinathan, Dakshayani Singaraju, D. Singaraju, & Srinivasulu Reddy Uyyala, S. Reddy Uyyala. (2021). Insect classification and detection in field crops using modern machine learning techniques. Information processing in agriculture, vol-8, pp.446-457.

[4]R. Uma Rani, P.Amsini.(2016). Pest Identification In Leaf Images Using SVM Classifier. International Journal Of Computational Intelligence And Informatics, Vol. 6: No. 1.

[5]Insect classification and detection in field crops using modern machine learning techniques

[6]Shen Y, Zhou H, Li J, Jian F, Jayas DS. Detection of stored-grain insects using deep learning. Comput Electron Agric 2018;145:319–25.

[7]Bhargava A, Bansal A. Fruits and vegetables quality evaluation using computer vision: A review. J King Saud Univ Comput Inf Sci 2018. https://doi.org/10.1016/j. jksuci.2018.06.002 [in press].

[8] Nanni L, Maguolo G, Pancino F. Research on insect pest image detection and recognition based on bio inspired methods; 2019. arXiv: 1910.00296.

[9] Deng L, Wang Z, Wang C, He Y, Huang T, Dong Y, et al. Application of agricultural insect pest detection and control map based on image processing analysis. J Intell Fuzzy Syst 2020;38:379–89.

[10] Bakkay MC, Chambon S, Rashwan HA, Lubat C, Barsotti S. Automatic detection of individual and touching moths from trap images by combining contour-based and region-based segmentation. IET Comput Vis 2017;12:138–45.

[11] Yaakob SN, Jain L. An insect classification analysis based on shape features using quality threshold ARTMAP and moment invariant. Appl Intell 2012;37(1):12–30.

[12]Xie C, Wang R, Zhang J, Chen P, Dong W, Li R, Chen T, Chen H. Multi-level learning features for automatic classification of field crop pests. Comput Electron Agric 2018;152:233–41.

[13] Wang J, Lin C, Ji L, Liang A. A new automatic identification system of insect images at the order level. Knowl-Based Syst 2012;33:102–10.

[14] Xie C, Zhang J, Li R, Li J, Hong P, Xia J, Chen P. Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning. Comput Electron Agric 2015;119:123–32.

[15] Yang H-P, Ma C-S, Wen H, Zhan Q-B, Wang X-L. A tool for developing an automatic insect identification system based on wing outlines. Sci Rep 2015;5(1):1–11.

[16] Coates A, Ng AY. Selecting receptive fields in deep networks. Adv Neural Inf Process Syst 2011:2528–36.

[17] Lucas B, Shifaz A, Pelletier C, O’Neill L, Zaidi N, Goethals B, Petitjean F, Webb GI. Proximity Forest: an effective and scalable distance-based classifier for time series. Data Min Knowl Disc 2019;33(3):607–35.

[18] Liu J-e, An F-P. Image classification algorithm based on deep learning-kernel function. Sci Program 2020;2020.

[19] Nanni L, Brahnam S, Ghidoni S, Maguolo G. General purpose (GenP) bioimage ensemble of handcrafted and learned features with data augmentation; 2019. arXiv: 1904.08084.

[20] Espinoza K, Valera DL, Torres JA, Lo´pez A, Molina-Aiz FD. Combination of image processing and artificial neural networks as a novel approach for the identification of Bemisia tabaci and Frankliniella occidentalis on sticky traps in greenhouse agriculture. Comput Electron Agric 2016;127:495–505.

[21] Asefpour Vakilian K, Massah J. Performance evaluation of a machine vision system for insect pests identification of field crops using artificial neural networks. Arch Phytopathol Pflanzenschutz 2013;46(11):1262–9.

[22] Martineau M, Conte D, Raveaux R, Arnault I, Munier D, Venturini G. A survey on image-based insect classification. Pattern Recogn 2017;65:273–84.

[23] Zhu LQ, Ma MY, Zhang Z, Zhang PY, Wu W, Wang DD, et al. Hybrid deep learning for automated lepidopteran insect image classification. Orient Insects 2017;51:79–91.

[24] Martins VA, Freitas LC, de Aguiar MS, de Brisolara LB, Ferreira PR. Deep learning applied to the identification of fruit fly in intelligent traps. In: Proc SBESC 19 IX Brazilian symposium on computing systems engineering. Rio Grande do Norte, Brazil. IEEE; 2019. p. 1–8.

[25] Sharma P, Berwal YP, Ghai W. Performance analysis of deep learning CNN models for disease detection in plants using image segmentation. Inf Process Agric 2019. https://doi.org/ 10.1016/j.inpa.2019.11.001 [in press].