



***DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SCHOOL OF ENGINEERING AND TECHNOLOGY,
SHARDA UNIVERSITY, GREATER NOIDA***

***A project Object Detection Software Submitted
In partial fulfillment of the requirement of the degree of
Bachelor of Technology in Computer Science and Engineering***

By

SHIV SARAD CHAUDHARY (180101295)

CHAITANYA KUMAR(180101091)

PARSHANT KUMAR(180101223)

Supervised by:

Dr.Gouri Shankar Mishra,Assoc.Prof

May,2022

DECLARATION

I hereby declare that the project entitled “_____Object Detection Software_____” submitted for the course of Major Project-I (Project) is my original work. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

**Signature of the
Student(s)**

Place: Sharda University Greater

Noida Date:

CERTIFICATE

It is certified that the work contained in the project report titled “Title of the Project Report,” by “Name of the Student(s),” has been carried out under my/our supervision and that this work has not been submitted elsewhere.

**Signature of the
Guide**

Place: Sharda University Greater

Noida Date:

Signature of the department

Name: Prof.(Dr.)Nitin Rakesh

Date:

Signature of External

Examiner

Date:

ACKNOWLEDGEMENT

A major Project Report is a great opportunity for learning and enhancement of Skills .I sincerely appreciate to have so many helpful people who lead us towards the completion of the Project. It can't be completed without guidance and suggestion of them. I heartily want to give big thank to our mentor,HOD and Guider who gives their valuable time to extract the capabilities which is hidden inside us.

My Greatful thanks to Dr.Gouri Shankar Mishra for his guidance in our project work.Dr.Pankaj Sharma and Dr.Ali Imam Abidi ,who in spite of being extraordinarily busy with academics,took time out to hear,guide and keep us on the correct path.

CSE department monitored our progress and arranged all facilities to make work easier.We choose this moment to acknowledge their contribution gratefully.

Name and signature of the Students

Shiv Sarad Chaudhary(180101295)

Chaitanya Kumar(180101091)

Prashant Kumar(180101223)

Abstract

Deep learning has evolved into a powerful machine learning technology that incorporates multiple layers of features or representations of data to get cutting-edge results. Deep learning has demonstrated outstanding performance in a variety of fields, including picture classification, segmentation, and object detection. Deep learning approaches have recently made significant progress in fine-grained picture categorization, which tries to discriminate subordinate-level categories. Due to strong intra-class and low inter-class variance, this task is highly difficult. Object detection and the identification of pedestrians is critical in autonomous driving applications. In real-time applications, approaches based on Convolutional Neural Networks have shown significant increases in accuracy and decision speed. In this research, authors present various state of the art deep learning algorithms i.e., VGG-16, VGG19, DenseNet-121, InceptionV3 and customized 3 layers CNN model for object detection. Model adopted is trained and validated on self-made five class of furniture dataset. After extensive experiments, highest accuracy obtained was 99.89% with VGG-19.

Keywords — Object Detection, Object classification, Convolutional Neural Network, VGG, Confusion Matrix, Accuracy.

Table of Contents

Title Page	i
Declaration of the Student	ii
Certificate of the Guide	iii
Abstract	iv
Acknowledgement	v
List of Figures	vi
List of Tables (optional)	vii
Timeline / Gantt Chart	viii
1. INTRODUCTION*	1
1.1 Problem Definition	1
1.2 Project Overview/Specifications* (page-1 and 3)	2
1.3 Hardware Specification	3
1.4 Software	4
Specification 1.3.1	4
1.3.2	
...	
2. LITERATURE SURVEY	5
2.1 Existing System	5
2.2 Proposed System	6
2.3 Feasibility Study* (page-4)	7
3. SYSTEM ANALYSIS & DESIGN	
3.1 Requirement Specification* (page-2)	9
3.2 Flowcharts / DFDs / ERDs	10
3.3 Design and Test Steps / Criteria	12
3.3 Algorithms and Pseudo	16
Code 3.3.1	18
3.3.2	19
3.4 Testing Process	22
...	27
4. RESULTS / OUTPUTS	40
5. CONCLUSIONS / RECOMMENDATIONS	47
6. REFERENCES	49
7. APPENDICES	50

Introduction

1. Problem Identification

Object location PC code visual insight PC code licenses PCs to detect and strategy pictures during a method practically like human vision. a few ASCII text document comes and pre-prepared models square measure currently available to help sight nonexclusive articles: somebody, a seat, food or a tree, for instance. In this in vogue time, using time effectively is that the significant feature of Life. subsequently we tend to ought to must be constrained to oversee it for our advancement and upliftment. inside the Waste assembling plant various sort of item is gathered and keep that is unquestionably irksome to search out with conventional human vision. subsequently it came to me the method for doing all together that we will locate it momentarily time. At last we tend to deliver partner degree object ID PC code which can decide the kind of item by sleuthing and perceiving through an image that is caught from the gathered waste which might be reuse also.

2. Software Specification

We made the product utilizing python, profound learning, AI.

Utilization of python open source: Now, this can be cutting edge innovation abuse pc vision is object recognition, and picture order in python. It manages trademark the thing gift in pictures or recordings by outline. numerous use of item recognition like face discovery, vehicle location, self-driving vehicles, and far a great deal of application. So for this situation python is best hotspot for this product.

AI : Object recognition could be a pc vision strategy for finding cases of articles in pictures or recordings. Object recognition calculations by and large influence AI or profound figuring out how to give substantial outcomes. The objective of item discovery is to duplicate this knowledge utilizing a pc.

Profound Learning: Object recognition might be finished by an AI approach and a profound learning approach. The profound learning approach is significantly upheld Convolutional Neural Networks (CNNs).

2(b). Writing Survey

Object recognition is that the distinguishing proof of partner degree object inside the picture close by its localisation and arrangement. it's wide unfurl applications and could be a significant component for vision fundamentally based bundle frameworks. This paper looks to play out a thorough review of late item identification calculations that utilization profound learning. As a piece of the overview, the points investigated typify various calculations, quality measurements, speed/size compromises and instructing techniques. This paper centers around 2|the 2} assortments of article location

calculations the SSD classification of single step identifiers and furthermore the faster .R-CNN classification of two stage locators.

Procedures to build finders that are moveable and fast on low battery-controlled gadgets are tended to by investigating new light-weight convolutional base structures. Eventually, a thorough survey of the qualities and shortcomings of each finder drives America to this best in class.

2.1. Proposed System

Object proposition plans to get a specific amount of competitor bounding boxes to see the likely items and their areas in an image, that is wide applied to a few visual assignments for pre-handling. Object detection system can be considered as much difficult to classify than image Classification during processing because of the dual priorities, limited data. Actually all Object detection frameworks continue to struggle with the small and same type of Objects. In the waste factory various type of object is stored which is very hard to find out with our normal human vision. so it will be better if we classify it in very short time. Finally we create an object detection system that will identify the type of objects given by identifying and recognising through the input which is captured from the stored object. The object using shape and color involved in the project is also significantly used. object detection tells about what is the object. Also what is the object via bounding box[x,y]- coordinates. So, object detection algorithm allows us to input one image and obtain multiple bounding boxes and class labels as output. This object detection follows the given pattern:

1. Input: an object which we want to detect.
2. Output: given three values indicating
 - .a list of bounding boxes, or the[x,y]-coordinates for each object in image.
 - .the class indicating all boxes
 - .the probability of score associated with each bounding box and label of the class.

2.2. Feasibility Study

This paper depicts a procedure for the identification of textureless articles. Our objective items typify household item and private apparatuses, that haven't any affluent textural choices or trademark shapes. gaining practical experience in the accommodation of utilization, we tend to diagram a model that addresses objects as far as three-dimensional edgels and surfaces. Object discovery is performed by superimposing input document on the model. A two-stage algorithmic principle is applied to bring out object presents. Surfaces region unit wont to remove applicants from the input record, and edgels region unit then wont to decide the reason for an objective article abuse two-dimensional model coordinating. Tests double-dealing four genuine household item and private machines were performed to call attention to the practicality of the arranged technique. We propose the possible significance in impediment and litter Condition for the better study of the material which we used for the detection .

In PC vision field, object location and following assumes a significant part. Object recognition implies that finding/recognizing objects in casing of video succession and while following is that the technique for finding moving article or numerous items throughout a measure of time exploitation camera. Actually, following is assessing mechanical peculiarity or way of partner degree object in the picture. the stock of high power PCs, superior grade and low worth camera can lead the decent arrangement of interest in object following calculations. 3 principle key stages for video examination square measure : Detection of intriguing moving Objects, following of such items from edge to line, Analysis of Object tracks to recognize their conduct. The fundamental application spaces of article identification and following are: Motion essentially based acknowledgment, machine-controlled police work, video compartmentalisation, human-PC communication, traffic recognition, vehicle route and so forth anyway object location and following is diverse strategy once protrusive 3D world on second picture owing to the deficiency of information, commotion gift in an image, convoluted item movement, verbalized nature of article, confounded article shapes and impediment. the greater part following calculations accept that the article movement is wash with no unexpected changes. whatever amount of it's impractical. regardless of whether such a great deal of troubles square measure exist in object identification and following, assortment of article discovery and following calculations square measure anticipated and upheld. each and every calculations square measure shifted with pertinence the article delineation utilized, picture choices and in this manner the movement, look and type of the article be displayed for identification and following. the structure representation should be joined with look delineation. There square measure a various manners by which to address the looks component of articles. some of the normal look outline inside the instance of item following is depict

An element of AN interest is AN item. Item will be drawn by their shapes and choices. There region unit fluctuated object portrayals open for pursuit. the decision of item outline for pursuit is principally founded on application. A portion of the item

delineation methods are: Points representation that is fitting for pursuit AN article that possesses small locale in an image, Primitive mathematical structure outline is suitable for pursuit each inflexible and non unbending items, Object outline representation is proper for pursuit confounded non unbending shapes, explained structure model is fitting for pursuit verbalized article with body part, legs, hands and so on, skeletal model is suitable for pursuit each expressed articles for the enhancement of the material for detection with accuracy and mobility of performance.

Hard Ware Analysis

Execution The model beginnings with downloading the dataset from the drive followed by corporate greed related libraries that exemplify TensorFlow Keras, NumPy, matplotlib.pyplot and a couple of something else. Assemble the Model The photos that might get into convent square measure 150x150 shading pictures on information Preprocessing, we'll add dealing with to estimate every one of the photos to 150x150 prior to taking care of them into the neural organization). We will stack three modules. Our convolutions care for 3x3 windows and our maxpooling layers care for 2x2 windows. Our first convolution separates sixteen channels, the ensuing one concentrates 32 channels, and furthermore the last one concentrates 64 channels. Prior to the result layer, 2 completely associated layers square measure further. The result layer is abuse softmax enactment to characterize pictureStandard HyperParameters

Conv Layers 3 Layers Filter Size 3

Ages 5

L2 Regularization 0 (REMOVE) Dropout 0 (REMOVE)

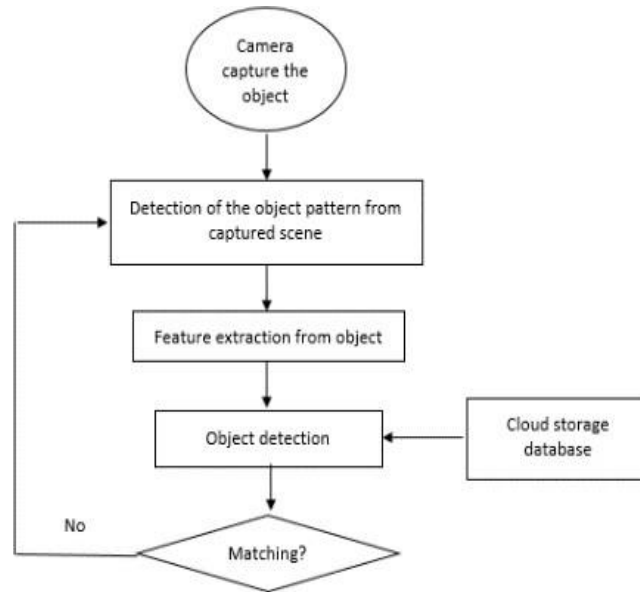
Cluster size 16 Optimizer adam

Misfortune Categorical crossentropy

Name Furniture-pictures Training set 4024 pictures Testing set 423 pictures

Picture size Varying from (96 x 96) to (450 x 450)

No. of Classes 5 classes - seat, couch, table, turn seat, bed



Testing Steps

1. Dataset Loading
2. Labeling
3. Organizing
4. Process
5. Train

6. Deploy

7. Display

Roboflow has associated account determined out about for every consumer. presently we can communicate approximately the whole lot about strides inside the errand of article recognition.

1. . Dataset Loading

Roboflow has each open and man or woman datasets. Public datasets might be gotten to from the real site and person datasets is probably transferred by means of clients. we will pick out any dataset we will pretty regularly want from these or circulate our own custom dataset from the 'your dataset' segment. Assuming you might want to make use of public datasets test that to fork it as soon as hole. when forking a dataset on the off hazard that we will extra frequently than not would like we will additionally upload our personal pictures.

2. Organise

institution cooperation is well ability exploitation roboflow group sharing. for each rendition of modifications made to facts could be pondered in to all and sundry from the group exploitation it. there may be additionally associated possibilities for public sharing which can be meditated in the neighborhood vicinity.

three. procedure

specific advances are involved in method like an examination of the knowledgeset must be possible to determine if the dataset is uneven for arrangement models and diverse such experiences regarding facts. records adjustments might be depleted the resulting methods:

1. Installation

Run model on cloud or device. Use fashions on cell or servers or each. Weights which have supplied the only consequences are automatically stored.

2. Display

I've skilled a custom dataset of Indian cash mistreatment python to are expecting the class of furniture as soon as following all of the on pinnacle of steps and coaching model on. beside type it moreover tracks the gadgets.

Our version seems to own foretold well. be aware that this photograph is from the take a look at set and now not the toy.

- Use of machine studying.

we will layout AI is teaching of package to perform certain motion , as in the course of this example acknowledgment items in openness . you should stock bundle set of pix so list encompass which may also later be applied in acknowledgment of article .

large benefit of AI is it essentially doesn't require substantial datasets or processing electricity , and consequently cost of bundle advancement is normally low .

obstacle of this innovation is it want a ton of endeavors of your group to set up absolutely . within the event that you have embedded confined snap shots to consolidate in your mercantilism sets , then, at that point, your package won't organized to decide with precision because it has no longer admittance to extra substantial association of photos .

object acknowledgment is becoming acclimated to seek out extra express classes of items , like canine or feline . At instances , seeing will also be carried out at the same time as now not double-dealing AI innovation . In those instances templet coordinating or photograph division is becoming acclimated to perceive objects .Even greater improvement seeing might be finished by following one in the whole thing about 2 methodologies : AI and profound getting to know , every one of the innovation have their own specialists and cons .

- Utilization of Deep studying

item acknowledgment is turning into familiar with hunt down more specific lessons of articles , like canine or feline . every now and then , seeing can even be executed even as not abuse AI innovation . In those instances templet coordinating or photograph division is becoming familiar with understand objects .Even greater improvement seeing might be completed by following one in everything about 2 methodologies : AI and profound learning , each one of the innovation have their very own professionals and cons .

Profound mastering has potential to modify Manny very surprising capacities or motion , anyway at an same time it desires monster statistics sets , widespread device yet as greater calculation strength . during this technique code has capability to mentor itself

abuse faux neural organization .

on this images square degree just to be named as " Cat " or " No Cat " , then, at that point, there'll be no extra endeavors to be spot to mentor code to split tom cat .

The disadvantage of this methodology is it wishes colossal datasets that may even include exceptional snap shots which could not be essentially to be had for extra modest companies to get to the ones records.

System Analysis and Design

1. Types of data sets used

- **Table**

Type 1



- Type 2
2 Leg Chair



- Type 3
Crossed Leg Chair



- **Chair**
- **There are various types of chair introduced in the datasets.**
 - Type 1
 - 4 Leg Chair



- **Type 2**
 - Crossed leg chair



- **Type 3**
Double leg Joint Steel Rod chair



- **Type 3**
Multi Leg chair



- **Type 4**

3 Leg Chair



- **Swivel chair**

Type 1

4 leg rolling chair



- **Type 2**
3 leg rolling chair



Type 3
Sofa Chair



- **Sofa**

There are various types of sofa datasets are introduced in the project.

- **Type 1**

4 seater sofa



- **Type 2**

3 seater sofa



- **Type 3**
2 seater sofa



- **Type 4**
Single seater sofa



- **Bed**

We have introduced more than 1 type of bed in the project.

- **Type 1**

Double bed



- **Type 2**

Single bed



- **Type 3**
Bed with wooden legs



Libraries used and introduced in the project.

```
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.preprocessing import image
from tensorflow.keras import regularizers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow import keras
from tensorflow.keras import models
from tensorflow.keras.applications.inception_v3 import preprocess_input
```

```
import os
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
import numpy as np
import pandas as pd
import cv2
import os
from PIL import Image
import PIL
from collections import Counter
from skimage.color import rgb2lab, deltaE_cie76
from skimage import io
from skimage import transform
```

```
%matplotlib inline
%matplotlib inline
```


Model Functions Trained into the project

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
conv2d (Conv2D)	(None, 148, 148, 16)	448
max_pooling2d (MaxPooling2D)	(None, 74, 74, 16)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
conv2d_2 (Conv2D)	(None, 34, 34, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 64)	0
flatten (Flatten)	(None, 18496)	0
dense (Dense)	(None, 512)	9470464
dense_1 (Dense)	(None, 5)	2565

Total params: 9,496,613

Trainable params: 9,496,613

Non-trainable params: 0

First Model Epochs To Train The Images

Epoch 1/5

251/251 [=====] - 95s 376ms/step - loss: 0.9
873 - accuracy: 0.6257 - val_loss: 0.6051 - val_accuracy: 0.7788

Epoch 2/5

251/251 [=====] - 94s 373ms/step - loss: 0.5
603 - accuracy: 0.8026 - val_loss: 0.5396 - val_accuracy: 0.8197

Epoch 3/5

251/251 [=====] - 97s 386ms/step - loss: 0.4
420 - accuracy: 0.8513 - val_loss: 0.4037 - val_accuracy: 0.8630

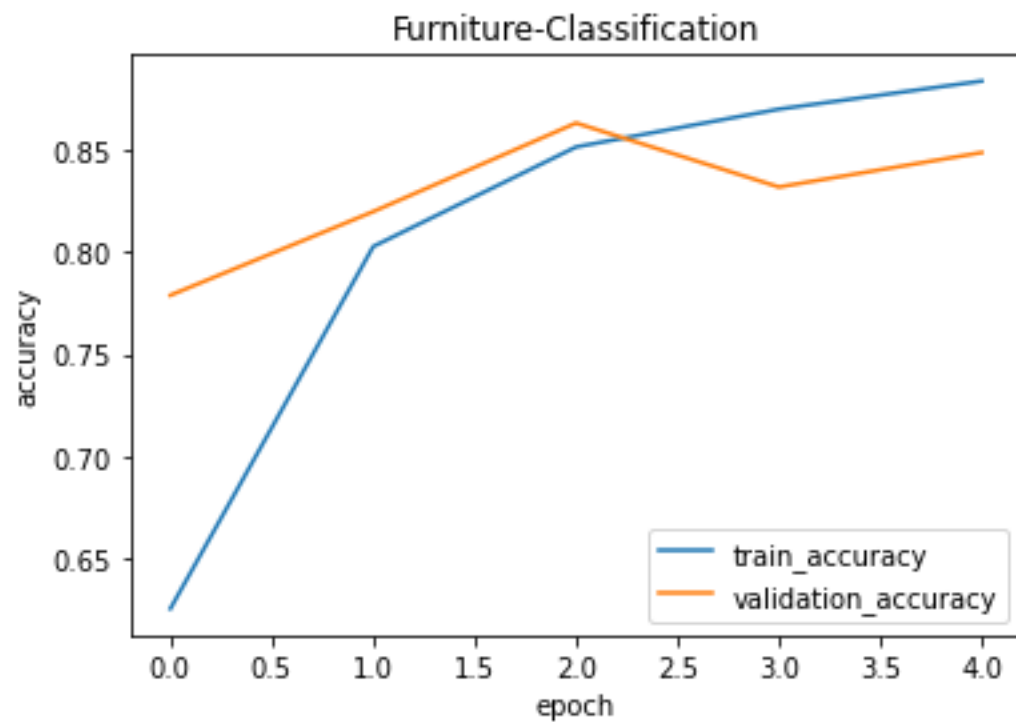
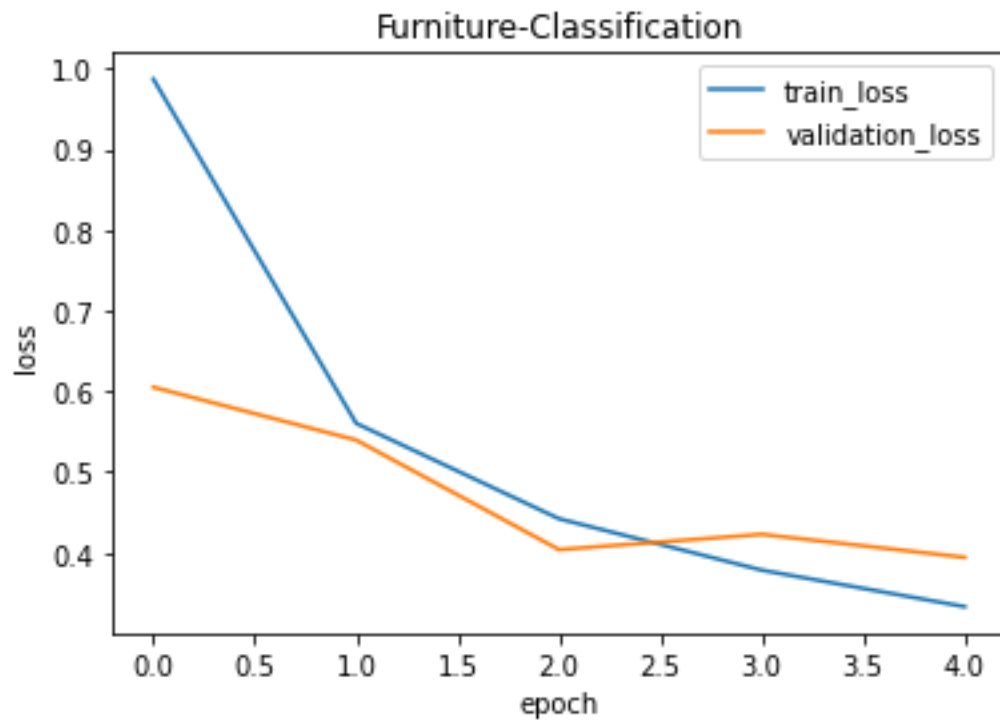
Epoch 4/5

251/251 [=====] - 101s 403ms/step - loss: 0.
3782 - accuracy: 0.8698 - val_loss: 0.4226 - val_accuracy: 0.8317

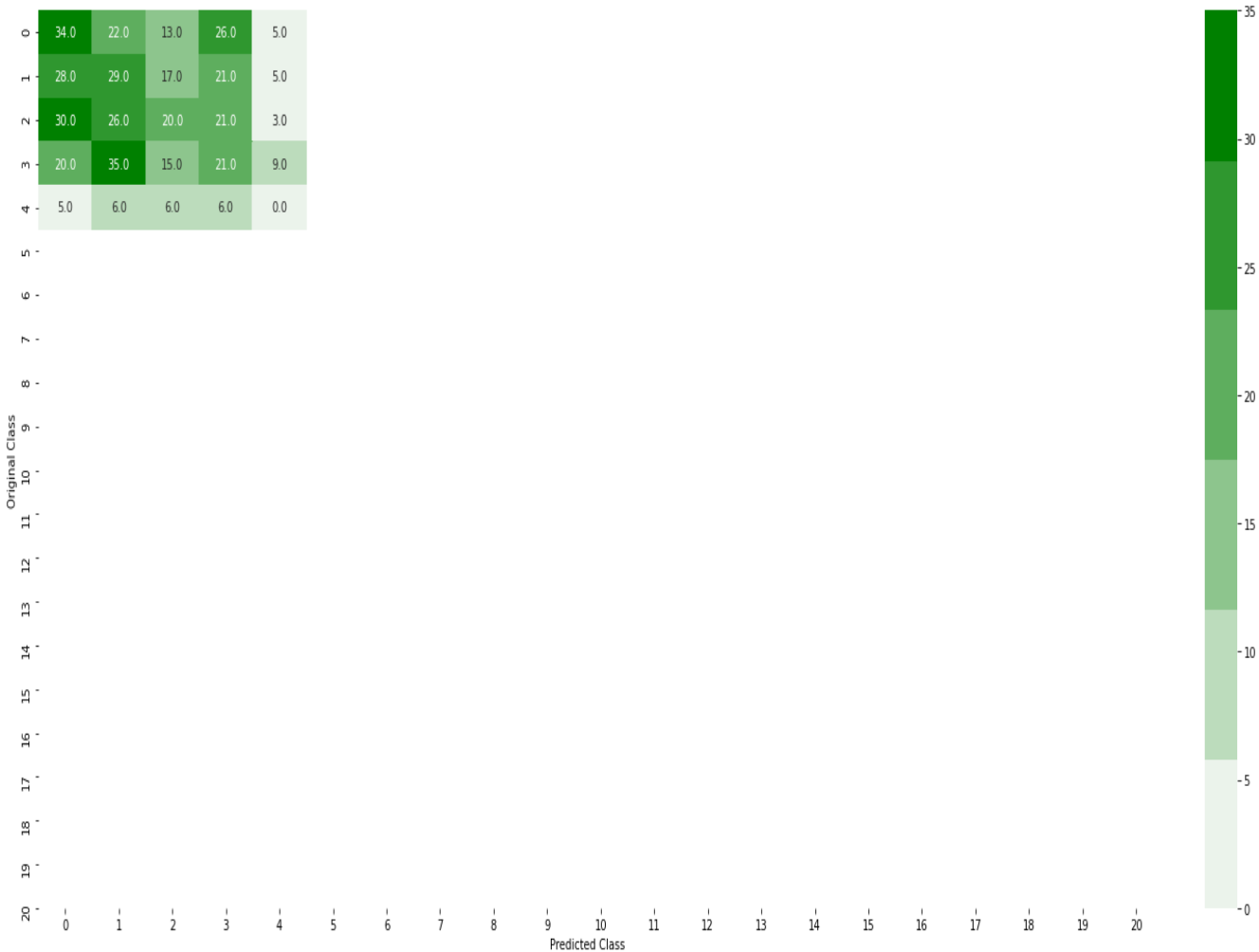
Epoch 5/5

251/251 [=====] - 102s 405ms/step - loss: 0.
3330 - accuracy: 0.8835 - val_loss: 0.3941 - val_accuracy: 0.8486

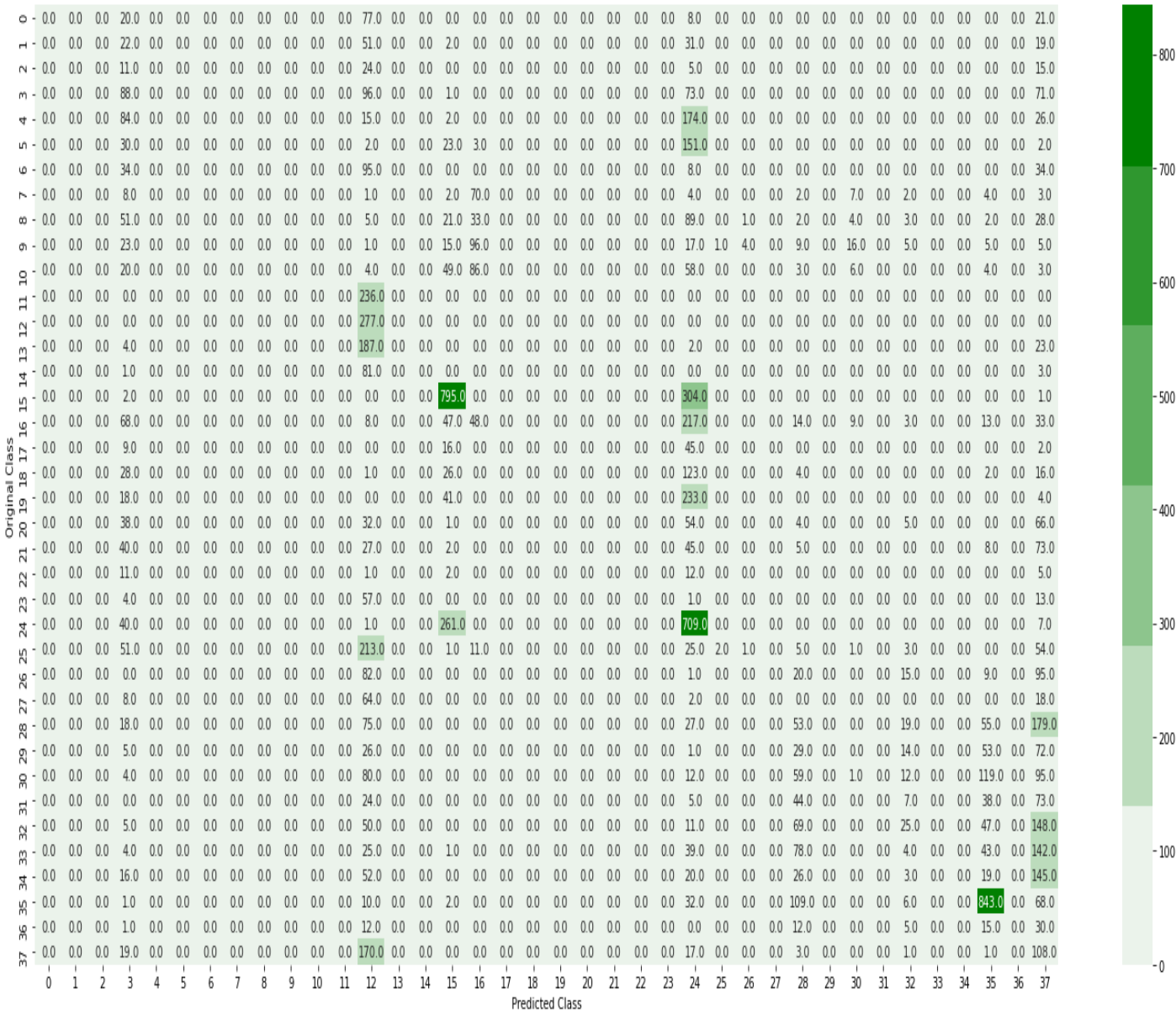
Graph Created by the Project For trained Images



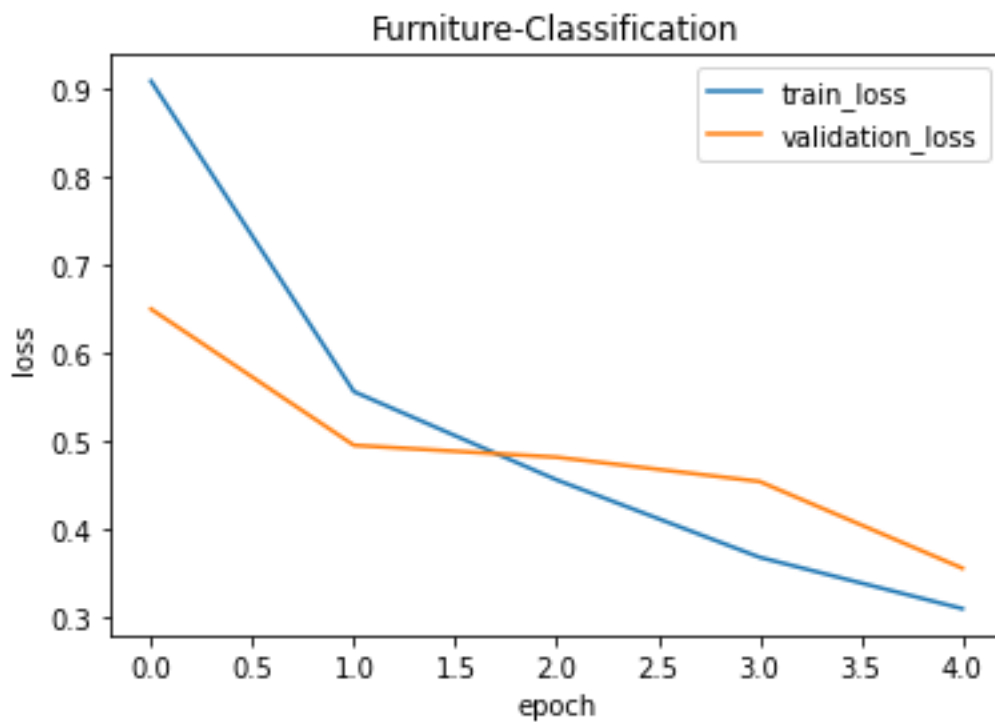
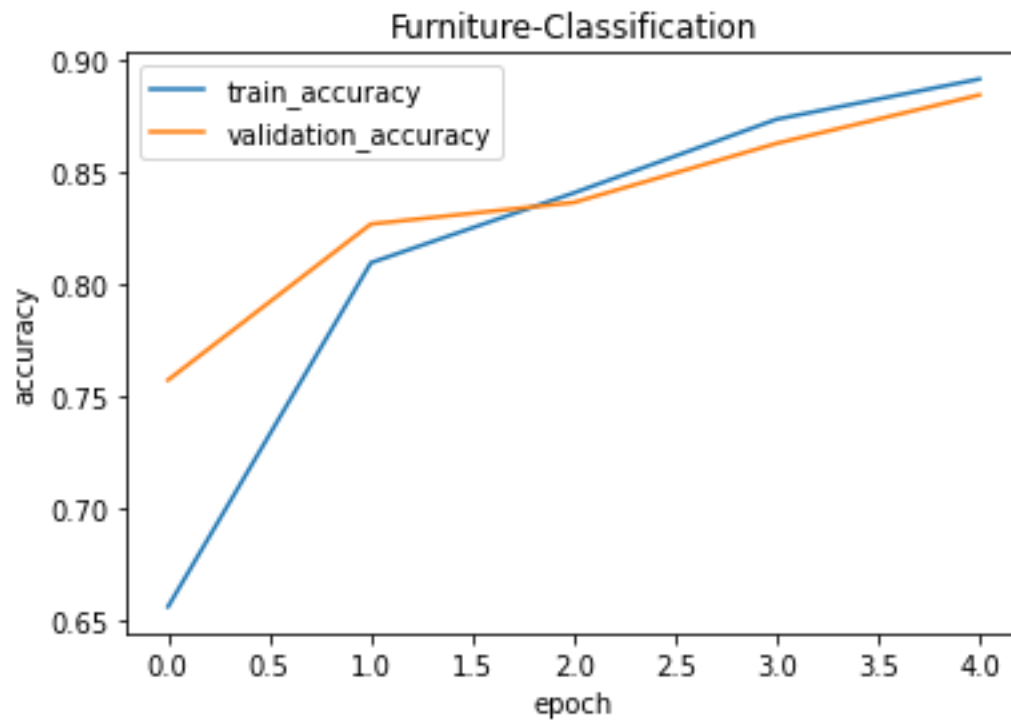
Confusion Matrix Created by the high accuracy software



Confusion Matrix Created by the previous accuracy software



Graph Created By the previous Project For trained Images



All trained images in the project

jupyter

Final_Year_Final_Code (autosaved)

Logout

FileEditViewInsertCellKernelWidgetsHelp

Not TrustedPython 3


+


%


Run


Code


plt.show()



















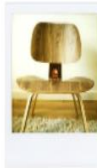
























































































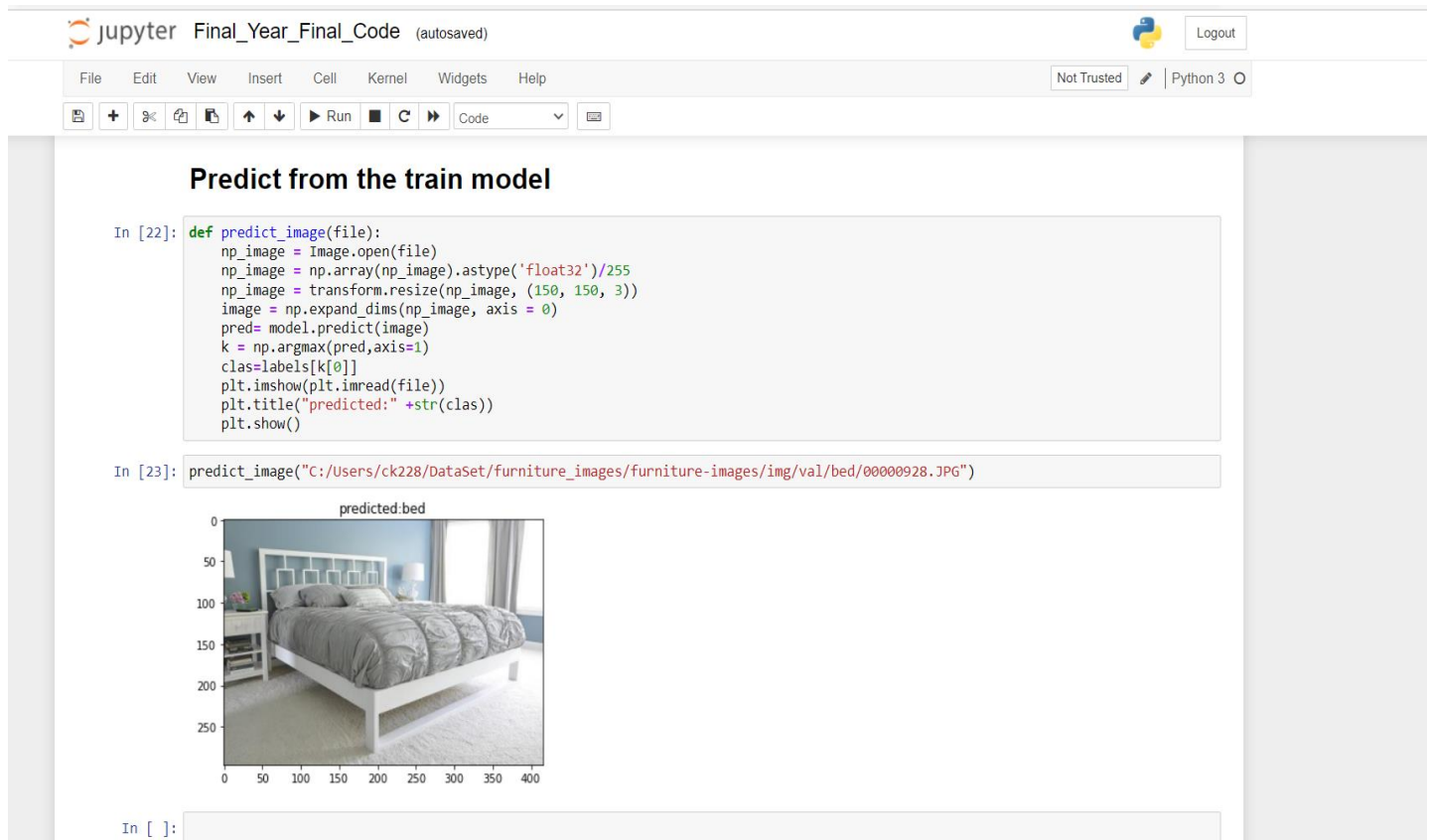








Sample image Result of the software



As we can see the software has predicted the image from the data set which we had introduced in the software to predict.

Waitage and result precision

```
from sklearn import metrics as m
```

```
print("Precision_weighted:", m.precision_score(y_test, y_pred,
average="weighted")*100)
print("Recall_weighted:", m.recall_score(y_test, y_pred, average="weighted")*100)
print("F1_weighted:", m.f1_score(y_test, y_pred, average="weighted")*100)
```

```
Precision_weighted: 24.565101061151662
Recall_weighted: 24.58628841607565
F1_weighted: 24.319665371410995
```

```
labels_map = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels_map.items())
predict = [labels[k] for k in y_pred]
```

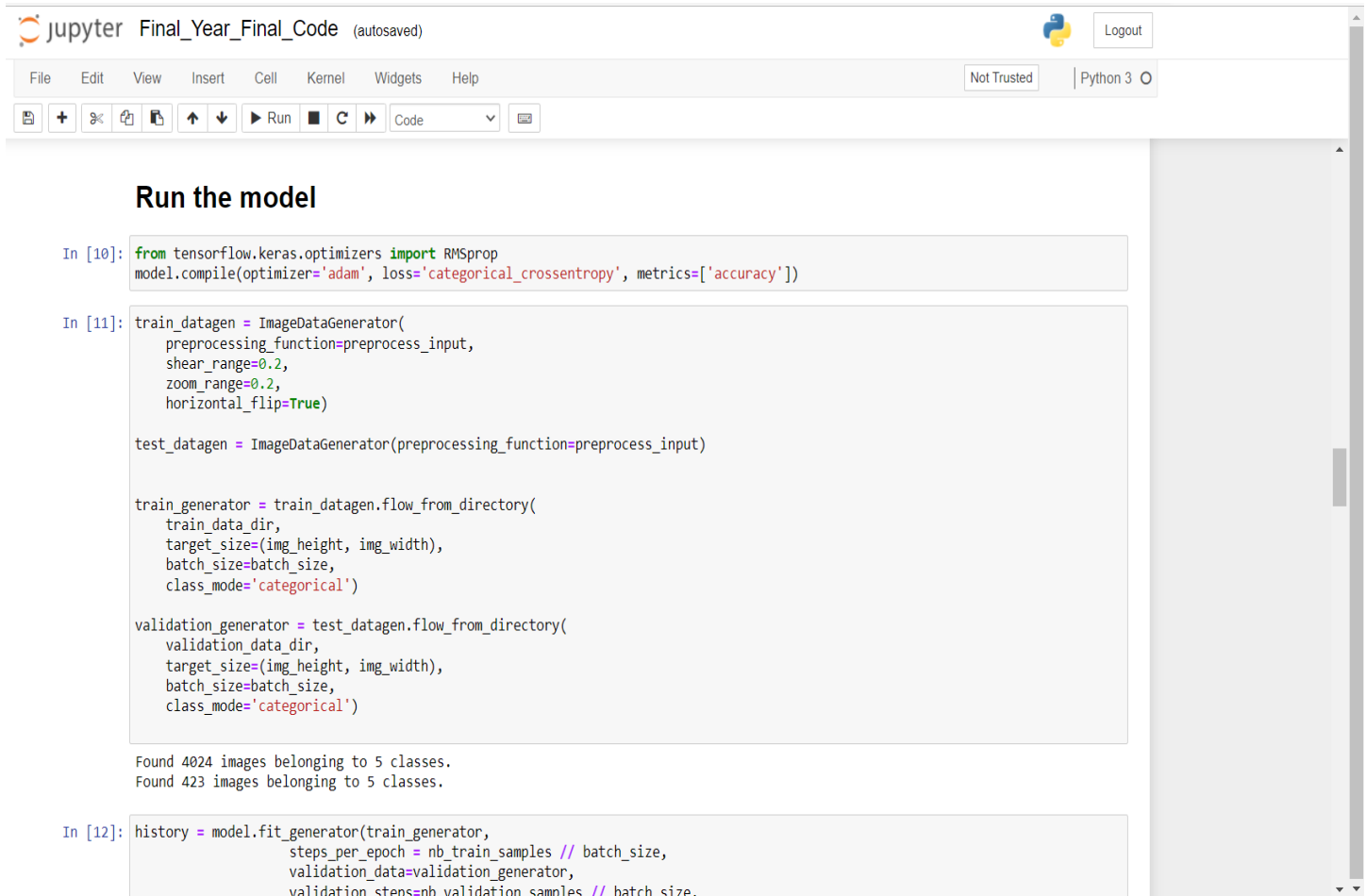
```
filenames = validation_generator.filenames
results = pd.DataFrame({"Filename":filenames,"Predictions":predict})
```

```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y,predict_y):
    c = confusion_matrix(test_y, predict_y)
```

```
    print("Number of misclassified images: ", (len(test_y)-np.trace(c)))
    print("Percentage of misclassified images: ", (len(test_y)-np.trace(c))*100/len(test_y))
```

```
    labels = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
    cmap = sns.light_palette("green")
    print("-"*50, "Confusion Matrix", "-"*50)
    plt.figure(figsize=(25,12))
    sns.heatmap(c, annot = True, cmap=cmap, fmt=".1f", xticklabels=labels,
yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

Running model images calculations and classes



The image shows a Jupyter Notebook interface with the title 'Final_Year_Final_Code (autosaved)'. The top bar includes a 'Logout' button and a 'Python 3' kernel indicator. The menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The toolbar has icons for saving, adding cells, undo, redo, and running code. The notebook content is titled 'Run the model' and contains three code cells. The first cell imports RMSprop and compiles the model. The second cell creates ImageDataGenerators for training and testing, and flows them into generators. The third cell fits the model using the generators. The output of the second cell shows the number of images found for each class.

```
In [10]: from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

In [11]: train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical')

Found 4024 images belonging to 5 classes.
Found 423 images belonging to 5 classes.

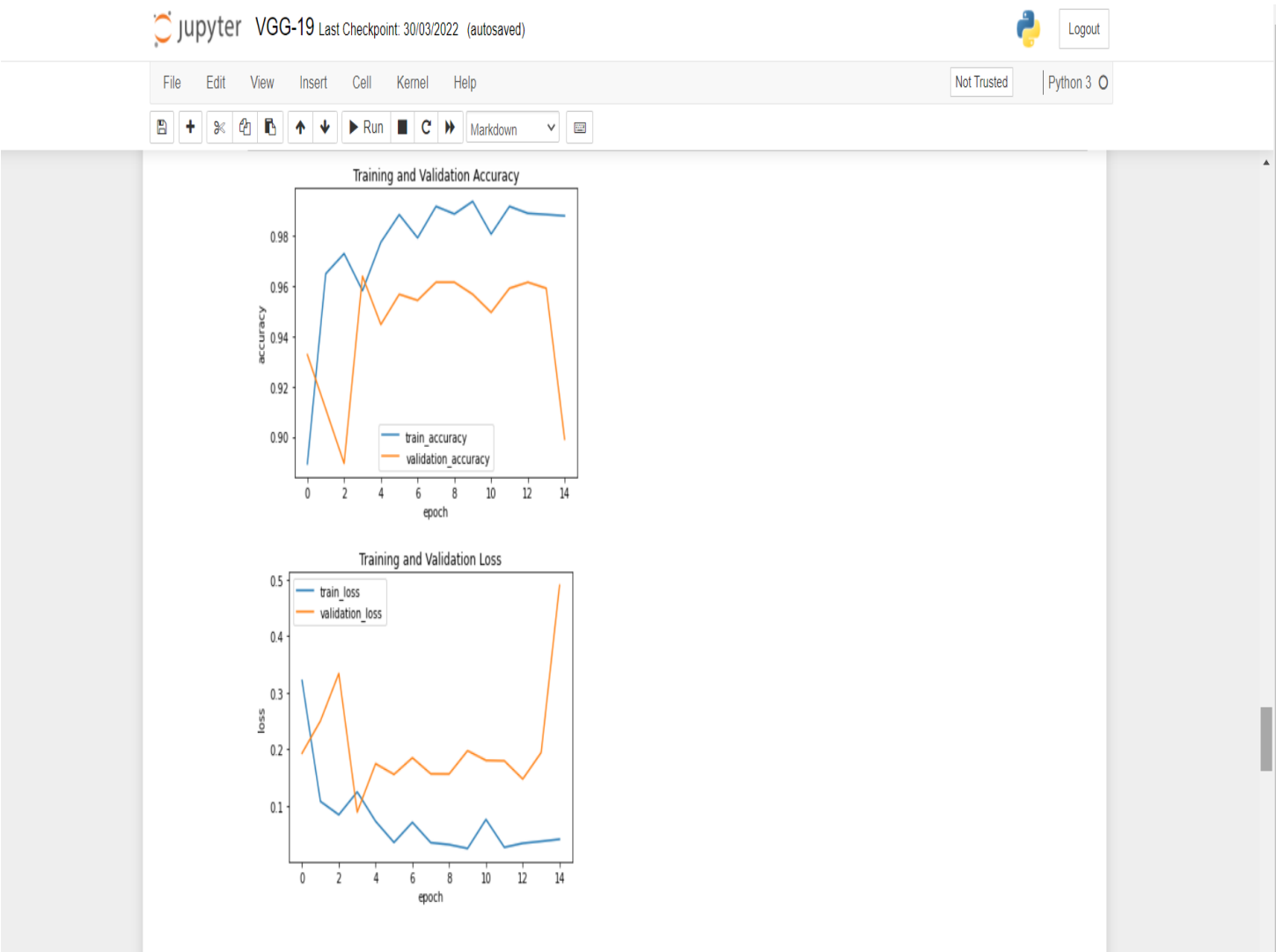
In [12]: history = model.fit_generator(train_generator,
    steps_per_epoch = nb_train_samples // batch_size,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
```

Total calculated images – **4024**

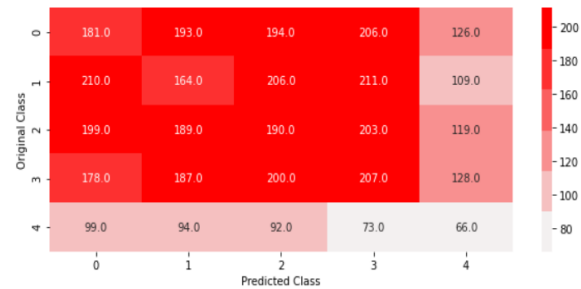
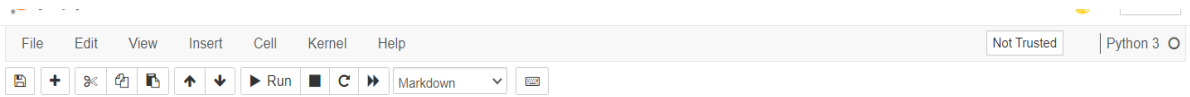
Total classes calculated which was introduced in the project – **5**

Result after the training and running the model

Graph depicted after the training



Final Confusion matrix created after training the project



Prediction

```
In [22]: from PIL import Image
from skimage import transform
def predict_image(file):
    np_image = Image.open(file)
    np_image = np.array(np_image).astype('float32')/255
    np_image = transform.resize(np_image, (256, 256, 3))
    image = np.expand_dims(np_image, axis = 0)
    pred = model.predict(image)
    k = np.argmax(pred,axis=1)
    clas=labels[k[0]]
    plt.imshow(plt.imread(file))
    plt.title("predicted:" +str(clas))
    plt.show()
```

Final Training of the images

jupyter VGG-19 Last Checkpoint: 30/03/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Help

Not Trusted

Python 3

Run

Confusion Matrix :-

Training :-

```
In [13]: train_generator.reset()
predictions = model.predict_generator(generator = train_generator)
y_pred = [np.argmax(probas) for probas in predictions]
y_test = train_generator.classes
```

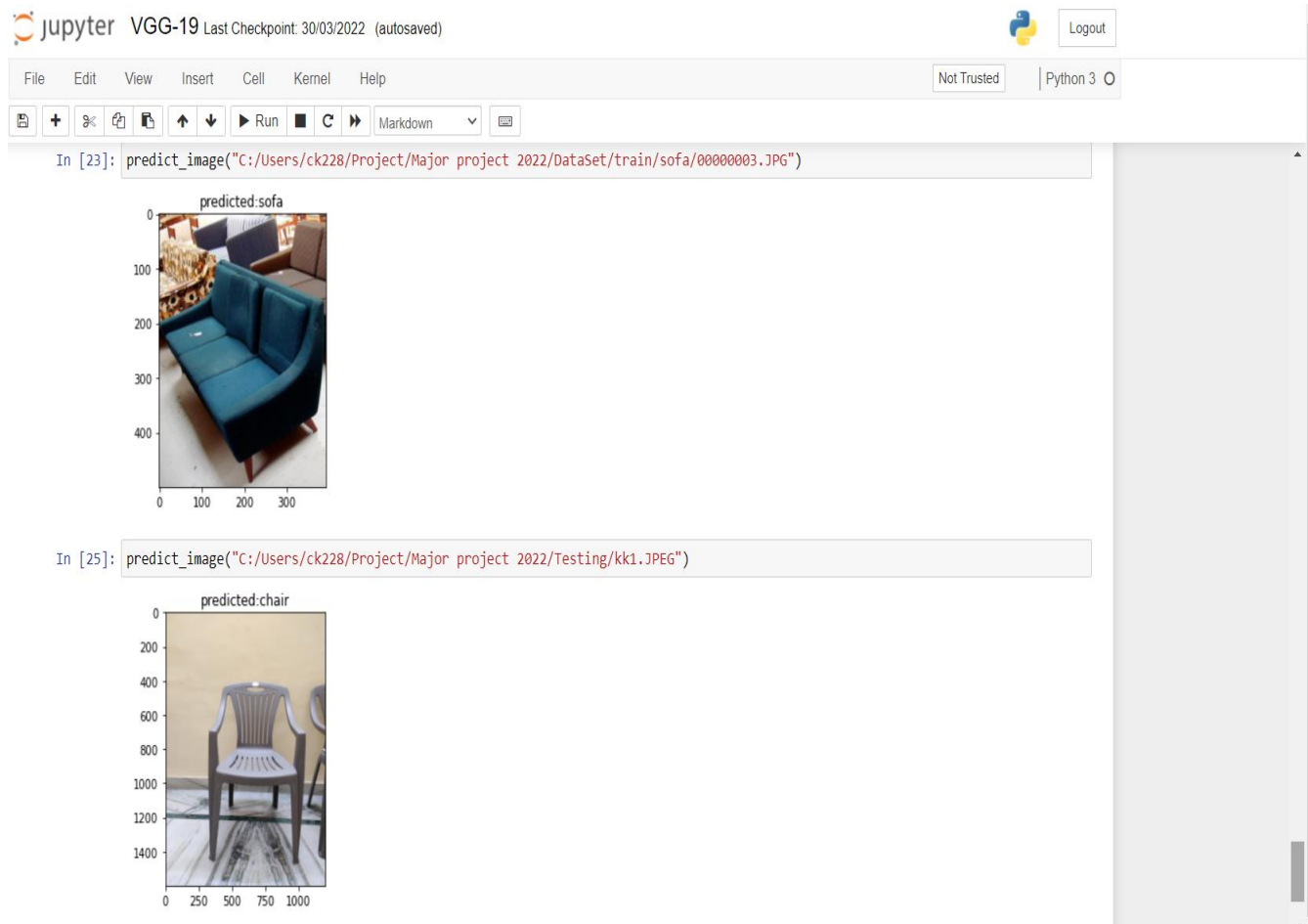
WARNING:tensorflow:From <ipython-input-13-78cb6ab4ebd2>:2: Model.predict_generator (from tensorflow.python.keras.engine.trainin
g) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.predict, which supports generators.

```
In [14]: labels_map = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels_map.items())
predict = [labels[k] for k in y_pred]

filenames = train_generator.filenames
results = pd.DataFrame({"Filename":filenames,"Predictions":predict})
```

```
In [15]: from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y,predict_y):
    c = confusion_matrix(test_y, predict_y)
    cmap = sns.light_palette("red")
    plt.figure(figsize=(10,4))
    sns.heatmap(c, annot = True, cmap=cmap, fmt=".1f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

Final Result of the software with testing



As we can see the software has predicted the image.

The testing of the software shows that the 2 images sofa and chair were introduced to predict the image.

RESULTS

The accuracy, F1 score, and confusion matrix of our dataset were calculated by the authors to evaluate our suggested technique. The degree to which the estimated findings reflect the ground reality is referred to as accuracy. The performance and accuracy of our proposed model were compared to [4], where the author estimated their performance and accuracy value is 98%. Our proposed approach uses DenseNet-121, VGG-16, VGG-19, InceptionV3 and three layers To identify the photos of the CNN, which is a more efficient neural network design. leaf disease dataset and VGG-16 gives the highest accuracy 99.89%. Our proposed model's (VGG-19) training and validation loss, as well as the training and validation accuracy, are plotted on a graph., Confusion matrix is shown in figure.

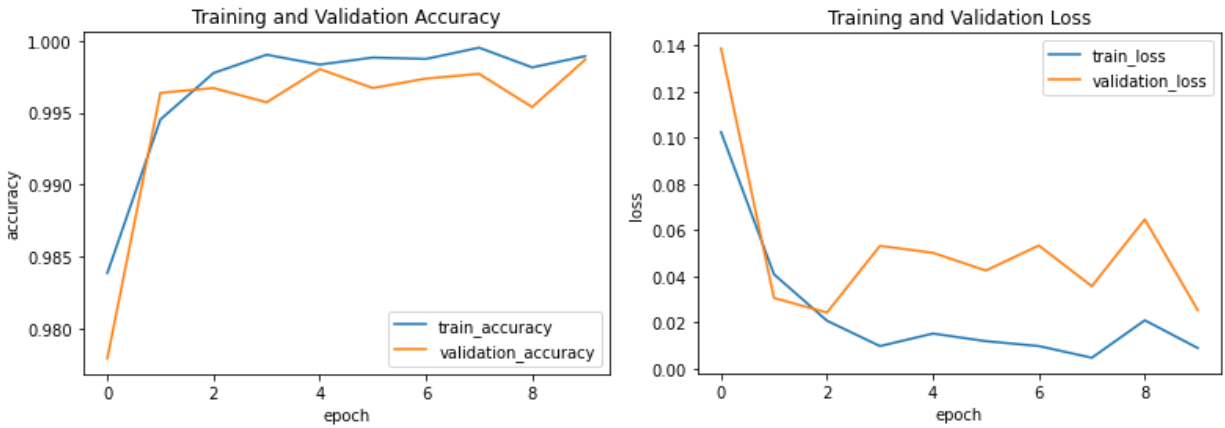


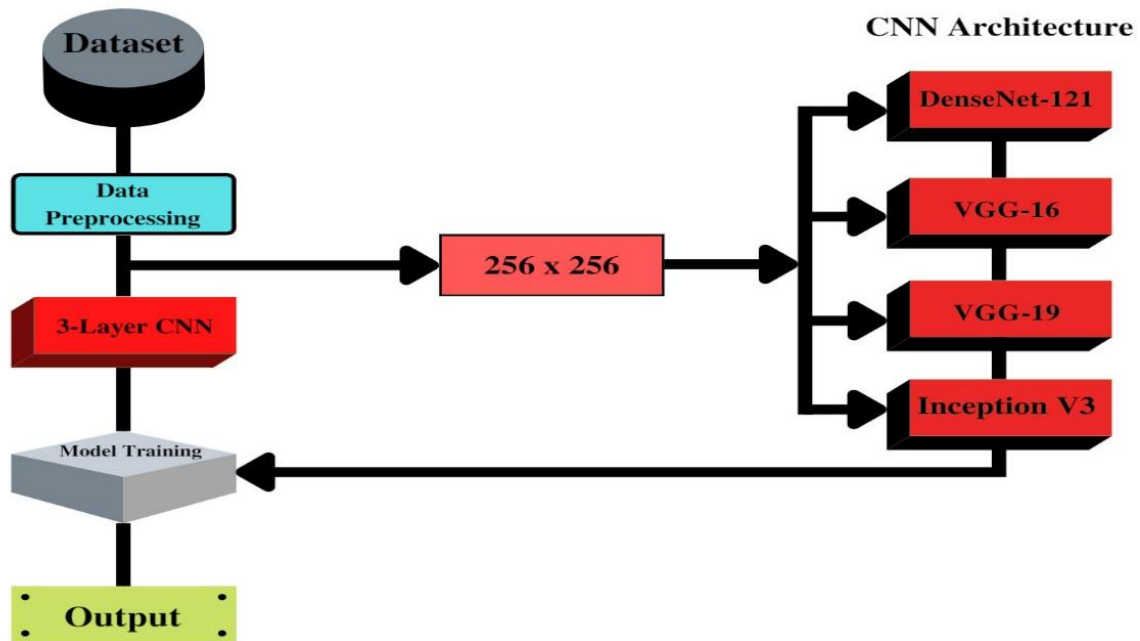
Fig-3 and Fig-4: Graphical Result

Table-3: Comparative Analysis of Different Models based on Training and Validation Accuracy.

Model	Train Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
3-Layer CNN	94.64	91.23	90.36
VGG-16	98.16	95.22	95.69
VGG-19	99.89	99.87	98.47
DenseNet-121	92.49	91.78	91.18
Inception V3	90.75	88.96	85.60

IMPLEMENTATION WORK

To train our images dataset authors used 3 convolutional neural networks and various pretrained models (VGG-16, VGG-19, DenseNet-121, InceptionV3) which is better performed on the real time object detection data set earlier. During model training, the author specified various hyperparameters (learning rate (lr) to 0.001, batch size to 32, starting function ReLu and Softmax, epoch size 10) and employed batch normalization during parameter building to avoid overfitting and underfitting. For model training, the ReLu and Softmax activation functions were utilised, and the Tensorflow and Keras frameworks were used for implementation. The first layer is a convolution layer with 64 filters, followed by a stack of three CNN blocks, each containing 32, 32 and 128 filters, with dimension reduction performed for each layer in the CNN block. In VGG-16, VGG-19 and DenseNet-121 having one filter with size 1024. After the data pre-processing and model building, we trained our model over 10 epochs utilizing the TensorFlow and Keras frameworks with CPU processing itself. Fig-2 demonstrates the implementation scenario.



DATASET AND METHODS for final set.

Dataset:

Five classes self-made dataset is used for this study which contains bed, chair, sofa, swivel chair and table images. We train our model on five class of furniture image classes in which bed has 9000 training and 100 validation images, chair has 9000 training and 100 validation images, sofa has 9000 training and 100 validation images, swivel chair has 9000 training and 100 validation images and table has 9000 training and 100 validation images. The total number of image samples in dataset is 5000. Table-1 below represent the structure of our dataset.

Table-1: Number of classes and total images in dataset

Classes	Training Images	Validation Images	Total Images
Bed	900	100	1000
Chair	900	100	1000
Sofa	900	100	1000
Swivel Chair	900	100	1000
Table	900	100	1000
Total Images	4500	500	5000

B. Image Pre-processing:

During image pre-processing our dataset images are compromised with the size of 256 x 256 pixel from 1572 x 1548 pixel to minimize the background area and 256 x 256 pixel is best fit for convolutional neural networks. The compromised image is then utilised for validation and training. Dataset on DenseNet-121, InceptionV3, VGG-16, VGG-19 and CNN with three layers. During model training authors set various hyperparameters eg, training and validation split to 0.1 during image pre-processing, rescale to 1./255, shear range to 0.2, zoom range to 0.2 .

C. Methods:

Contemporary, Convolutional Neural Networks Because of its capacity to extract features from images without complex pre-processing, as well as transfer learning and fine-tuning parameters, it is a state-of-the-art approach. These types of study uses VGG-16, VGG-19, DenseNet-121, and InceptionV3, which make use of transfer learning often used in deep learning. We use transfer learning receive the quality vector for arranging furniture (object) using CNN and differentiate the results to decide which learning is the perfect for object detection. Table-2 below presents the various convolutional neural networks models over different criteria.

Table-2: CNN Models over different Criteria

Model Name	Size (MB)	Parameters (Millions)	Depth
VGG-16	528	138.3	23
VGG-19	549	143.6	26

DenseNet-121	33	8	121
Inception V3	92	23.8	159

1). 3 Layer CNN: 3-layer CNN consists of a convolutional layer, a pooling layer, and a fully connected layer. The CNN's main building block is the convolution layer [3]. It accounts for most of the computational load of the network. The pooling layer uses the summary statistics of neighboring outputs to transform the outputs of the network at specific locations. This minimizes the spatial size of the representation, thereby reducing the amount of computation and load required. As in a conventional fully convolutional neural network, fully connected layers have full connection with all neurons in the preceding and subsequent layers. The completely linked layer aids in the mapping of the input and output representations.

2). DenseNet-121: A DenseNet is a type of convolutional neural network that employs dense connections between layers through dense blocks, with all layers directly connected to each other. DenseNet was created to address the problem of decreased accuracy caused by the longer path between the input and output layers, where information evaporates before it reaches its goal [6].

3). VGG-16: Visual Geometry Group is a convolutional neural networks architecture. They concentrated on having 3x3 convolution layers because it has a large number of hyper-parameters. filter size [2]. The VGG-16 convolutional neural network architecture is a simple and extensively used convolutional neural network design. VGG-16 is used in many deep learning image classification techniques and is popular due to its ease of implementation. VGG-16 is extensively used in learning applications due to the advantage that it has. In VGG-16 the number 16 defines the layers and depth. This CNN network has very large network approx 138 million parameters.

4). VGG-19: VGG-19 is a convolutional neural network that is a variant of the VGG model, with a total of 19 layers (16 convolution layers, 5 MaxPool layers, 3 fully connected layers and 1 softmax layer) [7]. Only 33 convolutional layers are placed on top of each other in increasing order of depth in this convolutional neural network architecture. This is a very large network, it has approx 143 million parameters.

5). Inception V3: The image recognition model Inception V3 is very popular. Convolution, Average Pooling, Max Pooling, Concats, Dropouts, and Fully Linked Layers are some of the symmetric and asymmetric building elements that make up the model [1]. Activation inputs are subjected to batch normalisation, which is employed throughout the model. The Softmax method is used to calculate the loss.

Final accuracy reached by the project

Epoch 1/10

176/176 [=====] - 3638s 21s/step - loss: 0.7035 - accuracy: 0.7686 - val_loss: 0.4168 - val_accuracy: 0.8467

Epoch 2/10

176/176 [=====] - 2176s 12s/step - loss: 0.4223 - accuracy: 0.8558 - val_loss: 0.4254 - val_accuracy: 0.8626

Epoch 3/10

176/176 [=====] - 2162s 12s/step - loss: 0.3452 - accuracy: 0.8887 - val_loss: 0.6516 - val_accuracy: 0.8278

Epoch 4/10

176/176 [=====] - 17296s 98s/step - loss: 0.3613 - accuracy: 0.8812 - val_loss: 0.3634 - val_accuracy: 0.8815

Epoch 5/10

176/176 [=====] - 1996s 11s/step - loss: 0.3055 - accuracy: 0.8942 - val_loss: 0.3865 - val_accuracy: 0.8866

Epoch 6/10

176/176 [=====] - 2098s 12s/step - loss: 0.2359 - accuracy: 0.9158 - val_loss: 0.4773 - val_accuracy: 0.8445

Epoch 7/10

176/176 [=====] - 2034s 12s/step - loss: 0.2641 - accuracy: 0.9156 - val_loss: 0.3856 - val_accuracy: 0.8903

Epoch 8/10

176/176 [=====] - 2010s 11s/step - loss: 0.2553 - accuracy: 0.9169 - val_loss: 0.3590 - val_accuracy: 0.9012

Epoch 9/10

176/176 [=====] - 3525s 20s/step - loss: 0.2556 - accuracy: 0.9116 - val_loss: 0.3568 - val_accuracy: 0.8903

Epoch 10/10

176/176 [=====] - 2351s 13s/step - loss: 0.2232 - accuracy: 0.9229 - val_loss: 0.3052 - val_accuracy: 0.8939

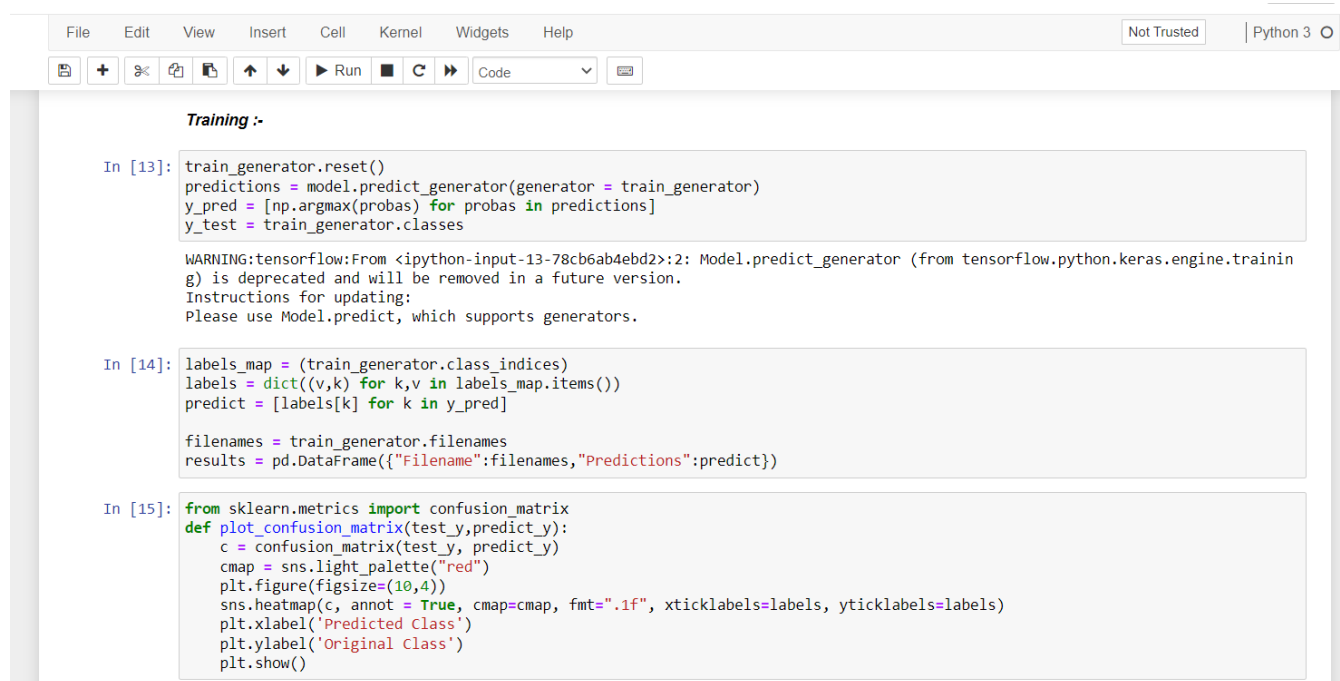
Final model layers

Model: "functional_1"

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 256, 256, 3)]	0
block1_conv1 (Conv2D)	(None, 256, 256, 64)	1792
block1_conv2 (Conv2D)	(None, 256, 256, 64)	36928
block1_pool (MaxPooling2D)	(None, 128, 128, 64)	0
block2_conv1 (Conv2D)	(None, 128, 128, 128)	73856
block2_conv2 (Conv2D)	(None, 128, 128, 128)	147584
block2_pool (MaxPooling2D)	(None, 64, 64, 128)	0
block3_conv1 (Conv2D)	(None, 64, 64, 256)	295168
block3_conv2 (Conv2D)	(None, 64, 64, 256)	590080
block3_conv3 (Conv2D)	(None, 64, 64, 256)	590080
block3_conv4 (Conv2D)	(None, 64, 64, 256)	590080
block3_pool (MaxPooling2D)	(None, 32, 32, 256)	0
block4_conv1 (Conv2D)	(None, 32, 32, 512)	1180160
block4_conv2 (Conv2D)	(None, 32, 32, 512)	2359808
block4_conv3 (Conv2D)	(None, 32, 32, 512)	2359808
block4_conv4 (Conv2D)	(None, 32, 32, 512)	2359808
block4_pool (MaxPooling2D)	(None, 16, 16, 512)	0
block5_conv1 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv4 (Conv2D)	(None, 16, 16, 512)	2359808
block5_pool (MaxPooling2D)	(None, 8, 8, 512)	0
flatten (Flatten)	(None, 32768)	0

```
dense (Dense)                (None, 5)                163845
=====
Total params: 20,188,229
Trainable params: 163,845
Non-trainable params: 20,024,384
```

Final Training of the images process



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook is titled "Training :-". It contains three code cells. The first cell (In [13]) resets the training generator and makes predictions. The second cell (In [14]) maps the predicted classes to the original classes and saves the results. The third cell (In [15]) imports the confusion matrix and plots it. The output of the first cell shows a warning from TensorFlow about the deprecated `Model.predict_generator` method.

```
Training :-

In [13]: train_generator.reset()
predictions = model.predict_generator(generator = train_generator)
y_pred = [np.argmax(probas) for probas in predictions]
y_test = train_generator.classes

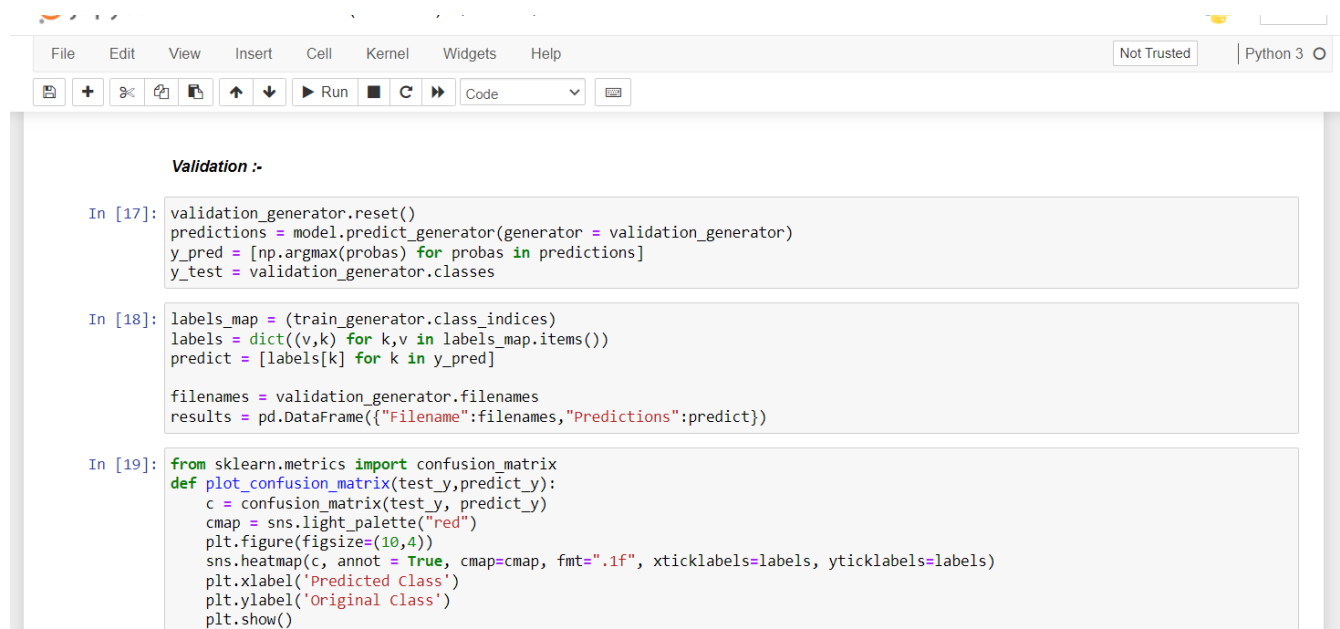
WARNING:tensorflow:From <ipython-input-13-78cb6ab4ebd2>:2: Model.predict_generator (from tensorflow.python.keras.engine.trainin
g) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.predict, which supports generators.

In [14]: labels_map = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels_map.items())
predict = [labels[k] for k in y_pred]

filenames = train_generator.filenames
results = pd.DataFrame({"Filename":filenames,"Predictions":predict})

In [15]: from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y,predict_y):
    c = confusion_matrix(test_y, predict_y)
    cmap = sns.light_palette("red")
    plt.figure(figsize=(10,4))
    sns.heatmap(c, annot = True, cmap=cmap, fmt=".1f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

Final Validation of the images process



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook is titled "Validation :-". It contains three code cells. The first cell (In [17]) resets the validation generator and makes predictions. The second cell (In [18]) maps the predicted classes to the original classes and saves the results. The third cell (In [19]) imports the confusion matrix and plots it.

```
Validation :-

In [17]: validation_generator.reset()
predictions = model.predict_generator(generator = validation_generator)
y_pred = [np.argmax(probas) for probas in predictions]
y_test = validation_generator.classes

In [18]: labels_map = (train_generator.class_indices)
labels = dict((v,k) for k,v in labels_map.items())
predict = [labels[k] for k in y_pred]

filenames = validation_generator.filenames
results = pd.DataFrame({"Filename":filenames,"Predictions":predict})

In [19]: from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y,predict_y):
    c = confusion_matrix(test_y, predict_y)
    cmap = sns.light_palette("red")
    plt.figure(figsize=(10,4))
    sns.heatmap(c, annot = True, cmap=cmap, fmt=".1f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
```

CONCLUSIONS AND FUTURE WORK

Accurately locating an object in a surveillance video is one of the most important research areas in computers imaginable and has a wide range of cutting-edge programs in modern times. In the present day it is very difficult to cut modern day leaves such as low resolution, models of lights, moving objects beyond the ancient, small adjustments in the historical past, due to subsequent gadget photographs obtained from a surveillance video. We have presented a top degree visual development in item detection strategies. The detection approach takes place in background modelling, item detection, and object categories. In this paper, all available item detection strategies are classified into history subtraction, optical float and spatial-temporal filter out techniques and the advantages and drawbacks of today's techniques implemented in many modern-day datasets are referenced. Object type techniques are further categorized into strategies based on form-based thoroughness, movement-based and texture-based altogether.

In this project we presented a comparative study of five Convolutional Neural Network Models classification of object detection. Out of the models under study VGG-19 outperformed all other in terms of accuracy. VGG-19 achieved an accuracy of 99.89% on training dataset, 99.87% accuracy on validation dataset and after testing on different furniture images, model obtained 98.47% accuracy.

References

- [1] Li, K., Wan, G., Cheng, G., Meng, L., & Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159, 296-307.
- [2] Dhillon, A., & Verma, G. K. (2020). Convolutional neural network: a review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, 9(2), 85-112.
- [3] Chandan, G., Jain, A., & Jain, H. (2018, July). Real time object detection and tracking using Deep Learning and OpenCV. In 2018 International Conference on inventive research in computing applications (ICIRCA) (pp. 1305-1308). IEEE.
- [4] Zhiqiang, W., & Jun, L. (2017, July). A review of object detection based on convolutional neural network. In 2017 36th Chinese control conference (CCC) (pp. 11104-11109). IEEE.
- [5] Jia, B., Pham, K. D., Blasch, E., Wang, Z., Shen, D., & Chen, G. (2018, March). Space object classification using deep neural networks. In 2018 IEEE Aerospace Conference (pp. 1-8). IEEE.
- [6] Wu, X., Sahoo, D., & Hoi, S. C. (2020). Recent advances in deep learning for object detection. *Neurocomputing*, 396, 39-64.
- [7] Yanagisawa, H., Yamashita, T., & Watanabe, H. (2018, January). A study on object detection method from manga images using CNN. In 2018 International Workshop on Advanced Image Technology (IWAIT) (pp. 1-4). IEEE.
- [8] Prabhakar, G., Kailath, B., Natarajan, S., & Kumar, R. (2017, July). Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving. In 2017 IEEE region 10 symposium (TENSYP) (pp. 1-6). IEEE.
- [9] Uçar, A., Demir, Y., & Güzeliş, C. (2016, August). Moving towards in object recognition with deep learning for autonomous driving applications. In 2016

International Symposium on Innovations in Intelligent Systems and Applications (INISTA) (pp. 1-5). IEEE.

[10] Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, 14(9), 4224-4231.

[11] Deng, Z., Sun, H., Zhou, S., Zhao, J., Lei, L., & Zou, H. (2018). Multi-scale object detection in remote sensing imagery with convolutional neural networks. *ISPRS journal of photogrammetry and remote sensing*, 145, 3-22.

[12] Sun, X., Wu, P., & Hoi, S. C. (2018). Face detection using deep learning: An improved faster RCNN approach. *Neurocomputing*, 299, 42-50.

[13] Ji, Y., Zhang, H., Zhang, Z., & Liu, M. (2021). CNN-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances. *Information Sciences*, 546, 835-857.

[14] Li, K., Wan, G., Cheng, G., Meng, L., & Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159, 296-307.

