A Project Report

on

INVINCIBLE DIGITAL IMAGE PROCESSING

Submitted in partial fulfillment of the
requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr Shobha Tyagi
Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1010613- ASHISH KUMAR SINHA
18SCSE1010142- HRITHIK RANA

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING / DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER , 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"DIGITAL IMAGE PROCESSING "** in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **Dr. Shobha Tyagi , PROFESSOR , Department of Computer Science and Engineering**, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010613 - ASHISH KUMAR SINHA

18SCSE1010142 - HRITHIK RANA

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Dr. SHOBHA TYAGI, PROFESSOR)

CERTIFICATE

The Final Project Viva-Voce examination of **18SCSE1010613 - ASHISH KUMAR SINHA, 18SCSE1010142 - HRITHIK RANA** has been held on _________________ and their work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place: Greater Noida

ACKNOWLEDGEMENT

Primarily We might thank God for having the ability to finish this mission with success .Then we would love to thank my mission manual Dr. Shobha Tyagi mam ,whose treasured steerage has been those that helped me patch this mission and make it complete evidence success, his pointers and his commands has served because the foremost contributor in the direction of the finishing touch of the mission.

Then We would love to thank our mother and father and friends who've helped me with their treasured pointers and steerage has been beneficial in diverse levels of the finishing touch of the mission.

Abstract

This project was assigned to seventh semester students for a better understanding of how to work on projects given by the department of computer science and engineering, GU. Computer vision technologies in combination with cameras to achieve image recognition have helped the machine to perceive, understand and interact with real world objects. The main objective of this project is to design and build a system which automatically identifies an image by implementing machine learning models and image processing techniques. Some of the machine learning models used are Google Lens, Instagram clone. Image classification is an important tool for extracting information from digital images. Image classification is a complex process that may be affected by many factors. This paper examines current practices, problems, and prospects of image classification. The emphasis is placed on the summarization of major advanced classification approaches and the techniques used for improving classification accuracy. In addition, some important issues affecting classification performance are discussed. The main role of image classification is to detect, recognize and classify the features of an object in an image depending on the type of class. For the fact that many people may not be very good at recognizing a particular object and its specification, our application will present a set of options for correctly verifying the selected Object. On completion of this project, we had a better understanding of machine learning, image processing which is our primary focus of this project.

Keywords:

Initial Topic: Image Classification

Broader Terms:

- TensorFlow
- CNN (Convolutional Neural Networks)
- Python
- Keras

Related Terms:

- Relevant Datasets
- Data abstraction
- Model training
- Evaluation of Model

Table of Contents

List of Table

List of Figures

Acronyms

CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
AI	Artificial Intelligence
ML	Machine Learning
CIFAR	Canadian Institute for Advanced Research

CHAPTER-1

Introduction

1.1 .1 Introduction of Project:

Lillsand and Kiefer defined image processing as involving manipulation of digital images with the use of computers. It is a broad subject and involves processes that are mathematically complex [1]. Image processing involves some basic operations namely image restoration/rectification, image enhancement, image classification, images fusion etc. Image classification forms an important part of image processing. The objective of image classification is the automatic allocation of images to thematic classes [1]. Two types of classification are supervised classification and unsupervised classification. The process of image classification involves two steps, training of the system followed by testing. The training process means to take the characteristic properties of the images (form a class) and form a unique description for a particular class. The process is done for all classes depending on the type of classification problem; binary classification or multi-class classification. The testing step means to categorize the test images under various classes for which system was trained. This assigning of class is done based on the partitioning between classes based on the training features. Artificial Intelligence(AI) is now at the heart of innovation economy and thus the base for this project is also the same. In the recent past a field of AI namely Deep Learning has turned a lot of heads due to its impressive results in terms of accuracy when compared to the already existing Machine learning algorithms. The task of being able to generate a meaningful sentence from an image is a difficult task but can have great impact, for instance helping the visually impaired to have a better understanding of images. The task of image captioning is significantly harder than that of image classification, which has been the main focus in the computer vision community. A description for an image must capture the relationship between the objects in the image. In addition to the visual understanding of the image, the above semantic knowledge has to be expressed in a natural language like English, which means that a language model is needed. The attempts made in the past have all been to stitch the two models together. Caption generation is an interesting artificial intelligence problem where a descriptive sentence is generated for a given image. It involves the dual techniques from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Image captioning has various applications such as recommendations in editing applications, usage in virtual assistants, for image indexing, for visually impaired persons, for social media, and several other natural language processing applications. Since 2006, deep structured learning, or more commonly called deep learning or hierarchical learning, has emerged as a new area of machine learning research [2]. Several definitions are available for Deep Learning; coating one of the many definitions from [2] Deep Learning is defined as: A class of machine learning techniques that exploit many layers of nonlinear information processing for supervised or unsupervised feature extraction and transformation and for pattern analysis and classification. This work aims at the application of Convolutional Neural Network or CNN for image classification. The image data used for testing the algorithm includes remote sensing data of aerial images and scene data from SUN database. The rest of the paper is organized as follows. Section 2 deals with the working of the network followed by section 2.1 with theoretical background. The working of CNN is described in section 2.2. Section 3 gives the experimental procedure in detail. Finally, section 4 deals with the experimental results obtained using CNN.

1.1 Problem Definition:

In day to day life we have seen lots of images on internet and almost everywhere like news, articles. Sometimes images having some short amount of description about it but out of them some images are just images and nothing extra as we are human we can figure out what’s in it. So we are trying to build a model that will take input image from user and then machine gives a suitable caption. So due to this our model can analyse thousands of images and then it will be used for test purposes to know what images says. It is very helpful in Artificial Intelligence to recognize images and gives responses to request sends comes from users.

1.1.1 Tool and Technology Used:

In the report we first consider the task of image classification separately. We try to classify the images of the cifar-10 dataset using various classifiers. We first try to train the model using a K-Nearest Neighbour classifier. Then we try to apply some linear classifiers. The accuracy with these models was much less than expected since a high loss factor at the time of classification will amplify the loss even further at the time of caption generation. We then try to train a simple Convolutional Neural Network and achieve decent results within few hours of training. Thus, by the end of this section we conclude that CNN are a good fit to be used as the image encoder for the captioning model .

Python:

Python is an interpreted , high-level , general-purpose programming language . Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace . Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and largescale projects

Chapter 2.

Literature Survey

The internet is now packed with an abundance of photos and videos, fostering search progress. Applications and algorithms that can analyse image and video semantic analysis[1] to present the user with Better content search and summarization of them. Significant breakthroughs have occurred in image marking, target recognition, Classification of scenes [2] [3], areas identified by various researchers worldwide. This adds to make it possible to formulate techniques for target identification and scene classification questions. Since artificial neural networks have been developed, In the field of object detection and scene classification, especially convolutionary neural classification, a performance breakthrough has been demonstrated. This thesis focuses on finding the right network for this reason (CNN)[4][5][6]. Extraction of functionality is a crucial component of the steps of these algorithms. Feature extraction from images requires extracting from low-level image pixel values a minimum number of features containing a high volume of object or scene detail, thus capturing the difference between the types of objects concerned. Scale-invariant feature transformation (SIFT)[7], histogram of directed gradients (HOG)[8], Local binary patterns (LBP)[10], Content-Based Model Retrieval (CBIR)[11], etc. are some of the conventional feature extraction strategies used on photos. Once characteristics have been extracted, their classification is dependent on objects that are present in an image. Help vector machines (SVM), Logistic Regression, Random Forest, decision trees etc. are a few examples of classifiers. In order to help understand the quality of an image, CNN has presented an organisational class of models, resulting in better image identification, segmentation, identification, and retrieval. In many pattern and image recognition applications, CNN's are used successfully and reliably, such as gesture recognition[14], face recognition[12], object classification[13], and scene description generation. Similarly, using the MNIST handwritten digit database[23], CNNs reached detection rates (CDRs) of 99.77 percent, 97.47 percent for the 3D object NORB dataset[24], and 97.6 percent on about 5600 images of more than 10 objects[25]. The efficient integration of all the mentioned applications is attributed to developments and advancement in deep network construction learning algorithms and a moderate open source wide labelled data set available for experimentation purposes, for example, ImageNet, CIFAR 10, 100, MNIST etc. [16] CNN has well-known trained networks that use these open-source network datasets and increase their classification efficacy after being trained over millions of images contained in CIFAR-100 and Image-Net datasets. Millions of tiny images are composed of the datasets used. They can thus simplify easily and correctly, and therefore effectively categorise the out-of-sample instances of the classes. It is important to note that when such comparisons are made on a broad data set such as Image-Net, CIFAR10, 100 etc, neural network classification and prediction accuracy and error rates are almost comparable to that of humans. The purpose of this work is to examine the capacity of convolutionary neural networks to categorise the video scene on the basis of objects defined. In the CIFAR-100, CIFAR 10 and ImageNet datasets for CNN training, a number of image categories are used. Images in multiple types and themes are the research datasets. Due to the role extraction capabilities of various CNNs, the contradiction branches out. TThe developing recognition of Machine learning plays a significant function in a wide scope of basic applications, for example, information mining, common language preparation, picture recognition, and master frameworks. Since Machine learning gives a potential arrangement in every one of those territories, it is supposed to be a mainstay of future development. AI contains a variety of building calculations that make the PC gain from the information and settling on information-driven choices just as expectations. The fast development of AI from the previous few years welcomes a huge impact on our everyday life with such instances of AI for insolvency expectation, precipitation determining, climate estimating, self-driving frameworks and optical character recognition and so forth. By consolidating AI approaches with counterfeit knowledge creates a superior outcome [3]. Despite the fact that machine learning shows an excellent presentation, it isn't productive in human data handling frameworks, for example, discourse acknowledgment and PC vision. This can be overwhelmed by the genuine forefront of Machine learning is Deep Learning. Image processing includes some fundamental activities to be specific: picture rebuilding/amendment, picture upgrade, image classification, pictures combination, and so forth. Image classification structures a significant piece of picture handling. The target of picture characterization is the programmed assignment of the picture to topical classes [4].

Since 2006, deep structured learning, or all the more usually called deep learning or progressive learning, has risen as a new region of AI research [5]. A few definitions are accessible for Deep Learning; covering one of the numerous definitions from [5] Deep Learning is characterized as: A class of AI methods that abuse numerous layers of nonlinear data handling for managed or solo highlight extraction and change and for design investigation furthermore, grouping. Computational models of neural networks have been around for quite a while, the first model proposed was by McCulloch and Pitts as in [6]. Neural networks are composed of various layers with each layer associated with different layers shaping the network. A feed forward neural network or FFNN can be thought of in terms of neural activation and the quality of the associations between each pair of neurons .

Feasibility Analysis

The feasibility of the project is analysed in this phase and a business proposal is put forth with a general plan for the project and some cost estimate. During project analysis the feasibility study of the proposed project is to be carried out. This is to ensure that the proposed project is not a burden to the company.

For feasibility analysis, some understanding of the major requirements for the project is essential. Three key considerations involved in the feasibility analysis are:

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

1. Economical Feasibility:

This study is carried out to check the economic impact the Project will have on the organization. The amount of funds that that company can pour into the research and development of the project is limited. The expenditures must be justified. Thus the developed project as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2. **Technical Feasibility:**

This study is carried out to check the technical feasibility, that is, the technical requirement of the project. Any project developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed project must have a modest requirement; as only minimal or null changes are required for implementing this system.

3. **Social feasibility:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Chapter 3.
Working of Project

CNN:

The Convolutionary Neural Networks (CNN) are used in a variety of tasks and have outstanding success in numerous applications. One of the first implementations where the CNN architecture was successfully applied was the identification of handwritten digits. With the development of CNN, networks with the invention of new layers and the intervention of numerous computer vision techniques have been constantly improved. In the ImageNet Challenge, Convolutionary Neural Networks are often used for different configurations of sketch datasets. Few of the researchers showed a link between the human subject and the recognition ability of a qualified network on image datasets. The comparative findings showed that human beings have an accuracy rate of 73.1 percent on the dataset, while the results of a qualified network have an accuracy rate of 64 percent. Similarly, the same was extended to Convolutionary NeuralNetworks as it yielded an accuracy of 74.9 percent, thereby outperforming the human accuracy rate. The techniques used often make use of the order of the strokes to achieve a much higher degree of accuracy. Studies are under way to explain the behaviour of the Deep Neural Network in different circumstances. These studies demonstrate how the effects of classification can be severely changed by minor changes made to an image. Also introduced in the work are pictures that are totally unrecognised by human beings but are identified by professional networks with high accuracy rates. In the field of attribute detectors and descriptors, there has been a lot of progress and many algorithms and techniques have been developed for object and scene classification. The resemblance between particle detectors, texture philtres, and philtre banks is usually appealing to us. There is an abundance of study in literature of object identification and scene classification. Researchers mostly use the latest up-to-date Felzenszwalb descriptors and Hoeim meaning classifiers.The idea of creating different object detectors for simple image analysis is analogous to the multimedia community's work in which a vast number of "semantic concepts" are used for image and video annotations and semantic indexing. Each semantic definition is learned by using either the image or frames of videos in the literature that applies to our work. Therefore, with many cluttered items in the scene, the approach is hard to use to understand the picture. The previous approaches concentrated on the identification and classification of single objects based on a human-defined feature set. Ses suggested methods to investigate the linkage of artefacts in the description of scenes. To measure its usefulness, several scene classification techniques were carried out on the object bank. Several styles of research have been carried out to emphasise their emphasis on low-level object recognition and classification feature extraction, namely Histogram of directed gradient (HOG), GIST, philtre bank, and feature abag (BoF) applied via term vocabulary.

Image Classification

The main aim of our work is to understand the performance of the networks for static as well as live video feeds. The first step for the following is to perform transfer learning on the networks with image datasets. This is followed by checking the prediction rate of the same object on static images and real- time video feeds. The different accuracy rates are observed and noted and presented in the tables given in further sections. Third important criteria for evaluating the performance was to check whether prediction accuracy varies across all CNNs chosen for the study. It must be noted that videos are not used as a training dataset, they are used as testing datasets. Hence we are looking for the best image classifier where the object is the main attribute for classification of scene categories. Different layers of the convolutional neural network used are:

• Input Layer: The first layer of each CNN used is ‘input layer’ which takes images, resizing them for passing onto further layers for feature extraction.

• Convolution Layer: The next few layers are ‘Convolution layers’ which act as filters for images, hence finding out features from images and also used for calculating the match feature points during testing.

• Pooling Layer: The extracted feature sets are then passed to ‘pooling layer’. This layer takes large images and shrinks them down while preserving the most important information in them. It keeps the maximum value from each window, it preserves the best fits of each feature within the window.

• Rectified Linear Unit Layer: The next ‘Rectified Linear Unit’ or ReLU layer swaps every negative number of the pooling layer with 0. This helps the CNN stay mathematically stable by keeping learned values from getting stuck at 0 or blowing up toward infinity.

• Fully Connected Layer: The final layer is the fully connected layer which takes the high-level filtered images and translates them into categories with labels.

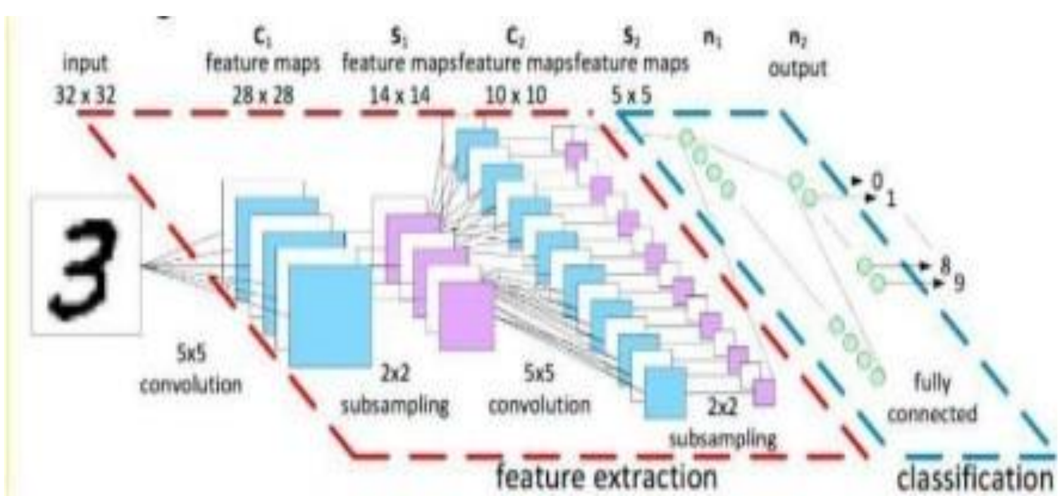


Fig. 1 Internal Layers of CNNs

The steps of proposed method are as follows:

Creating training and testing dataset: The super classes images used for training are resized [32,32] pixels and the dataset is divided into two categories i.e. training and validation data sets.

Modifying CNNs network: Replace the last three layers of the network with a fully connected layer, a softmax layer, and a classification output layer. Set the final fully connected layer to have the same size as the number of classes in the training data set. Increase the learning rate factors of the fully connected layer to train the network faster.

Train the network: Set the training options, including learning rate, mini-batch size, and validation data according to GPU specification of the system. Train the network using the training data.

Test the accuracy of the network: Classify the validation images using the fine-tuned network, and calculate the classification accuracy. Similarly testing the fine tune network on real time video feeds for accurate results.

Activity Time Schedule

Phase 1 (Research):

- Choose a topic
- Define the task and prepare a working theory.
- Brainstorm all possible sources
- Locate and evaluate sources for appropriateness for the project.
- Write a Report

Phase 2 (Implementation):

- Research the prerequisites
- Data Aggregation/ Mining/ Scraping
- Data Preparation/ Pre-processing/ Augmentation
- Model Implementation
- Training
- Evaluation

Design

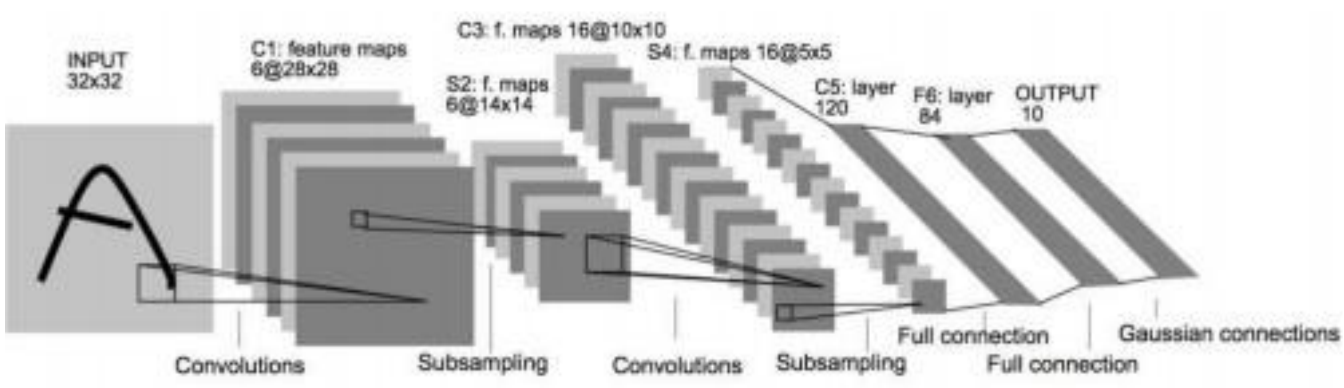
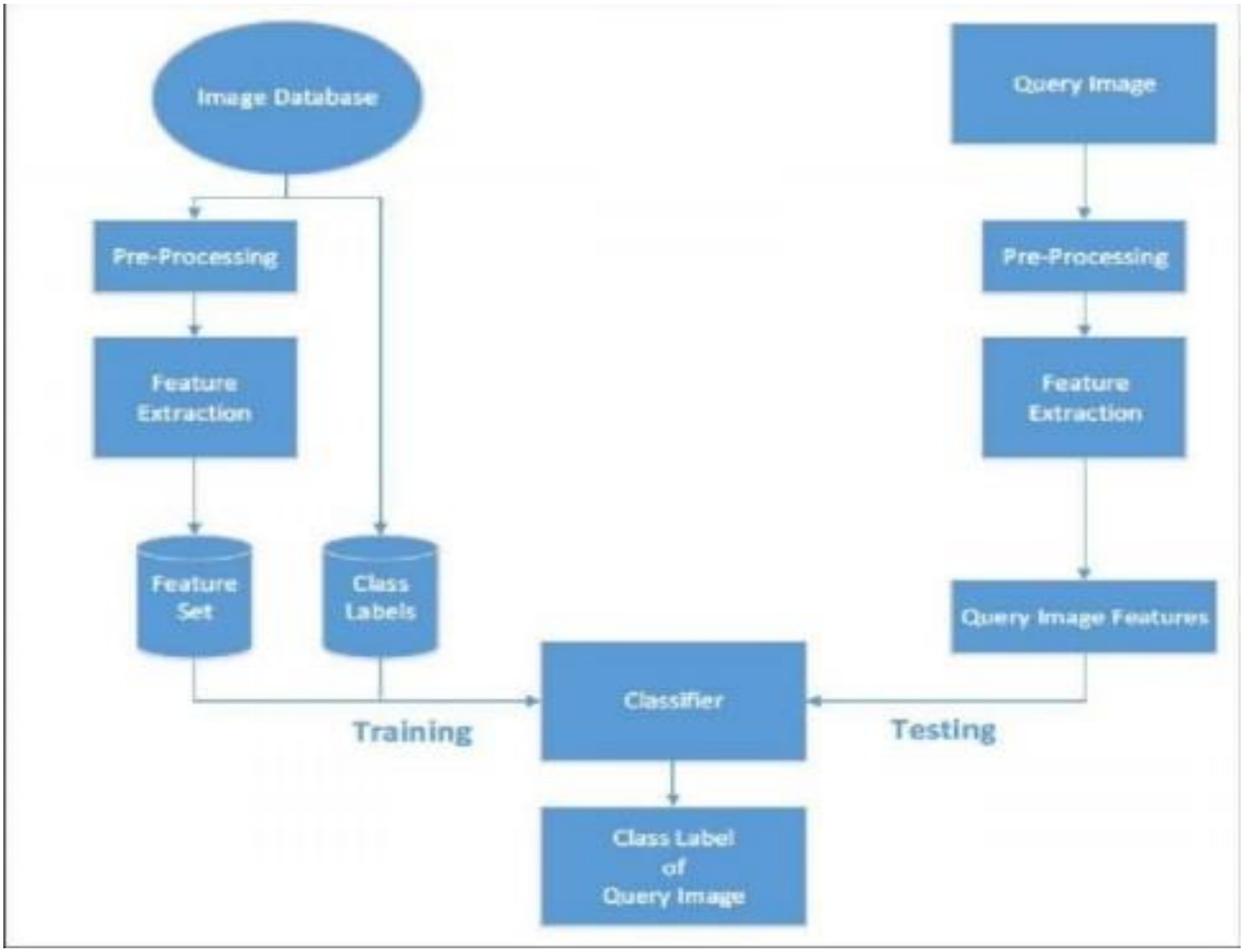


Figure 1. Image Feature Extraction on CNN



Flowchart of CNN Classification Model

Table 1. Performance of CNN's on CIFAR100 test dataset

CIFAR-100		AlexNet	GoogLeNet	ResNet50
Image Category	Bed	0.00%	70.80%	49.60%
	Bicycle	21.0	74.2%	55.00%
	Bus	84.00%	63.20%	36.80%
	Chair	90.00%	89.60%	57.60%
	Couch	11.00%	14.60%	76.40%
	Motorcycle	95.00%	74.60%	99.20%
	Streetcar	21.00%	0.84%	63.80%
	Table	00.00%	73.60%	33.40%
	Train	30.00%	95.60%	34.20%
	Wardrobe	89.00%	89.40%	92.20%

Table 2. Performance of CNN's on the CIFAR10 test dataset

CIFAR-10		AlexNet	GoogLeNet	ResNet50
Image Category	Airplane	41.80%	51.10%	90.80%
	Automobile	21.80%	62.10%	69.10%
	Bird	00.02%	56.70%	72.60%
	Cat	00.03%	78.80%	61.90%
	Deer	87.60%	49.50%	75.40%
	Dog	23.00%	57.50%	82.10%
	Frog	24.20%	90.20%	76.60%
	Horse	34.70%	78.20%	84.70%
	Ship	31.70%	95.50%	83.20%
	Truck	95.90%	97.10%	84.60%

Table 7. Performance of CNNs on live video feeds

Object Category	AlexNet Prediction Accuracy(%)	GoogleNet Prediction Accuracy(%)	ResNet50 Prediction Accuracy(%)	Object Category	AlexNet Prediction Accuracy(%)	GoogleNet Prediction Accuracy(%)	ResNet50 Prediction Accuracy(%)
Bed	12	85	25	Airplane	14	84	96
Bicycle	11	80	55	Automobile	12	59	56
Bus	14	74	25	Bird	11	45	53
Chair	12	47	30	Cat	11	62	49
Couch	12	25	90	Deer	12	45	33
Motorcycle	14	50	35	Dog	12	57	58
Streetcar	11	45	25	Frog	13	60	25
Table	11	63	50	Horse	12	87	65
Train	15	72	45	Ship	15	91	25
Wardrobe	14	84	32	Truck	22	95	52

Table 3. Performance on Bicycle class of CIFAR-100 dataset

AlexNet's Output	Prediction Accuracy (%)	GoogLeNet's Output	Prediction Accuracy (%)	ResNet50 Output	Prediction Accuracy (%)
Motorcycle	45	Bicycle	74.2	Bicycle	55
Bus	28	Train	13	Motorcycle	35
Bicycle	21	Table	7.6	Streetcar	4.4
Chair	2	Motorcycle	4.4	Couch	2.6
Train	2	Chair	0.4	Bed	1
Streetcar	1	Wardrobe	0.2	Train	0.8
Wardrobe	1	Bus	0.2	Wardrobe	0.6
Couch	0	Streetcar	0	Table	0.6
Bed	0	Couch	0	Bus	0
Table	0	Bed	0	Chair	0

Table 5. Performance on Deer class of CIFAR-10 dataset

AlexNet's Output	Prediction Accuracy (%)	GoogLeNet's Output	Prediction Accuracy (%)	ResNet50 Output	Prediction Accuracy (%)
Deer	87.6	Deer	49.5	Deer	75.4
Horse	3.7	Horse	24.4	Horse	10.7
Ship	3.4	Cat	13.3	Bird	3.5
Frog	2.2	Frog	6	Airplane	3.3
Truck	1.6	Bird	3	Dog	2.6
Airplane	1.2	Ship	2	Cat	2.5
Automobile	0.2	Airplane	1.1	Frog	1.6
Dog	0.1	Truck	0.3	Ship	0.3
Bird	0	Dog	0.4	Truck	0.1
Cat	0	Automobile	0	Automobile	0

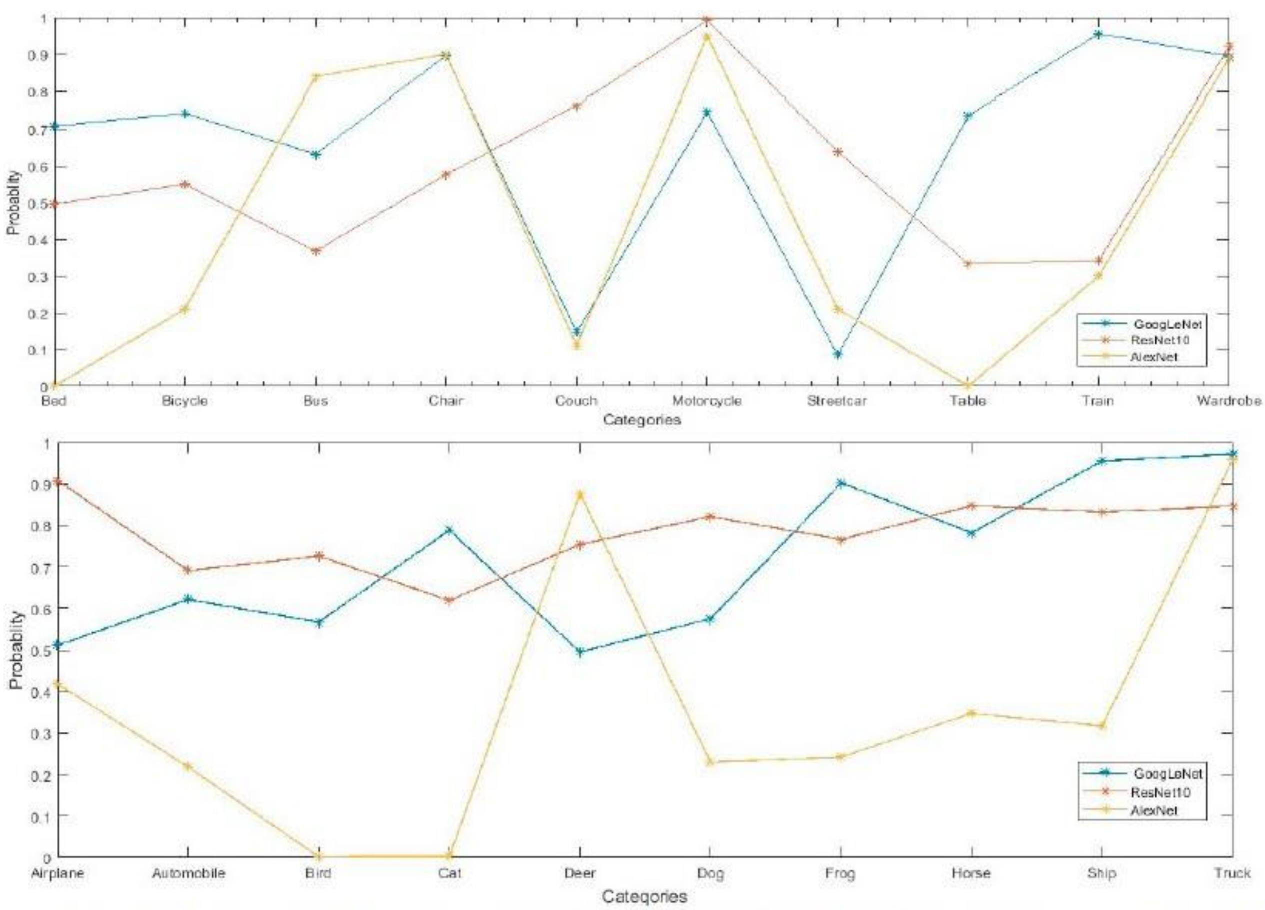


Fig 3: (a) Probability vs Categories graph for CIFAR- 100 dataset (b) Probability vs Categories graph for CIFAR- 10 dataset

Chapter 5. Implementation and Description of Project Modules

1. Importing the Libraries

In [2]:
```
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pylab import rcParams
import tensorflow as tf
import keras
%matplotlib inline
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam, SGD
from keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau
from keras.utils import to_categorical
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from skimage.transform import resize
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit
import seaborn as sns
import cv2
import albumentations as albu
```

```
Using TensorFlow backend.
```

2. Loading the CIFAR-100 Dataset

In [3]:
```
#function to open the files in the Python version of the dataset
def unpickle(file):
    with open(file, 'rb') as fo:
        myDict = pickle.load(fo, encoding='latin1')
    return myDict
```

In [4]:
```
trainData = unpickle('train')

#type of items in each file
for item in trainData:
    print(item, type(trainData[item]))
```

```
filenames <class 'list'>
batch_label <class 'str'>
fine_labels <class 'list'>
coarse_labels <class 'list'>
data <class 'numpy.ndarray'>
```

In [5]:
```
print(len(trainData['data']))
print(len(trainData['data'][0]))
```

```
50000
3072
```

There are 50000 images in the training dataset and each image is a 3 channel 32 *32 pixel image (32* 32 * 3 = 3072).

Output

3. Superclass and subclass

In [12]:
```
#storing coarse labels along with its number code in a dataframe
category = pd.DataFrame(metaData['coarse_label_names'], columns=['SuperClass'])
category
```

Out[12]:

	SuperClass
0	aquatic_mammals
1	fish
2	flowers
3	food_containers
4	fruit_and_vegetables
5	household_electrical_devices
6	household_furniture
7	insects
8	large_carnivores
9	large_man-made_outdoor_things
10	large_natural_outdoor_scenes

In [13]:
```
#storing fine labels along with its number code in a dataframe
subCategory = pd.DataFrame(metaData['fine_label_names'], columns=['SubClass'])
subCategory
```

Out[13]:

	SubClass
0	apple
1	aquarium_fish
2	baby
3	bear
4	beaver
...	...
95	whale
96	willow_tree
97	wolf
98	woman
99	worm

100 rows × 1 columns

4. Visualization

In [16]:
```
#generating a random number to display a random image from the dataset along with the label's number and name

rcParams['figure.figsize'] = 2,2

imageId = np.random.randint(0, len(X_train))

plt.imshow(X_train[imageId])

plt.axis('off')

print("Image number selected : {}".format(imageId))
print("Shape of image : {}".format(X_train[imageId].shape))
print("Image category number: {}".format(trainData['coarse_labels'][imageId]))
print("Image category name: {}".format(category.iloc[trainData['coarse_labels'][imageId]][0].capitalize()))
print("Image subcategory number: {}".format(trainData['fine_labels'][imageId]))
print("Image subcategory name: {}".format(subCategory.iloc[trainData['fine_labels'][imageId]][0].capitalize()))
```

```
Image number selected : 23147
Shape of image : (32, 32, 3)
Image category number: 3
Image category name: Food_containers
Image subcategory number: 28
Image subcategory name: Cup
```

5. Data Pre-processing

In [18]:
```
#transforming the testing dataset
X_test = testData['data']
X_test = X_test.reshape(len(X_test),3,32,32).transpose(0,2,3,1)
X_test.shape
```

Out[18]: (10000, 32, 32, 3)

In [19]:
```
y_train = trainData['fine_labels']
#y_train

y_test = testData['fine_labels']
#y_test
```

6. Converting class vectors to binary class vectors

In [20]:
```
n_classes = 100

y_train = to_categorical(y_train, n_classes)
#y_train

y_test = to_categorical(y_test, n_classes)
#y_test
```

7. Custom Data Generator Class

In [24]:
```
class DataGenerator(keras.utils.Sequence):
    def __init__(self, images, labels=None, mode='fit', batch_size=batch_size, dim=(height, width), channels=ch
annels, n_classes=n_classes, shuffle=True, augment=False):

        #initializing the configuration of the generator
        self.images = images
        self.labels = labels
        self.mode = mode
        self.batch_size = batch_size
        self.dim = dim
        self.channels = channels
        self.n_classes = n_classes
        self.shuffle = shuffle
        self.augment = augment
        self.on_epoch_end()

    #method to be called after every epoch
    def on_epoch_end(self):
        self.indexes = np.arange(self.images.shape[0])
        if self.shuffle == True:
            np.random.shuffle(self.indexes)

    #return numbers of steps in an epoch using samples and batch size
    def __len__(self):
        return int(np.floor(len(self.images) / self.batch_size))

    #this method is called with the batch number as an argument to obtain a given batch of data
```

7. Using Pre-trained EfficientNetB0

In [26]:
```
# !pip install -U efficientnet
```

In [27]:
```
import efficientnet.keras as efn

efnb0 = efn.EfficientNetB0(weights='imagenet', include_top=False, input_shape=input_shape, classes=n_classes)

model = Sequential()
model.add(efnb0)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b0 (Model)      (None, 7, 7, 1280)        4049564
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1280)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               128100
=================================================================
Total params: 4,177,664
Trainable params: 4,135,648
Non-trainable params: 42,016
```

8. Train our model with 15 Epochs

In [28]:
```
optimizer = Adam(lr=0.0001)

#early stopping to monitor the validation loss and avoid overfitting
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=10, restore_best_weights=True)

#reducing learning rate on plateau
rlrop = ReduceLROnPlateau(monitor='val_loss', mode='min', patience= 5, factor= 0.5, min_lr= 1e-6, verbose=1)
```

In [29]:
```
#model compiling
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

In [30]:
```
model_history = model.fit_generator(train_data_generator,
                                    validation_data=valid_data_generator,
                                    callbacks=[early_stop, rlrop],
                                    verbose=1,
                                    epochs=epochs)

#saving the trained model weights as data file in .h5 format
model.save_weights("cifar_efficientnetb0_weights.h5")
```

```
Epoch 1/15
5000/5000 [==============================] - 1598s 320ms/step - loss: 2.6624 - accuracy: 0.3432 - val_loss: 1.4
835 - val_accuracy: 0.6754
Epoch 2/15
5000/5000 [==============================] - 1558s 312ms/step - loss: 1.6425 - accuracy: 0.5491 - val_loss: 0.4
246 - val_accuracy: 0.7424
Epoch 3/15
5000/5000 [==============================] - 1561s 312ms/step - loss: 1.3483 - accuracy: 0.6242 - val_loss: 0.9
113 - val_accuracy: 0.7580
Epoch 4/15
```

9. Evaluating the model

Model Evaluation

In [33]:
```
valid_loss, valid_accuracy = model.evaluate_generator(generator=valid_d
ata_generator, verbose=1)

print('Validation Accuracy: ', round((valid_accuracy * 100), 2), "%")
```

```
1250/1250 [==============================] - 85s 68ms/step
Validation Accuracy:  82.3 %
```

10. Predicting the model

Model Prediction

In [34]:
```
y_pred = model.predict_generator(DataGenerator(X_test, mode='predict',
augment=False, shuffle=False), verbose=1)
y_pred = np.argmax(y_pred, axis=1)
test_accuracy = accuracy_score(np.argmax(y_test, axis=1), y_pred)
print('Test Accuracy: ', round((test_accuracy * 100), 2), "%")
```

```
1250/1250 [==============================] - 78s 63ms/step
Test Accuracy:  81.79 %
```

11. Confusion Matrix

Confusion Matrix

In [35]:
```
cm = confusion_matrix(np.argmax(y_test, axis=1), y_pred)
print(cm)
```

```
[[93  0  0 ...  0  0  0]
 [ 0 92  0 ...  0  0  0]
 [ 0  0 81 ...  0  0  0]
 ...
 [ 0  0  0 ... 88  0  0]
 [ 0  0  3 ...  0 66  0]
 [ 0  0  0 ...  0  1 84]]
```

In [36]:
```
#report to see which category has been predicted incorectly and which h
as been predicted correctly
target = ["Category {}".format(i) for i in range(n_classes)]
print(classification_report(np.argmax(y_test, axis=1), y_pred, target_n
ames=target))
```

```
              precision    recall  f1-score   support

  Category 0       0.89      0.93      0.91       100
  Category 1       0.88      0.92      0.90       100
  Category 2       0.60      0.81      0.69       100
  Category 3       0.75      0.82      0.78       100
  Category 4       0.72      0.71      0.71       100
  Category 5       0.80      0.78      0.79       100
```

12. Visualizing the Predictions

Visualizing the Predictions

```
In [37]: #dataframe of predictions
         prediction = pd.DataFrame(y_pred)
         prediction.head()
```

Out[37]:

	0
0	68
1	33
2	55
3	51
4	71

13. Model Testing

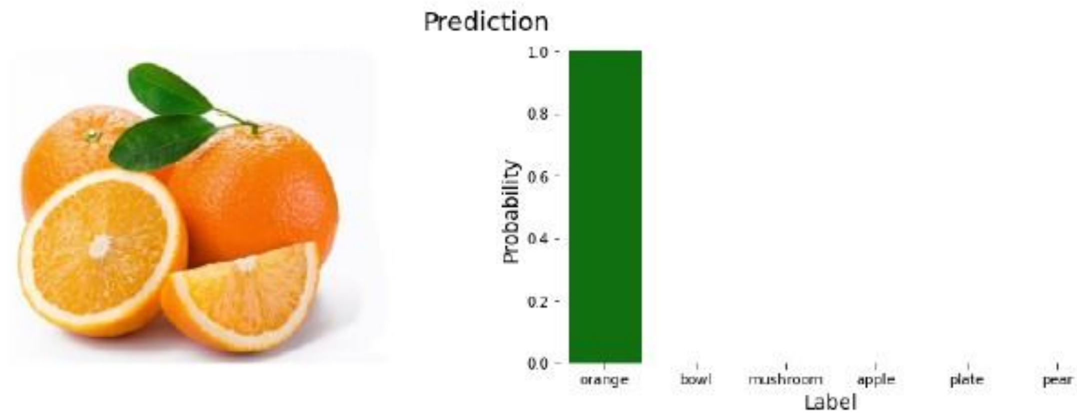
Testing the model

```
In [44]: #function to get the plot for top 5 predictions
         def plot_top5_prediction_test_image(test_img):

             fig, axes = plt.subplots(1, 2, figsize=(15,4))
             fig.suptitle("Prediction", fontsize=18)

             new_img = plt.imread(test_img)
             axes[0].imshow(new_img)
             axes[0].axis('off')

             data = df_top5_prediction_test_image(test_img)
             x=df_top5_prediction_test_image(test_img)['Label']
             y=df_top5_prediction_test_image(test_img)['Probability']

             axes[1] = sns.barplot(x=x, y=y, data=data, color="green")

             plt.xlabel('Label', fontsize=14)
             plt.ylabel('Probability', fontsize=14)

             plt.ylim(0,1.0)

             axes[1].grid(False)
             axes[1].spines["top"].set_visible(False)
             axes[1].spines["right"].set_visible(False)
             axes[1].spines["bottom"].set_visible(False)
             axes[1].spines["left"].set_visible(False)
```

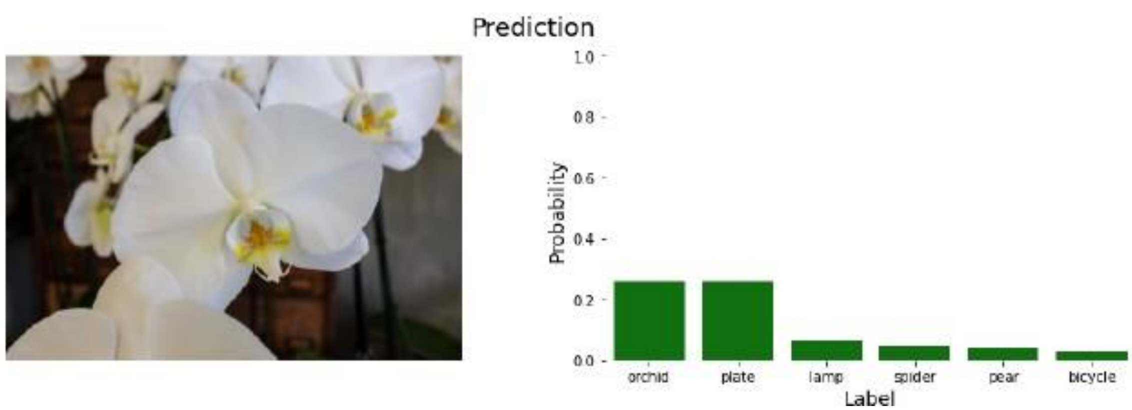
Chapter 4 :

Results and Discussion

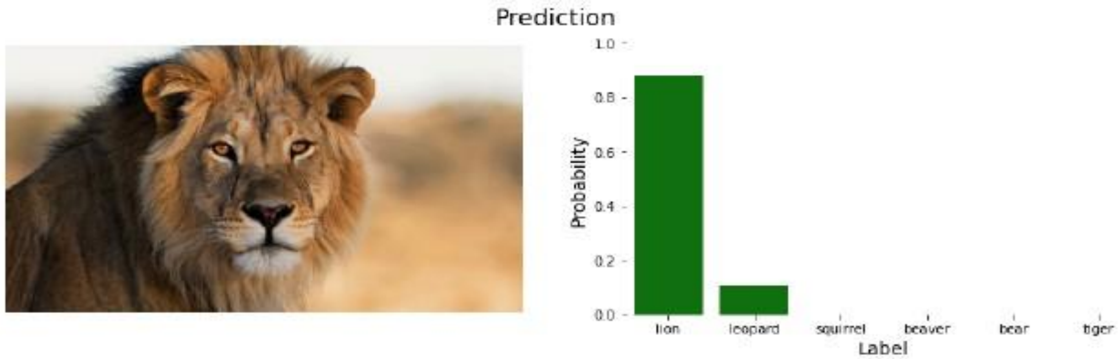
```
In [45]: plot_top5_prediction_test_image('orange.jpeg')
```



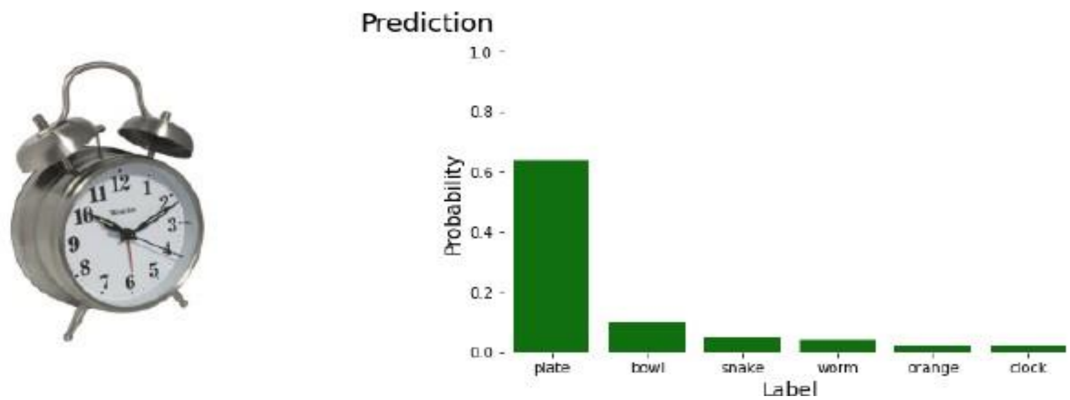

```
In [46]: plot_top5_prediction_test_image('Orchid.jpg')
```



```
In [49]: plot_top5_prediction_test_image('lion.jpg')
```

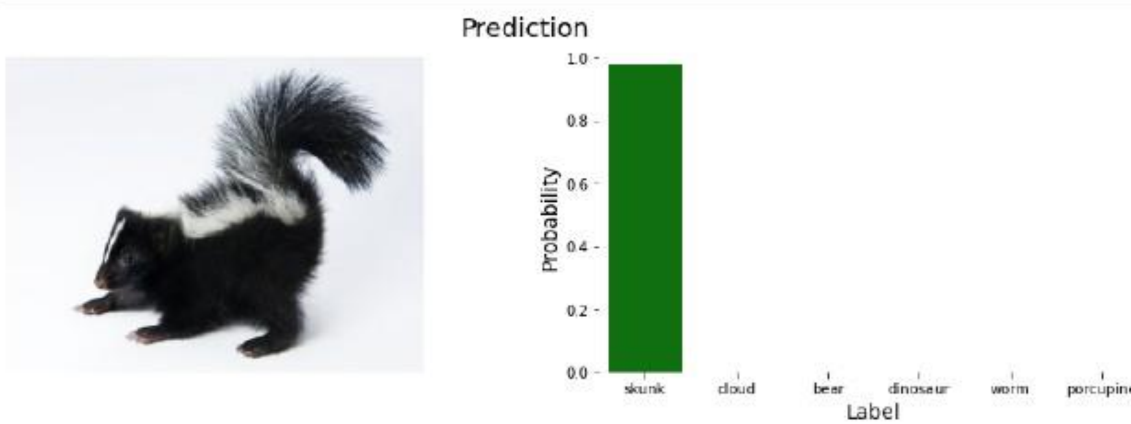



```
In [50]: plot_top5_prediction_test_image('clock.jpg')
```



14. Saving the Model

```
In [53]: plot_top5_prediction_test_image('skunk.jpg')
```




```
In [54]: #saving the trained model as data file in .h5 format
         model.save('cifar_efficientnetb0_model.h5')
```

Chapter 5.
Conclusion and Future Scope:

Conclusion:

Here we demonstrate a model which can recognize and classify the image. Later it can be extended for object recognition, character recognition, and real-time object recognition. Image recognition is an important step to the vast field of artificial intelligence and computer vision. As seen from the results of the experiment, CNN proves to be far better than other classifiers. The results can be made more accurate by increasing the number of convolution layers and hidden neurons. People can recognize the object from blurry images by using our model. Image recognition is an excellent prototype problem for learning about neural networks, and it gives a great way to develop more advanced techniques of deep learning. In the future, we are planning to develop a real-time image recognition system.

Future Scope:

Convolutional Neural Networks (CNNs) are the backbone of image classification, a deep learning phenomenon that takes an image and assigns it a class and a label that makes it unique. **DIGITAL IMAGE PROCESSING** forms a significant part of machine learning experiments.Together with using CNN and its induced capabilities, it is now widely used for a range of applications-right from Facebook picture tagging to Amazon product recommendations and healthcare imagery to automatic cars. The reason CNN is so popular is that it requires very little pre-processing, meaning that it can read 2D images by applying filters that other conventional algorithms cannot

References:

[1] *Barboza, Flavio, Herbert Kimura, and Edward Altman. "Machine learning models and bankruptcy prediction." Expert Systems with Applications 83 (2017): 405-417.*

[2] *Cramer, Sam, et al. "An extensive evaluation of seven machine learning methods for rainfall prediction in weather derivatives." Expert Systems with Applications 85 (2017): 169-181.*

[3] *Ling, Zhen-Hua, et al. "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends." IEEE Signal Processing Magazine 32.3 (2015): 35-52.*

[4] *Lillesand, T.M. and Kiefer, R.W. and Chipman, J.W., in "Remote Sensing and Image Interpretation" 5th ed. Wiley, 2004*

[5] *Li Deng and Dong Yu "Deep Learning: methods and applications" by Microsoft research [Online] available at: http://research.microsoft.com/pubs/209355/NOW-Book RevisedFeb2014-online.pdf*

[6] *McCulloch, Warren; Walter Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics 5 (4): 115–133(1943)*

[7] *An introduction to convolutional neural networks [Online]available at:http://white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_ Neural_ Networks*