**A PROJECT REPORT**

**on**

# “OBJECT RECOGNITION”

**Submitted to**

**KIIT Deemed to be University**

**In Partial Fulfilment of the Requirement for the Award of**

**BACHELOR’S DEGREE IN**

**COMPUTER SCIENCE ENGINEERING**

**BY**

**SIDDHARTH RANJAN** 1605233

**UNDER THE GUIDANCE OF**

**PROF. GUIDE NAME**

**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA - 751024**

**April 2020**

KIIT Deemed to be University

School of Computer Engineering

Bhubaneswar, ODISHA 751024

# CERTIFICATE

This is certify that the project entitled

“NAME OF PROJECT“

submitted by

| | |
|---|---|
| SIDDHARTH RANJAN | 1605233 |

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science

& Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2019-2020, under our guidance.

Date: / /

| | |
|---|---|
| (Prof. Co-Guide Name) | (Prof. Guide Name) |
| Project Co-Guide | Project Guide |

# Acknowledgements

We are profoundly grateful to Prof. SUJOY DATTA for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion. .....................

SIDDHARTH RANJAN

# ABSTRACT

Object identification and recognition is an area within the field of artificial intelligence (AI) that focuses on recognizing different objects. As AI machines become more and more part of our everyday lives, machine learning is making improvements on image tagging and object identification skills.

It Locate the presence of objects in an image and assign a label to that image.
Region-Based Convolutional Neural Networks, or R-CNNs, are a family of techniques for addressing object localization and recognition tasks, designed for model performance.
Project contains datasets of Kaggle's image classification.

Image classification is a dataset of Kaggle consisting of a training set of 1399 examples and a test set of 60,000 examples. Each example is a 300x300 colour image, associated with a label from 4 classes. We intend image classification data to serve as a direct drop-in replacement for the original dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

Python libraries used matplotlib, numpy, pandas, tenserflow, OS, Keras.
Recognises images based on data used in Kaggle's Image classification datasets.

Your abstract, paragraph 2.

**Keywords:**Write at least five keywords.

# Contents

**Chapter 4: Result and Analysis**

# List of Figures

# Chapter 1

# Introduction

## 1.1 Object Recognition

Object recognition refers to the task of inputting an image into a neural network and having it output some kind of label for that image. The label that the network outputs will correspond to a pre-defined class. There can be multiple classes that the image can be labeled as, or just one. If there is a single class, the term "recognition" is often applied, whereas a multi-class recognition task is often called "classification".

A subset of image classification is object detection, where specific instances of objects are identified as belonging to a certain class like animals, cars, or people

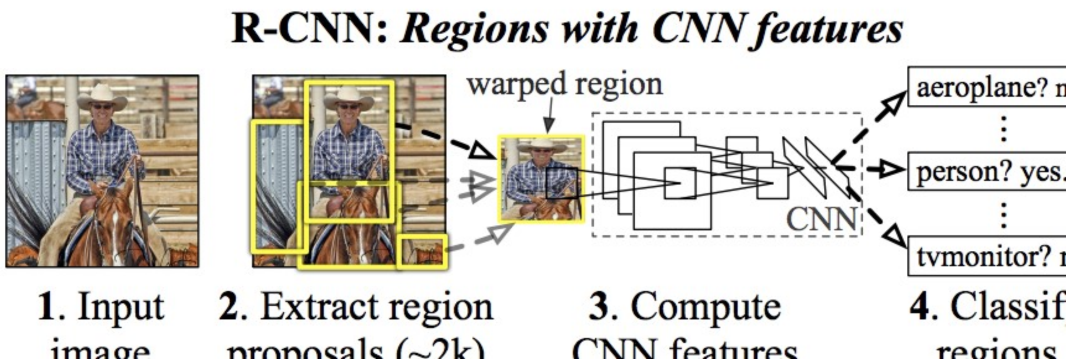


Fig : Regions with CNN feature

**Feature Extraction**

In order to carry out object  recognition/classification, the neural network must carry out feature extraction. Features are the elements of the data that you care about which will be fed through the network. In the specific case of image recognition, the features are the groups of pixels, like edges and points, of an object that the network will analyze for patterns.

Feature recognition (or feature extraction) is the process of pulling the relevant features out from an input image so that these features can be analyzed. Many images contain annotations or metadata about the image that helps the network find the relevant features.

## How Neural Networks Learn to Recognize Images

Getting an intuition of how a neural network recognizes images will help us when we are implementing a neural network model, so let's briefly explore the image recognition process in the next few sections.

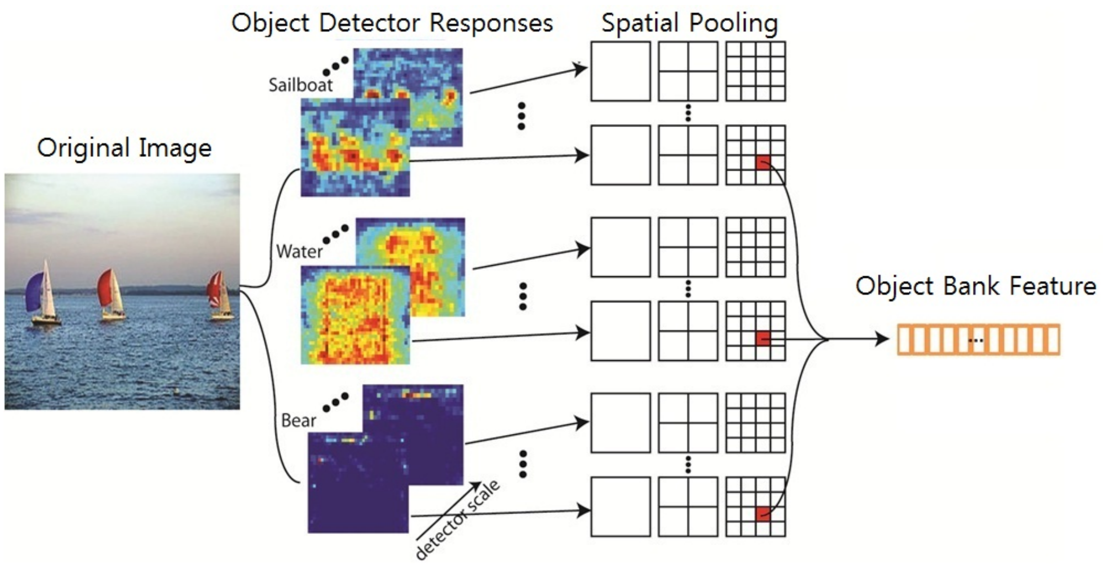


Figure number 1.1 Feature Extraction

The first layer of a neural network takes in all the pixels within an image. After all the data has been fed into the network, different filters are applied to the image, which forms representations of different parts of the image. This is feature extraction and it creates "feature maps".

This process of extracting features from an image is accomplished with a "convolutional layer", and convolution is simply forming a representation of part of an image. It is from this convolution concept that we get the term Convolutional Neural Network (CNN), the type of neural network most commonly used in image classification/recognition.

If we want to visualize how creating feature maps works, think about shining a flashlight over a picture in a dark room. As we slide the beam over the picture we are learning about features of the image. A filter is what the network uses to form a representation of the image, and in this metaphor, the light from the flashlight is the filter.

The width of our flashlight's beam controls how much of the image you examine at one time, and neural networks have a similar parameter, the filter size. Filter size affects how much of the image, how many pixels, are being examined at one time. A common filter size used in CNNs is 3, and this covers both height and width, so the filter examines a 3 x 3 area of pixels.

While the filter size covers the height and width of the filter, the filter's depth must also be specified.

Digital images are rendered as height, width, and some RGB value that defines the pixel's colors, so the "depth" that is being tracked is the number of color channels the image has. Grayscale (non-color) images only have 1 color channel while color images have 3 depth channels.

All of this means that for a filter of size 3 applied to a full-color image, the dimensions of that filter will be 3 x 3 x 3. For every pixel covered by that filter, the network multiplies the filter values with the values in the pixels themselves to get a numerical representation of that pixel. This process is then done for the entire image to achieve a complete representation. The filter is moved across the rest of the image according to a parameter called "stride", which defines how many pixels the filter is to be moved by after it calculates the value in its current position. A conventional stride size for a CNN is 2.

The end result of all this calculation is a feature map. This process is typically done with more than one filter, which helps preserve the complexity of the image.

**Activation Functions**

After the feature map of the image has been created, the values that represent the image are passed through an activation function or activation layer. The activation function takes values that represent the image, which are in a linear form (i.e. just a list of numbers) thanks to the convolutional layer, and increases their non-linearity since images themselves are non-linear.

The typical activation function used to accomplish this is a Rectified Linear Unit (ReLU), although there are some other activation functions that are occasionally used.

**Pooling Layers**

After the data is activated, it is sent through a pooling layer. Pooling "downsamples" an image, meaning that it takes the information which represents the image and compresses it, making it smaller. The pooling process makes the network more flexible and more adept at recognizing objects/images based on the relevant features.

When we look at an image, we typically aren't concerned with all the information in the background of the image, only the features we care about, such as people or animals.

Similarly, a pooling layer in a CNN will abstract away the unnecessary parts of the image, keeping only the parts of the image it thinks are relevant, as controlled by the specified size of the pooling layer.

Because it has to make decisions about the most relevant parts of the image, the hope is that the network will learn only the parts of the image that truly represent the object in question. This helps prevent overfitting, where the network learns aspects of the training case too well and fails to generalize to new data.

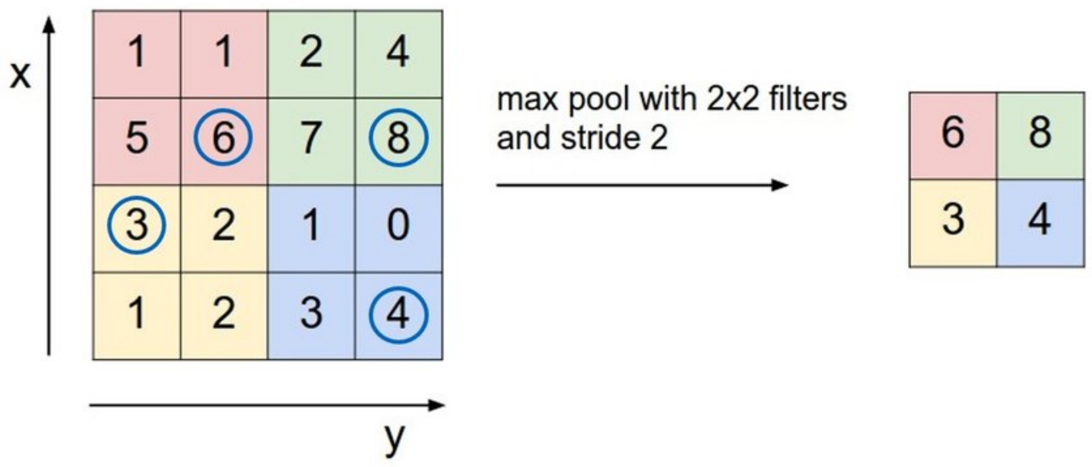


Figure number 1.3 depth slice

There are various ways to pool values, but max pooling is most commonly used. Max pooling obtains the maximum value of the pixels within a single filter (within a single spot in the image). This drops 3/4ths of information, assuming 2 x 2 filters are being used.

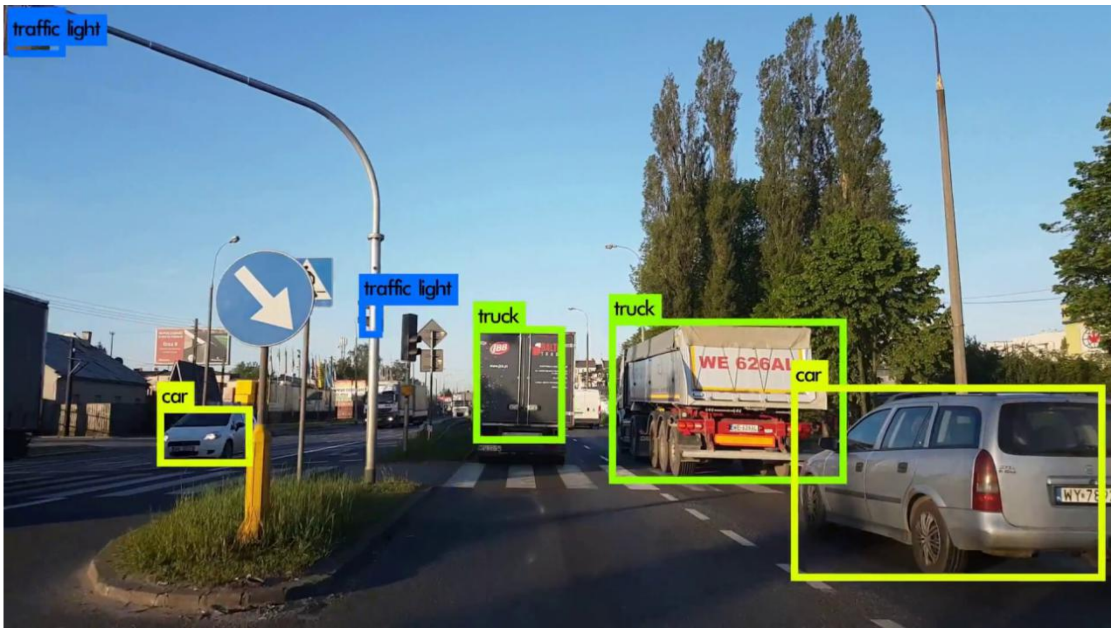


Figure Understanding Object detection model

The maximum values of the pixels are used in order to account for possible image distortions, and the parameters/size of the image are reduced in order to control for overfitting. There are other pooling types such as average pooling or sum pooling, but these aren't used as frequently because max pooling tends to yield better accuracy.

**Flattening**

The final layers of our CNN, the densely connected layers, require that the data is in the form of a vector to be processed. For this reason, the data must be "flattened". The values are compressed into a long vector or a column of sequentially ordered numbers.

**Fully Connected Layer**

The final layers of the CNN are densely connected layers, or an artificial neural network (ANN). The primary function of the ANN is to analyze the input features and combine them into different attributes that will assist in classification. These layers are essentially forming collections of neurons that represent different parts of the object in question, and a collection of neurons may represent the floppy ears of a dog or the redness of an apple. When enough of these neurons are activated in response to an input image, the image will be classified as an object.

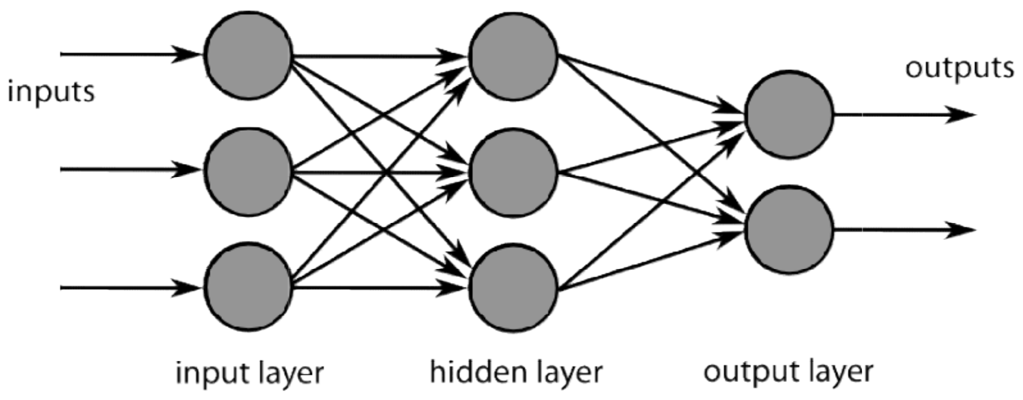


Figure number 1.4 Deep learning layers

The error, or the difference between the computed values and the expected value in the training set, is calculated by the ANN. The network then undergoes backpropagation, where the influence of a given neuron on a neuron in the next layer is calculated and its influence adjusted. This is done to optimize the performance of the model. This process is then repeated over and over. This is how the network trains on data and learns associations between input features and output classes.

The neurons in the middle fully connected layers will output binary values relating to the possible classes. If you have four different classes (let's say a dog, a car, a house, and a person), the neuron will have a "1" value for the class it believes the image represents and a "0" value for the other classes.

The final fully connected layer will receive the output of the layer before it and deliver a probability for each of the classes, summing to one. If there is a 0.75 value in the "dog" category, it represents a 75% certainty that the image is a dog.

The image classifier has now been trained, and images can be passed into the CNN, which will now output a guess about the content of that image.

## 1.2 The Machine Learning Workflow

Before we jump into an example of training an image classifier, let's take a moment to understand the machine learning workflow or pipeline. The process for training a neural network model is fairly standard and can be broken down into four different phases.

### Data Preparation

First, you will need to collect your data and put it in a form the network can train on. This involves collecting images and labeling them. Even if you have downloaded a data set someone else has prepared, there is likely to be preprocessing or preparation that you must do before you can use it for training. Data preparation is an art all on its own, involving dealing with things like missing values, corrupted data, data in the wrong format, incorrect labels, etc.

In this project, we will be using a preprocessed data set.

### Creating the Model

Creating the neural network model involves making choices about various parameters and hyperparameters. You must make decisions about the number of layers to use in your model, what the input and output sizes of the layers will be, what kind of activation functions you will use, whether or not you will use dropout, etc.

Learning which parameters and hyperparameters to use will come with time (and a lot of studying), but right out of the gate there are some heuristics you can use to get you running and we'll cover some of these during the implementation example.

**Training the Model**

After you have created your model, you simply create an instance of the model and fit it with your training data. The biggest consideration when training a model is the amount of time the model takes to train. You can specify the length of training for a network by specifying the number of epochs to train over. The longer you train a model, the greater its performance will improve, but too many training epochs and you risk overfitting.

Choosing the number of epochs to train for is something you will get a feel for, and it is customary to save the weights of a network in between training sessions so that you need not start over once you have made some progress training the network.

**Model Evaluation**

There are multiple steps to evaluating the model. The first step in evaluating the model is comparing the model's performance against a validation dataset, a data set that the model hasn't been trained on. You will compare the model's performance against this validation set and analyze its performance through different metrics.

There are various metrics for determining the performance of a neural network model, but the most common metric is "accuracy", the amount of correctly classified images divided by the total number of images in your data set.

After you have seen the accuracy of the model's performance on a validation dataset, you will typically go back and train the network again using slightly tweaked parameters, because it's unlikely you will be satisfied with your network's performance the first time you train. You will keep tweaking the parameters of your network, retraining it, and measuring its performance until you are satisfied with the network's accuracy.

Finally, you will test the network's performance on a testing set. This testing set is another set of data your model has never seen before.

It's a good idea to keep a batch of data the network has never seen for testing because all the tweaking of the parameters you do, combined with the retesting on the validation set, could mean that your network has learned some idiosyncrasies of the validation set which will not generalize to out-of-sample data.

Therefore, the purpose of the testing set is to check for issues like over fitting and be more confident that your model is truly fit to perform in the real world.

# Chapter 2

## 2.1Python 3.0

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is named after a TV Show called ëMonty Pythonís Flying Circusí and not after Python-the snake.

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later on many of its important features have been backported to be compatible with version 2.7.This tutorial gives enough understanding on Python 3 version programming language. Please refer to this link for our Python 2 tutorial.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python 3.0 (a.k.a. "Python 3000" or "Py3k") is a new version of the language that is incompatible with the 2.x line of releases. The language is mostly the same, but many details, especially how built-in objects like dictionaries and strings work, have changed considerably, and a lot of deprecated features have finally been removed. Also, the standard library has been reorganized in a few prominent places.

## 2.2 TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name
TensorFlow derives from the operations that such neural networks 2

perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In Jan 2018, Google announced TensorFlow 2.0. In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript and TensorFlow Graphics for deep learning in computer graphics.

## 2.3 IPython Notebook

The IPython Notebook is now known as the Jupyter Notebook. It is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media.

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

# Chapter 3

## 3.1 Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

**Key Features of Pandas**

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

## 3.2 Numpy

Numpy can be abbreviated as Numeric Python, is a Data analysis library for Python that consists of multi-dimensional array-objects as well as a collection of routines to process these arrays. In this tutorial, you will be learning about the various uses of this library concerning data science.NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be views into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK). NumPy has built-in support for memory-mapped ndarrays.

## 3.3 Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like atter, histogram etc. Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They're typically instruments for reasoning about quantitative information.Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter

## 3.4 scikit-learn

Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

**Components of scikit-learn:**

- **Supervised learning algorithms**: Think of any supervised learning algorithm you might have heard about and there is a very high chance that it is part of scikit-learn. Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox. The spread of algorithms is one of the big reasons for high usage of scikit-learn. I started using scikit to solve supervised learning problems and would recommend that to people new to scikit / machine learning as well.
- **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data**.**
- **Unsupervised learning algorithms:** Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
- **Various toy datasets:** This came in handy while learning scikit-learn. I had learnt SAS using various academic datasets (e.g. IRIS dataset, Boston House prices dataset). Having them handy while learning a new library helped a lot.
- **Feature extraction:** Useful for extracting features from images and text (e.g. Bag of words)

## 3.5SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing.

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms, and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multidimensional container of data with arbitrary datatypes. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

# Chapter 4

## RESULT AND ANALYSIS

### 4.1 Loading The Dataset into Model

First we load the dataset from the system to model, by using the OS module & All the images are read by cv2 library.

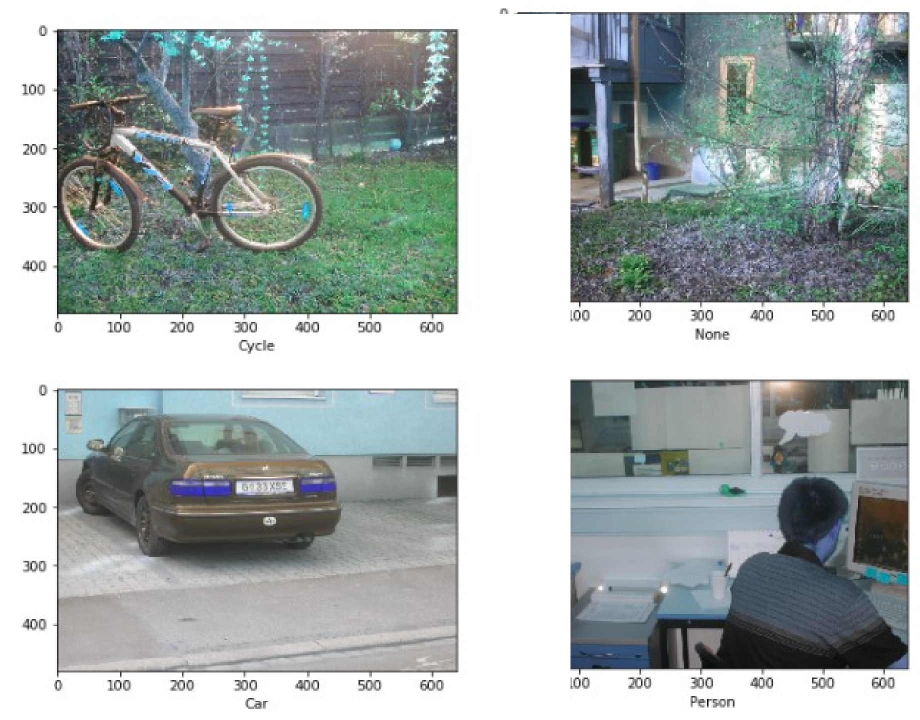


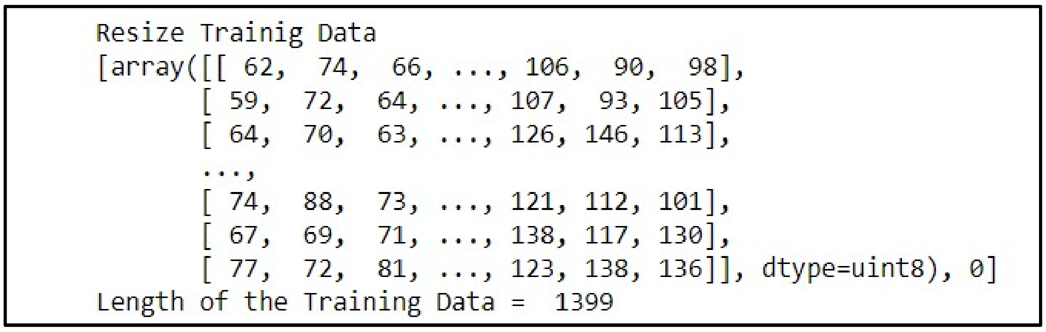
Figure 4.1: Data in image form

### 4.2 Resize the Dataset

we've ensured that all images are 640x480 pixels, it's time to scale each image appropriately. We've decided to have images with width and height of 300 pixels.

Figure 4.2:Data in Pixel form

Now it is observed that the Second column is the label data and because it has 4 classes so it is going to have from 0 to 3.The remaining columns are the actual pixel.

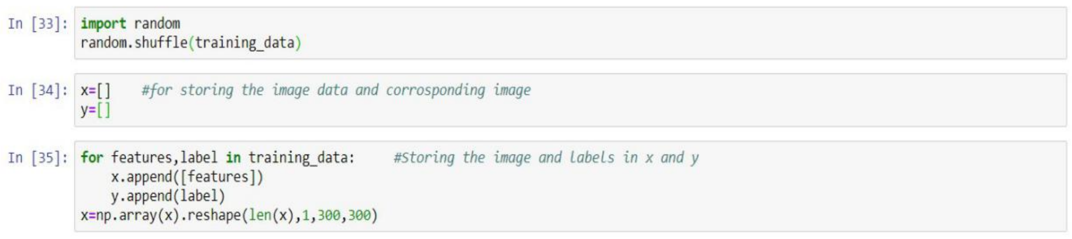


```
Resize Trainig Data
[array([[ 62,  74,  66, ..., 106,  90,  98],
       [ 59,  72,  64, ..., 107,  93, 105],
       [ 64,  70,  63, ..., 126, 146, 113],
       ...,
       [ 74,  88,  73, ..., 121, 112, 101],
       [ 67,  69,  71, ..., 138, 117, 130],
       [ 77,  72,  81, ..., 123, 138, 136]], dtype=uint8), 0]
Length of the Training Data =  1399
```

Now it is observed that the Second column is the label data and because it has 4 classes so it is going to have from 0 to 3.The remaining columns are the actual pixel data, Here as you can see there are about 300 columns that contain pixel data. Here each row is a different image representation in the form pixel data.

Now let us split the train data into x and y arrays where x represents the image data and y represents the labels. To do that we need to convert the data frames into NumPy arrays of float32 type which is the acceptable form for TensorFlow.

Now let us slice the train arrays into x and y arrays namely x_train, y_train to store all image data and label data respectively. i.e x_train contains all the rows and all columns except the label column and excluding header info .y_train contains all the rows and first column and excluding header info .Similarly slice the test arrays into x and y arrays namely x_train, y_train to store all image data and label data respectively. i.e x_test contains all the rows and all columns except the label column and excluding header info. y_test contains all the rows and first column and excluding header info .



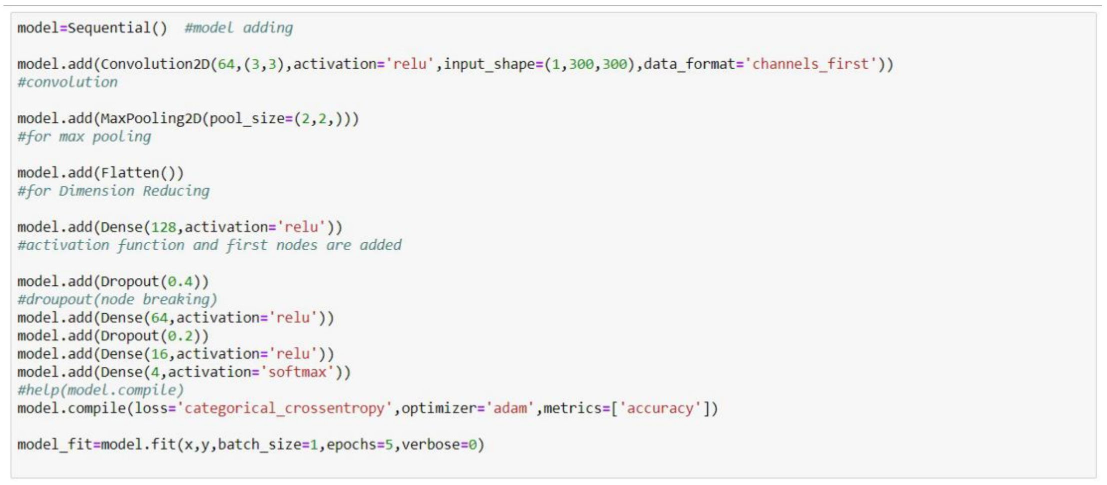
```
In [33]: import random
         random.shuffle(training_data)

In [34]: x=[]    #for storing the image data and corrosponding image
         y=[]

In [35]: for features,label in training_data:     #Storing the image and labels in x and y
             x.append([features])
             y.append(label)
         x=np.array(x).reshape(len(x),1,300,300)
```

Figure 4.3: splitting of data

## 4.3 Create the Convolutional Neural Networks (CNN)

- Define the model
- Compile the model
- Fit the model

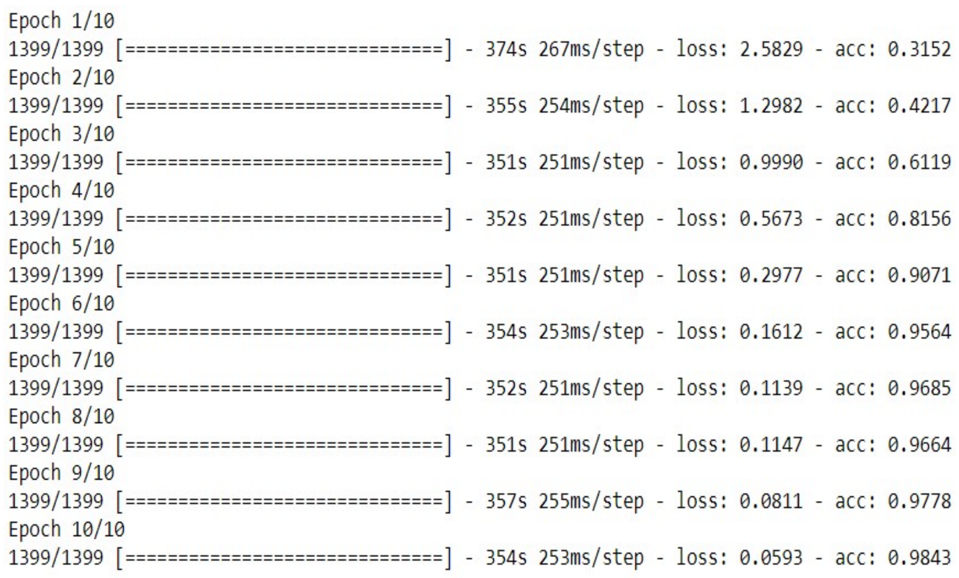
**Define the model**



```
model=Sequential()  #model adding

model.add(Convolution2D(64,(3,3),activation='relu',input_shape=(1,300,300),data_format='channels_first'))
#convolution

model.add(MaxPooling2D(pool_size=(2,2,)))
#for max pooling

model.add(Flatten())
#for Dimension Reducing

model.add(Dense(128,activation='relu'))
#activation function and first nodes are added

model.add(Dropout(0.4))
#droupout(node breaking)
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(16,activation='relu'))
model.add(Dense(4,activation='softmax'))
#help(model.compile)
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

model_fit=model.fit(x,y,batch_size=1,epochs=5,verbose=0)
```

Figure 4.4:Defining the model

**Compile the model**



```
Epoch 1/10
1399/1399 [==============================] - 374s 267ms/step - loss: 2.5829 - acc: 0.3152
Epoch 2/10
1399/1399 [==============================] - 355s 254ms/step - loss: 1.2982 - acc: 0.4217
Epoch 3/10
1399/1399 [==============================] - 351s 251ms/step - loss: 0.9990 - acc: 0.6119
Epoch 4/10
1399/1399 [==============================] - 352s 251ms/step - loss: 0.5673 - acc: 0.8156
Epoch 5/10
1399/1399 [==============================] - 351s 251ms/step - loss: 0.2977 - acc: 0.9071
Epoch 6/10
1399/1399 [==============================] - 354s 253ms/step - loss: 0.1612 - acc: 0.9564
Epoch 7/10
1399/1399 [==============================] - 352s 251ms/step - loss: 0.1139 - acc: 0.9685
Epoch 8/10
1399/1399 [==============================] - 351s 251ms/step - loss: 0.1147 - acc: 0.9664
Epoch 9/10
1399/1399 [==============================] - 357s 255ms/step - loss: 0.0811 - acc: 0.9778
Epoch 10/10
1399/1399 [==============================] - 354s 253ms/step - loss: 0.0593 - acc: 0.9843
```

Figure 4.5:Compile the model

Let's plot training and validation accuracy as well as loss.

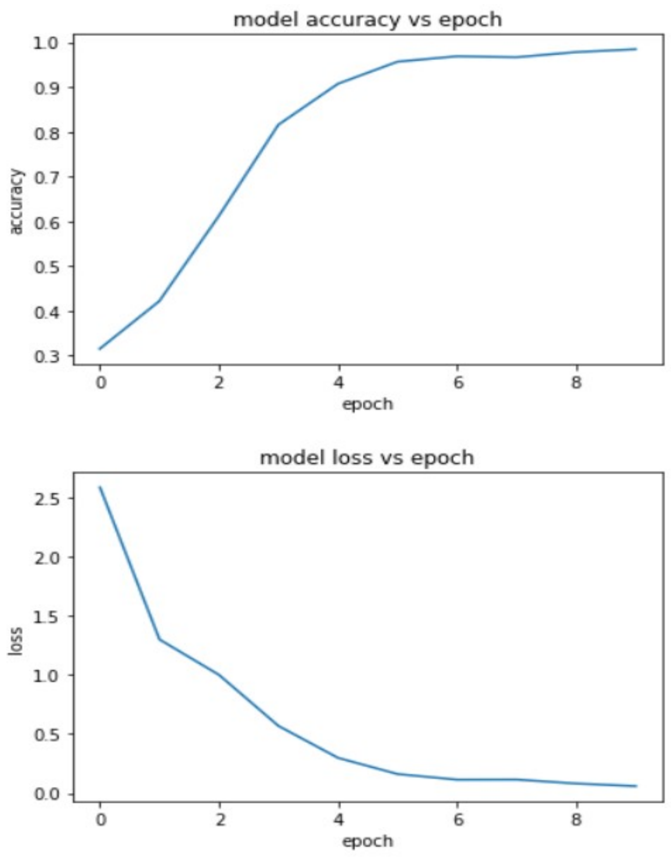
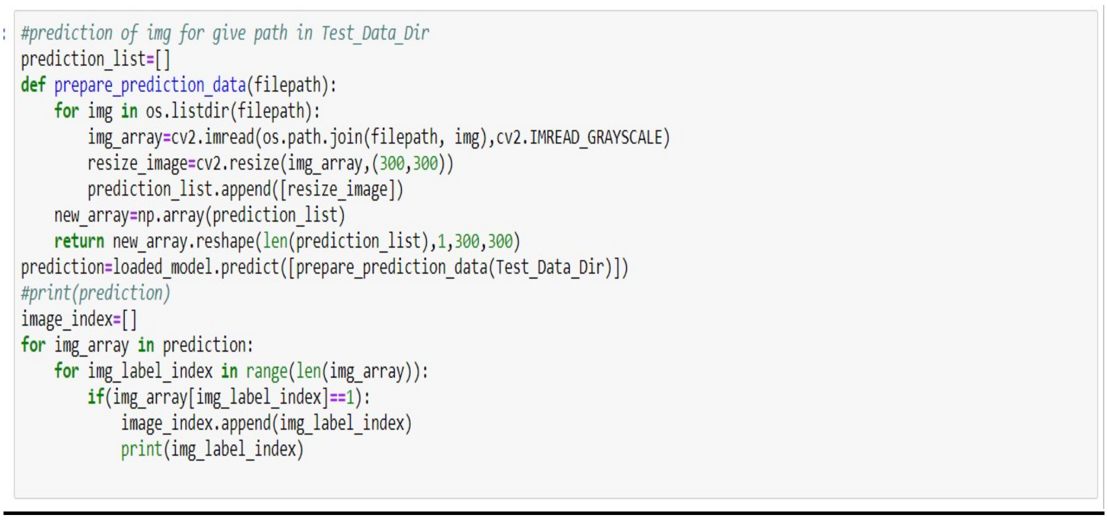


Figure 4.6:Graph of accuracy and loss

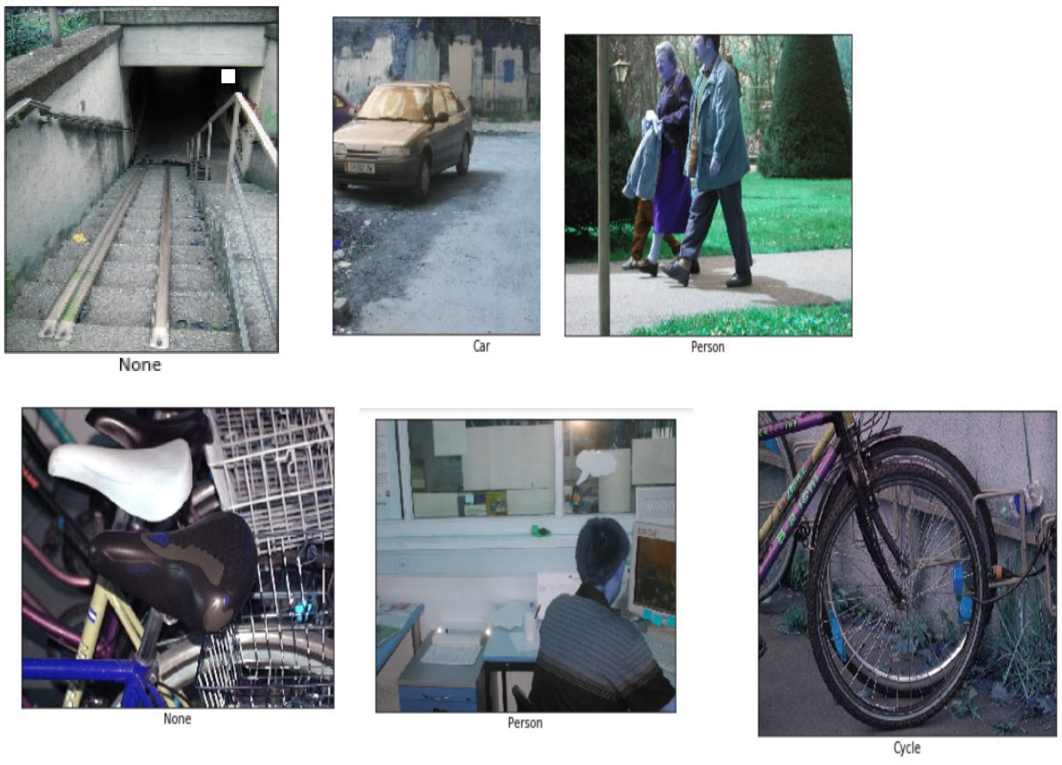
## Fit the Model



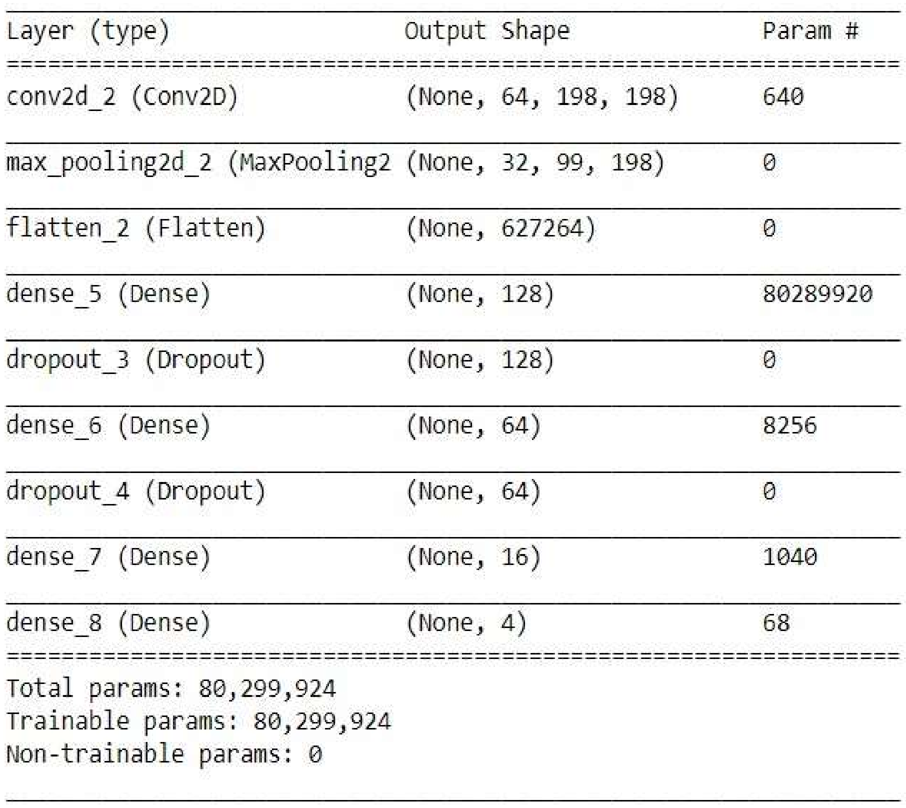
```
#prediction of img for give path in Test_Data_Dir
prediction_list=[]
def prepare_prediction_data(filepath):
    for img in os.listdir(filepath):
        img_array=cv2.imread(os.path.join(filepath, img),cv2.IMREAD_GRAYSCALE)
        resize_image=cv2.resize(img_array,(300,300))
        prediction_list.append([resize_image])
    new_array=np.array(prediction_list)
    return new_array.reshape(len(prediction_list),1,300,300)
prediction=loaded_model.predict([prepare_prediction_data(Test_Data_Dir)])
#print(prediction)
image_index=[]
for img_array in prediction:
    for img_label_index in range(len(img_array)):
        if(img_array[img_label_index]==1):
            image_index.append(img_label_index)
            print(img_label_index)
```

Figure 4.7:model fited on testing Data

## 4.4 Displaying



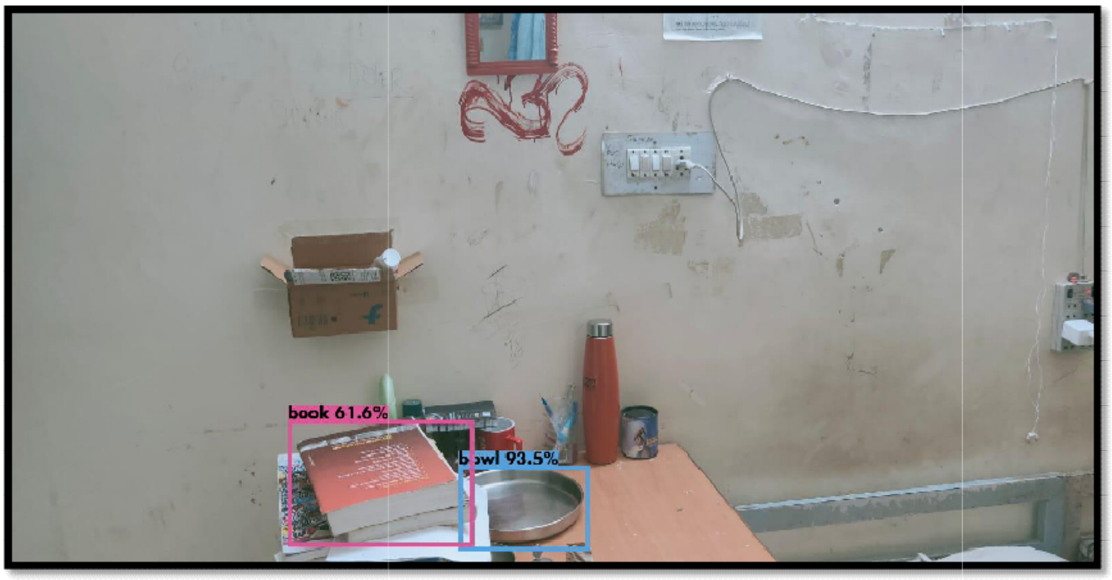
## 4.5 Model Summary



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 64, 198, 198)      640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 99, 198)       0
_________________________________________________________________
flatten_2 (Flatten)          (None, 627264)            0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               80289920
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_7 (Dense)              (None, 16)                1040
_________________________________________________________________
dense_8 (Dense)              (None, 4)                 68
=================================================================
Total params: 80,299,924
Trainable params: 80,299,924
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.7:Model Summery

## 4.6 Detection Images



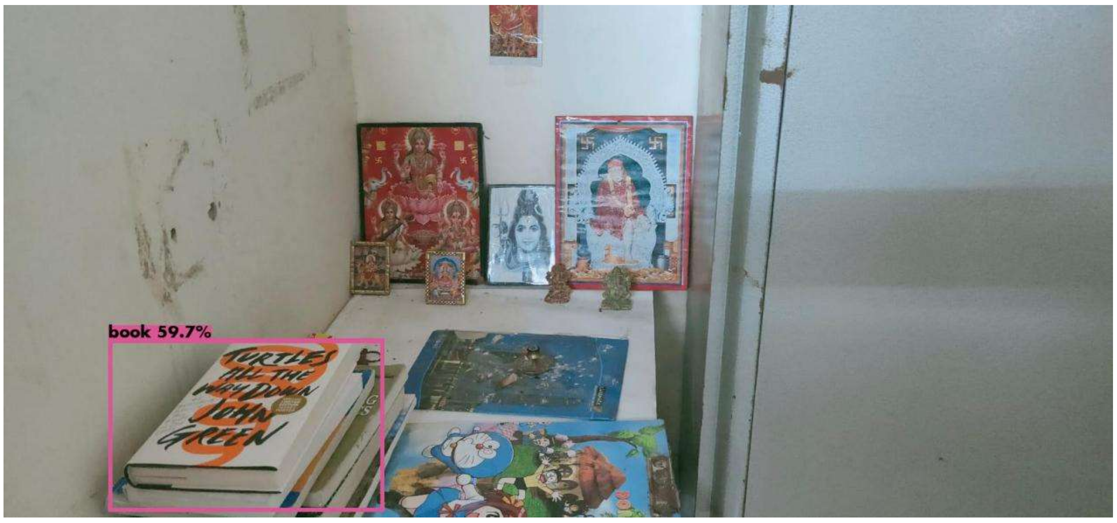
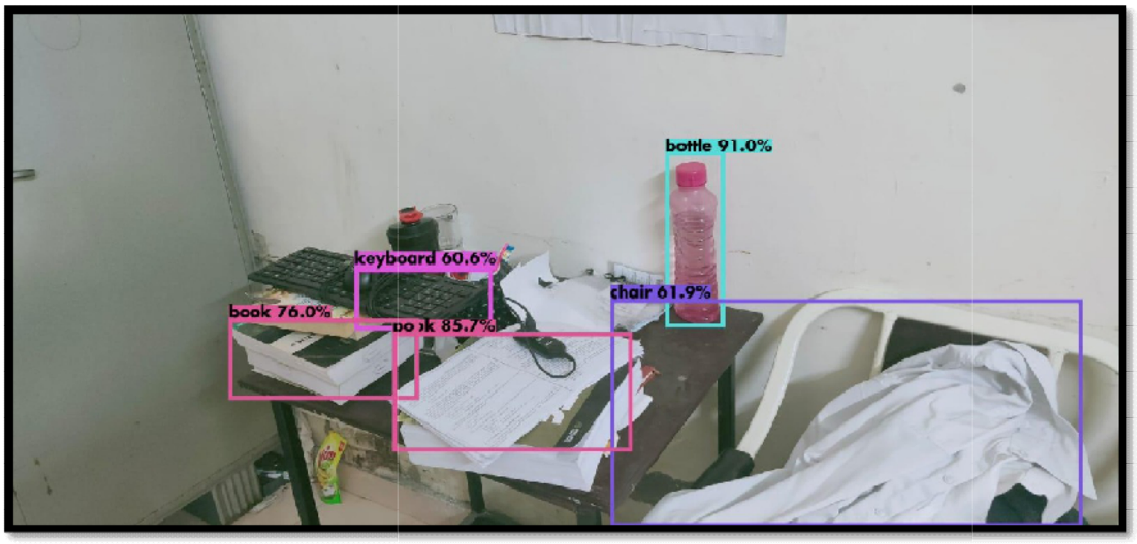


Figure : Detection 2

Figure : Detection 3



# Chapter 10

# Conclusion and Future Scope

**Object Classification:**

Object classification uses Object Classification datasets and on the basis training set of 1399 examples and a test set of 60,000 examples, build a model of neural networks that classifies the image feed in the model. The more it contain the training data the more accurate the model will become. The Dataset contain the image of t-shirts, trousers, pants, pullover, sweaters, Jeans, court, sandals, shirts, sneakers, bags and boot etc stored with a 640x480 image. By visualising the datasets a user can get more insight of the data.

One can learn with the datasets.

- Splitting the datasets into training and testing set.
- Making model with the dataset.
- Insight of data.
- Visualisation of datasets.
- Finding accuracy of the model.

By applying the above operation on datasets a user can easily learn to get insight of any datasets.

## FUTURE SCOPE

Object classification is a key ability for most computer and robot vision system. Although great progress has been observed in the last years, and some existing techniques are now part of many consumer electronics (e.g., face detection for auto-focus in smartphones) or have been integrated in assistant driving technologies, we are still far from achieving human-level performance, in particular in terms of open-world learning. It should be noted that object detection has not been used much in many areas where it could be of great help. As mobile robots, and in general autonomous machines, are starting to be more widely deployed (e.g., quad-copters, drones and soon service robots).

The need of object detection and classification systems is gaining more importance. Finally, we need to consider that we will need object detection systems for nano-robots or for robots that will explore areas that have not been seen by humans, such as depth parts of the sea or other planets, and the detection systems will have to learn to new object classes as they are encountered. In such cases, a real-time open-world learning ability will be critical.

# References

**All references should be numbered and cited in the content.**

[1] https://en.wikipedia.org/wiki/Python_

[2] https://en.wikipedia.org/wiki/TensorFlow

[3] https://searchitoperations.techtarget.com/definition/GitHub

[4]https://en.wikipedia.org/wiki/object classification

[5] https://github.com/Rahul-Kumar-Tiwari/Artificial-Intelligence/

[6]https://en.wikipedia.org/wiki/IPython Notebook

[7]https://en.wikipedia.org/wiki/Data Science

[8] https://ict.iitk.ac.in/courses/artificial-intelligence-2/

## INDIVIDUAL CONTRIBUTION REPORT:

**<Object recognition>**

<siddharth ranjan>

<1605233>

**Abstract:** Object Detection is used almost everywhere these days.The use cases are endless,be it tracking objects,videosurveillence, pedestrian detection,anomaly detection,people counting,self-driving cars or face detection,the list goes on.

**Individual contribution and findings:** i have done this with the gudiance of my teacher sujoy datta.

**Individual contribution to project report preparation:** I have used some python library and python language.

**Individual contribution for project presentation and demonstration:** I have prepared a faster way to find object recognition.

Full Signature of Supervisor: Full signature of the student:

……………………………  ……………………………..