CLASSIFICATION OF HOME OBJECTS

A Main Project submitted
in partial fulfillment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

1. D. Hema Sri (16PA1A0540)

2. Ch. Annaji Rao (16PA1A0530)

3. Ch. Syam Babu (16PA1A0533)

4. A. Akhilash Varma (16PA1A0509)

5. A. Mani Shankar Sai (16PA1A0503)

Under the esteemed guidance of
U. S. S. P. Jyothi
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VISHNU INSTITUTE OF TECHNOLOGY (Autonomous)
(Approved by AICTE, Accredited by NBA & NAAC and permanently affiliated to JNTU Kakinada)
BHIMAVARAM – 534 202

2019 – 2020

VISHNU INSTITUTE OF TECHNOLOGY

(Autonomous)

(Approved by AICTE, Accredited by NBA & NAAC and permanently affiliated to JNTU Kakinada)

BHIMAVARAM-534202

2019-2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "CLASSIFICATION OF HOME OBJECTS", is being submitted by ***D. HEMASRI, CH. ANNAJIRAO, CH. SYAM BABU, A. AKHILASH VARMA and A. MANI SHANKAR SAI*** bearing the **REGD.NOS: 16PA1A0540, 16PA1A0530, 16PA1A0533, 16PA1A0509 and 16PA1A0503** submitted in fulfilment for the award of the degree of **"BACHELOR OF TECHNOLOGY"** in **"COMPUTER SCIENCE AND ENGINEERING"** is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2019-2020 and it has been found worthy of acceptance according to the requirements of university.

Internal Guide
U. S. S. P. Jyothi

Head of the Department
G. Sri Lakshmi

External Examiner

ACKNOWLEDGEMENT

It is nature and inevitable that the thoughts and ideas of other people tend to drift in to the subconscious due to various human parameters, where one feels acknowledge the help and guidance derived from others. We acknowledge each of those who have contributed for the fulfilment of this project.

We take the opportunity to express our sincere gratitude to **Dr.D.Suryanaryana,** director and principal, VIT**,** Bhimavaram whose guidance from time to time helped us to complete this project successfully.

We are very much thankful to **Dr. Sumit Gupta**, Head of the Department, Department of Computer Science and Engineering for his continuous and unrelenting support and guidance. We thank and acknowledge our gratitude to her for his valuable guidance and support expended to us right from the conception of the idea to the completion of this project.

We are very much thankful to **U. S. S. P. Jyothi**, Assistant Professor, our internal guide whose guidance from time to time helped us to complete this project successfully.

Project Associates

D. Hema Sri	**(16PA1A0540)**
Ch. Annajirao	**(16PA5A0530)**
Ch. Syam Babu	**(16PA1A0533)**
A. Akhilash Varma	**(16PA1A0532)**
A. Mani Shankar Sai	**(16PA1A0503)**

ABSTRACT

This project is to detect the name of the object. As we already had an Google application called Google lens, when an image is uploaded in this app, then it displays similar images but it doesn't give the name of an image. So, this project takes care of that feature. For example, if an unknown object is given and we will give the name of that object. Basically, we are working with 505 objects like different models of a cup, scissors, Binocular, Camera, Chair, Ceiling-fan, Headphone, Lamp, Umbrella, Wrench. The dataset consists of jpg files and they are of different home objects. Walking into the directory of dataset and making count of every image in each class. As data consists of different sized images, resized all the images into the same size. So, each image is resized into 300 X 300 pixels. Plotting the count of images of every class in a graph to show how many images were there in each class. Splitting data as train data and test data with 75% and 25% respectively. Sequential Model is used to train the data. As the images are of 2D shape. So, we added Conv2D layers. MaxPooling is used to reduce spatial dimensions of the output volume. Activation parameter used to keep the code cleaner. Then to test, an image of a camera is uploaded. It is sent to the predict function of a trained CNN model. As we predict the class label of the image the result is a "camera".

TABLE OF CONTENTS

INTRODUCTION

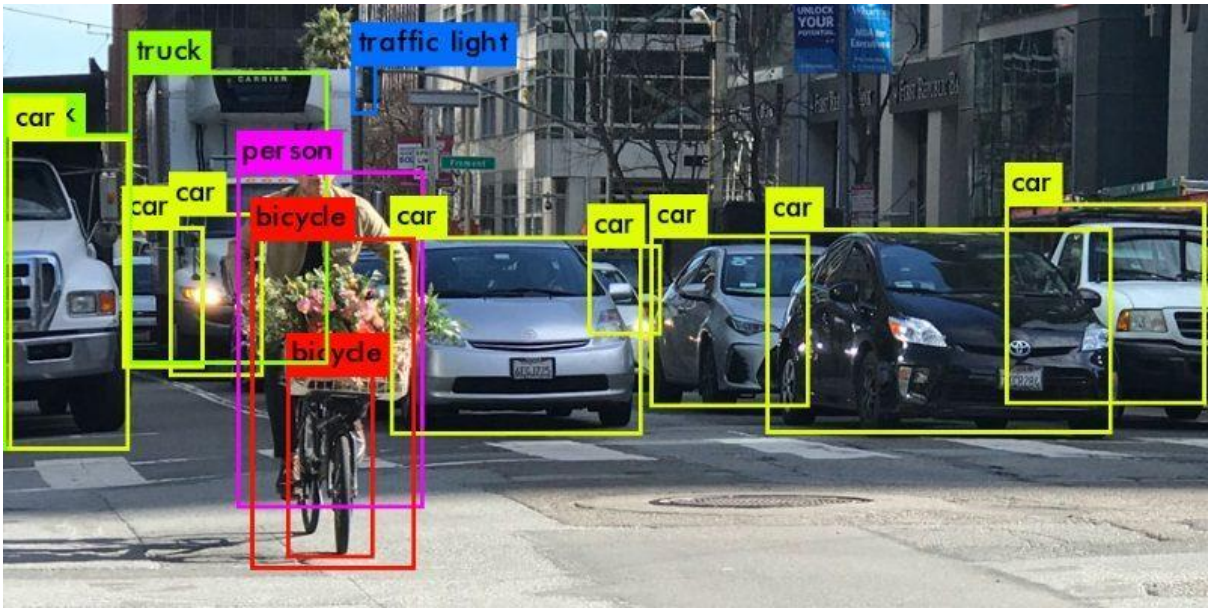
1. INTRODUCTION

1.1 INTRODUCTION:

Artificial Intelligence is one of the most fascinating and controversial technologies in the modern world. Some people are afraid of the consequences. Others can't wait to see AI-powered machines. And still, others are skeptical about them thinking that AI will never exceed the capability of human intelligence.

This way or another, developers keep working on improving machine learning solutions, and Artificial Intelligence gets more and more advanced. But there is one major issue – despite evolution, AI still seems to struggle when it comes to rendering images. That's why Image Detection, Classification, and Recognition are hot topics in the developer's world.

These three branches might seem similar. Although each of them has one goal – improving AI's abilities to understand visual content – they are different fields of Machine Learning. So, if you look closer at each branch, you'll see that there are some critical differences. But, of course, all three branches should merge to ensure that Artificial Intelligence can actually understand visual content.



1.2 WHAT IS IMAGE DETECTION?

Image or Object Detection is a computer technology that processes the image and detects objects in it. People often confuse Image Detection with Image Classification. Although the difference is rather clear. If you need to classify image items, you use Classification. But if you just need to locate them, for example, find out the number of objects in the picture, you should use Image Detection.

Let us give you an example. Think of how you're looking for the keys that are placed somewhere among other things on the table. Even though you're trying to find one single item, you still scan all the items, and your brain quickly decides whether these are the keys or not. This is how Image Detection works.

The technology is used not only for detecting needed objects. Another popular application area is fake image detection. Using it, you can tell the original picture from the photoshopped or counterfeited one. It is a very powerful and much-needed tool in the modern online world.

1.3 WHAT IS IMAGE CLASSIFICATION?

It is a process of labeling objects in the image – sorting them by certain classes. For example, ask Google to find pictures of dogs and the network will fetch you hundreds of photos, illustrations and even drawings with dogs. It is a more advanced version of Image Detection – now the neural network has to process different images with different objects, detect them and classify by the type of the item on the picture.

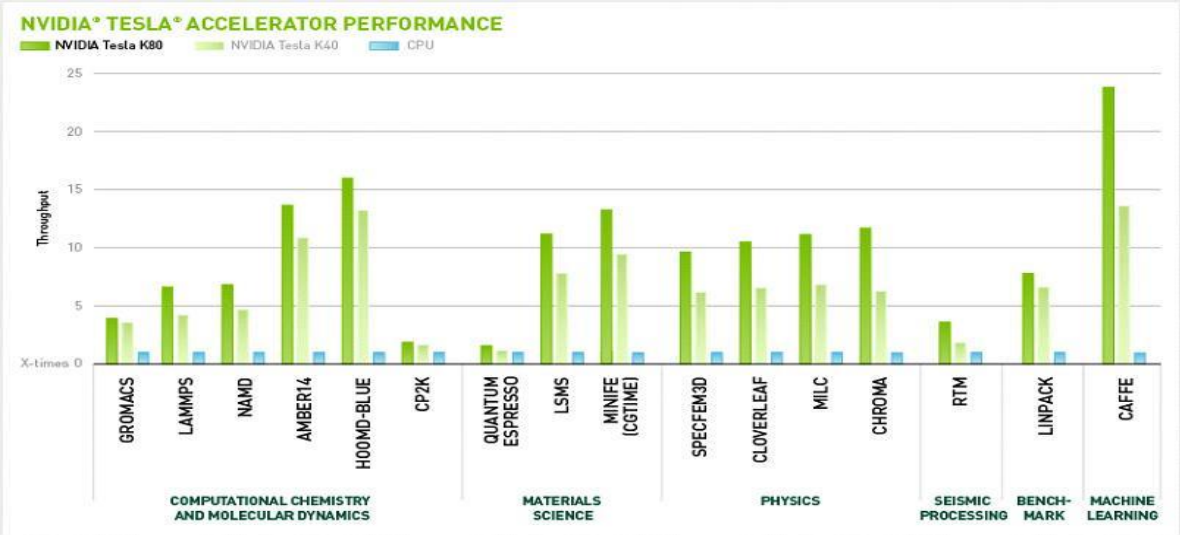
1.4 HOW MACHINE LEARNING IMPROVES IMAGE DETECTION?

This activity of looking for a specific object among others is really simple for a human brain. We do it all the time, we are used to this process. However, computers have obvious challenges with this seemingly easy task. That's why computer engineers around the world are trying their best to train Artificial Intelligence on how to find the needed objects in pictures. And this is no small task for developers.

To train the AI tool to detect certain objects, you have to show these objects first. In other words, you should 'feed' AI with the labeled data – images containing the needed objects, item coordinates, location, and class labels. The most frequently asked question here is "How many images are needed?" The answer is the more, the better.

Also, you should choose images with different locations of the object, so that items change their coordinates and sizes during machine learning. It will help AI understand that even though this object can be located in different places on the image and be both big and small, these changes don't affect its class.

So, as you can see, it is a time-consuming process that requires lots of resources and efforts. But let's look on the bright side. Artificial Intelligence is already making quite a progress here. With GPUs – Graphics Processing Units – deep learning has become much faster and easier. GPU is an electronic circuit that allows to manipulate the memory and accelerate graphics processing.

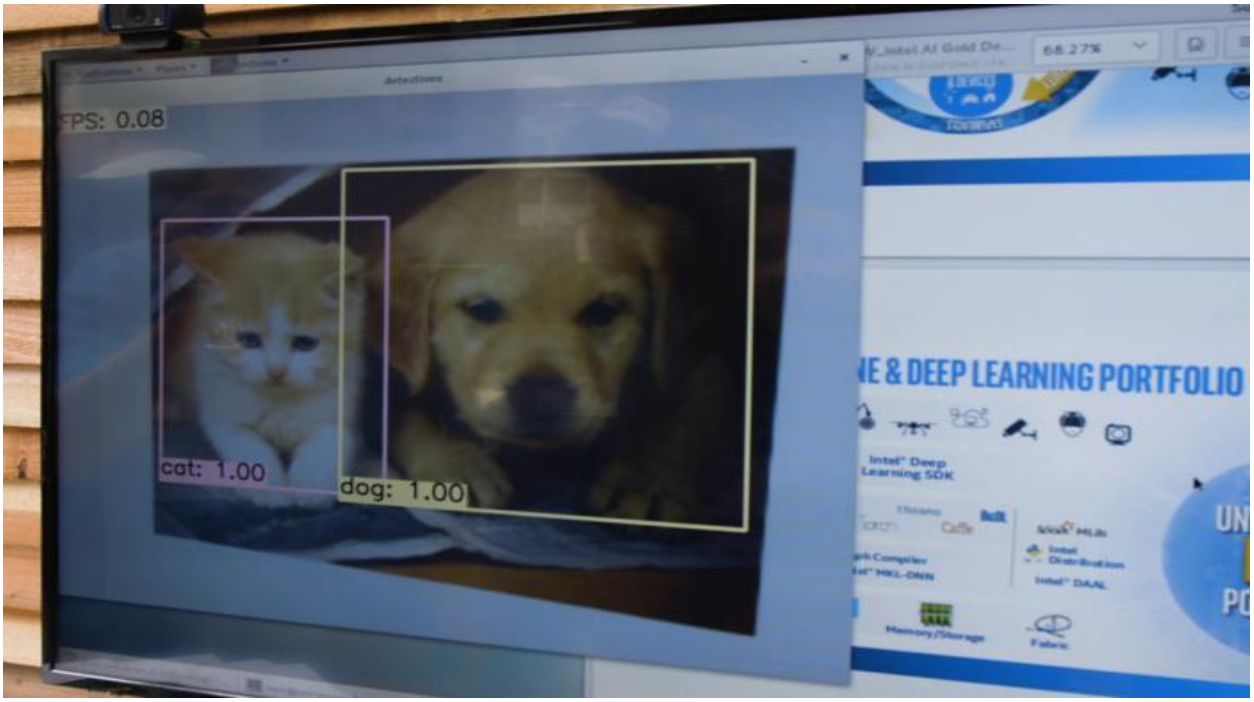


Obviously, deep learning is the best option for image detection. Training a single deep neural network to solve several problems is more efficient than training several networks to solve one single problem. Thus, smaller parts of the deep neural network will improve its overall performance.

1.5 HOW TO TRAIN A NEURAL NETWORK TO CLASSIFY IMAGES?

There are different types of machine learning solutions for image classification. But the best and the most accurate one is CNN – the Convolutional Neural Network. To understand how it works, let's talk about convolution itself. It's a process during which two functions integrate producing a new product. When it comes to pictures, we have to think of an image as a matrix of pixels. Each pixel has its own value but is integrated with other pixels, and it generates a result – an image.

CNN applies filters to detect certain features in the image. The way the convolutional neural network will work fully relies on the type of the applied filter. So, when applying machine learning solutions to image classification, we should provide the network with as many different features as possible. It will then analyze their values upon training.



1.6 PROBLEM STATEMENT

The main aim of this project is to detect the name of the object. For example, if an unknown object is given and we will give the name of that object. As we are new to Machine Learning, we're basically working with 505 objects like a cup, scissors, etc.., After that, we will use more objects. Our dataset consists of jpg files and they are of different home objects.

DATASET DESCRIPTION

2. DATASET DESCRIPTION

2.1 HARDWARE & SOFTWARE REQUIREMENTS:-

- **RAM:** 8GB.
- **Processor:** Intel® Core™ i7 – 5500U CPU @ 2.40 GHZ
- **Operating System:** Windows.
- **Environment:** Google Colab .
- **Language:** Python (v.3.8.1)
- **Domain:** Convolutional Neural Networks**.**
- **Dataset:** Caltech 101.

2.2 DATASET DESCRIPTION:-

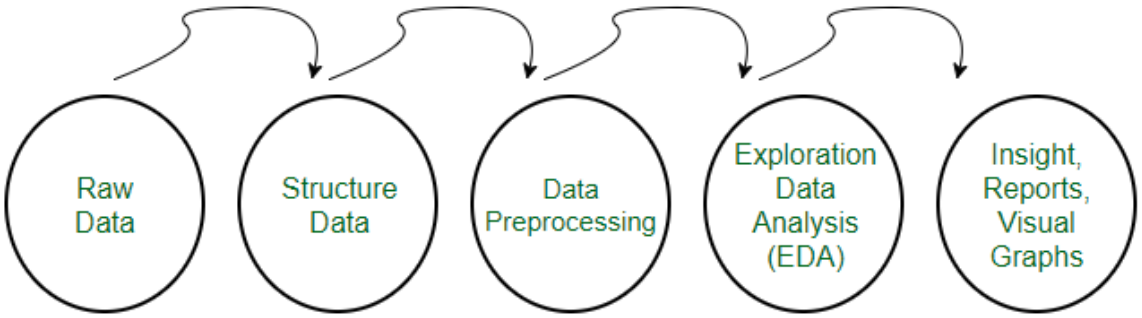
- **Dataset Name:** ObjectCategories.
- **Source:** Caltech 101.
- **Classes:** 10.
- **No of Images:** 505.
- The dataset consists of images (.jpg files) of different home objects like cup, scissors, etc., Each image consists of a single object.
- **Classes:**
    1. Cup
    2. Scissors
    3. Binocular
    4. Camera
    5. Chair
    6. Ceiling-fan
    7. Headphone
    8. Lamp
    9. Umbrella
    10. Wrench

DATA PREPROCESSING

3. DATA PREPROCESSING

3.1 DATA PRE-PROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.



3.1.1 Need of Data Pre-processing:-

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.

3.1.2 Resizing all the Images:

- As data consists of different sized images, we need to resize all the images into the same shape.
- So, each image is resized into 300 X 300 pixels

3.2 NOISE:

- Dataset didn’t have any blurred images.

PRELIMINARY VISUALIZATION

4. PRELIMINARY VISUALIZATION

4.1 PRELIMINARY VISUALIZATION

Data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data. To communicate information clearly and efficiently, data visualization uses statistical graphics, plots, information graphics and other tools. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message.

4.1.1 Importance of Visualization:-

The CSV data (panda dataframes) can be really difficult to approach if you want to get some insights. It doesn't matter if your data is formatted or not formatted correctly.

The way the human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than poring over spreadsheets or reports. Data visualization is a quick, easy way to convey concepts in a universal manner — and you can experiment with different scenarios by making slight adjustments.

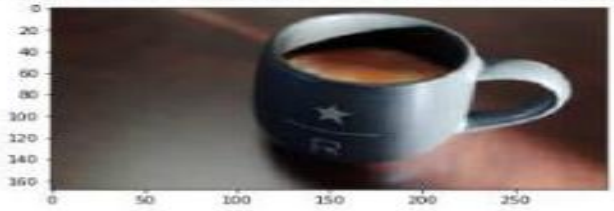
4.1.2 Why Visualization:-

Do you think giving you the data of 1 Million points in a table/Database file and asking you to provide your inferences by just seeing the data on that table is feasible? Unless you're a super human it's not possible. This is when we make use of Data visualization, wherein all the data will be transformed into some form of plots and analyzed further from that. As being a human, we are more used to grasp a lot of info from diagrammatic representation than the counterparts. As a human, we can just visualize anything in either in 2-d or 3-d. But trust me almost of the data that you obtain in real world won't be this way. As a Machine learning engineer, working with more than 1000-dimensional data is very common. So what can we do in such cases where data is more than 3D? There are some **Dimensionality Reduction(DR)** techniques
like PCA, TSNE , LDA etc., which helps you to convert data from a higher dimension to a 2D or 3D data in order to visualize them. There may be some loss of information with each DR techniques, but only they can help us visualize very high dimensional data on a 2d plot. TSNE is one of the state of the art DR technique employed for visualization of high dimensional data.

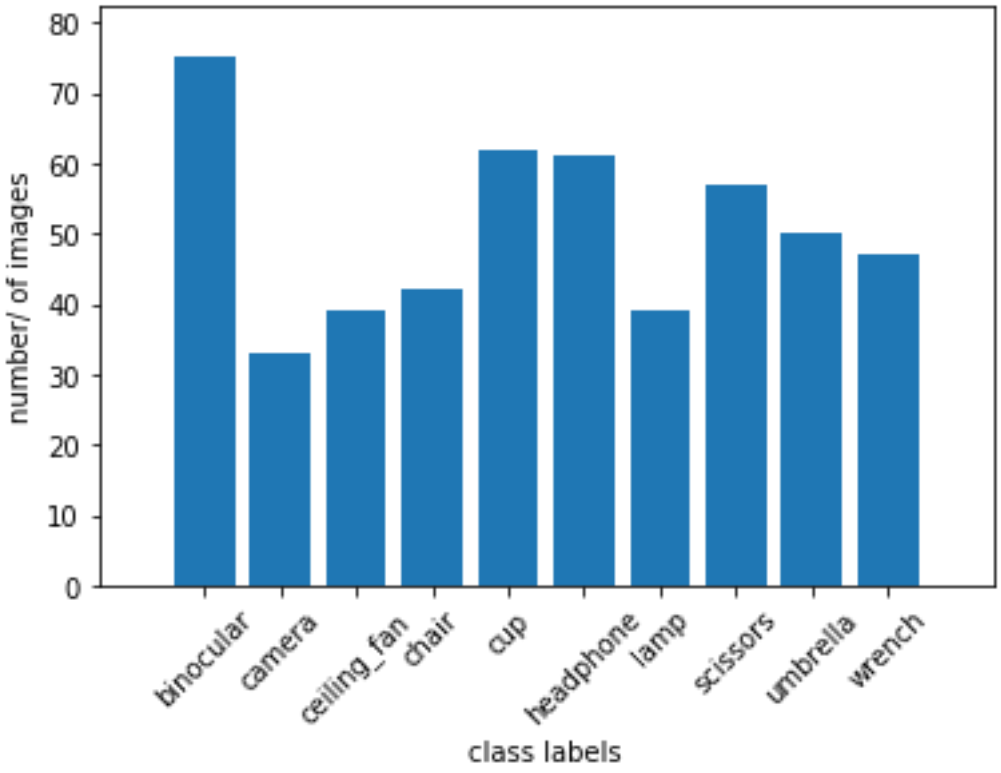
4.1.3 Data Visualization:-

The first plotting shows the diagram before resizing the image and the second plotting shows the image after resizing it.

```
[81] plt.imshow(img)

     <matplotlib.image.AxesImage at 0x7f9cdc37ba90>
```




```
[82] img1 = cv2.resize(img,(300,300))

[83] plt.imshow(img1)

     <matplotlib.image.AxesImage at 0x7f9cdc2dbba8>
```



DATA FLOW DIAGRAMS

5. DATA FLOW DIAGRAM

5.1 DATA FLOW DIAGRAM

A **data-flow diagram** (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

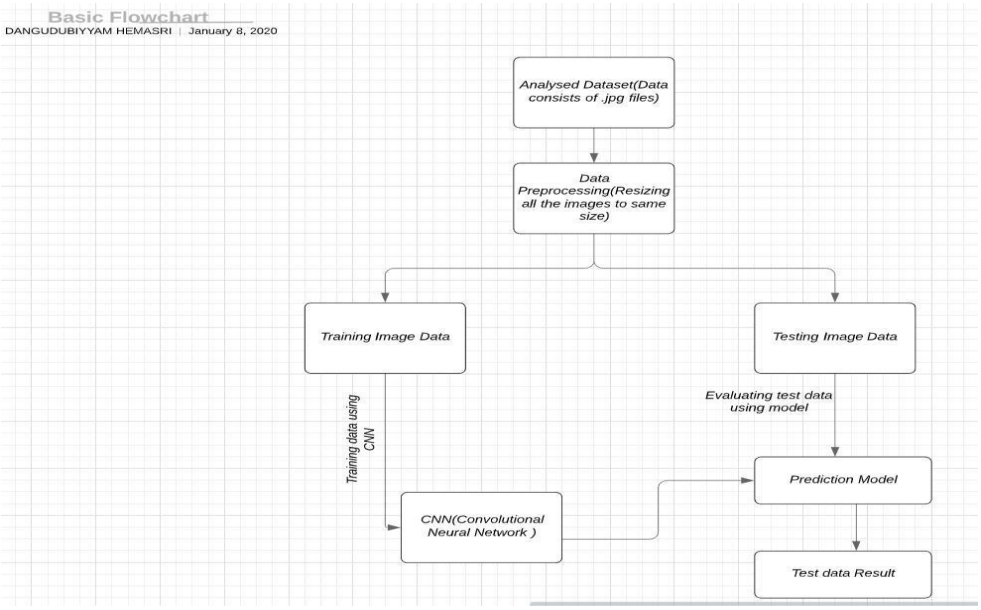
There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of Structured Analysis.

For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

The data-flow diagram is part of the structured-analysis modeling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.

\



IMPLEMENTATION

6. IMPLEMENTATION

6.1 IMPLEMENTATION

It is standard in ML to split data into training and test sets. The reason for this is very straightforward: if you try and evaluate your system on data you have trained it on, you are doing something unrealistic. The whole point of a machine learning system is to be able to work with unseen data: if you know you are going to see all possible values in your training data, you might as well just use some form of lookup.

6.1.1 Splitting the Data Set:

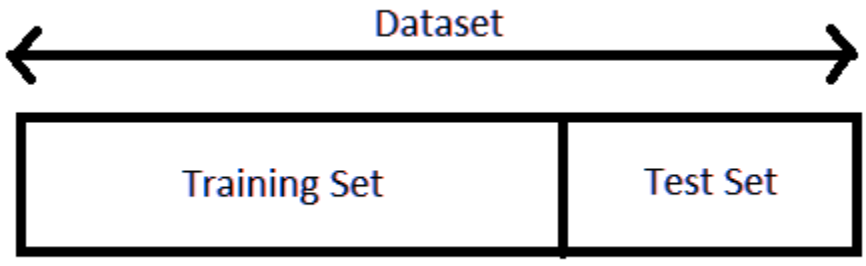
As we have seen,In Machine learning we have two kinds of datasets

6.1.2 Training Data

The part of data we use to train our model. This is the data which your model actually sees(both input and output) and learn from.

6.1.3 Testing Data:

Once our model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of Testing data, our model will predict some values (without seeing actual output). After prediction, we evaluate our model by comparing it with actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.



6.2 SAMPLE CODE:

6.2.1 Importing modules and mounting dataset:

Importing modules required for the program and mounting dataset from Google drive.

```
[ ] import os
    import matplotlib.pyplot as plt
    from skimage.transform import resize
    #from sklearn.utils.validation import train_test_split
    import cv2
    import numpy as np
    from matplotlib.pyplot import imshow

    from sklearn.preprocessing import LabelEncoder
    from keras.utils import to_categorical
    from sklearn.model_selection import train_test_split


    from keras.models import Sequential, Model
    from keras.layers import Conv2D, Input, Dropout, Activation, Dense, MaxPooling2D, Flatten, GlobalAveragePooling2D
    from keras.optimizers import Adadelta
    from keras.callbacks import ModelCheckpoint
    from keras.callbacks import EarlyStopping
    from keras.models import load_model
```

▾ dataset mounting from google drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=1
```

6.2.2 Exploring Dataset:

Walking into the directory of dataset and making count of every image in each class and making X to store image and Y to store category of that image.

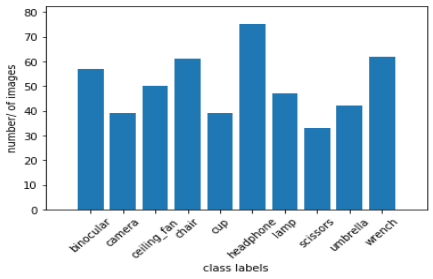
```
[ ] path = '/content/drive/My Drive/'
    dataset_path = '/content/drive/My Drive/dataset'

    class_labels = []

    # features = []
    # labels = []
    count_of_images = []

    X = np.ndarray((505, 300, 300, 3), dtype=np.uint8)
    Y = []
    c = 0
    for i in os.listdir(dataset_path):
        a = dataset_path + "/" + i
        if not a.endswith(".ipynb_checkpoints"): #Images only
          for k in os.listdir(a):
              l = a + "/" + k
              img = cv2.imread(l)
              img1 = cv2.resize(img,(300,300))
              # features.append(img1)
              # labels.append(classes[i])
              X[c] = img1
              Y.insert(c, i)
              c += 1
          count_of_images.append(len(os.listdir(a)))
    #print(count_of_images)
    # print(features)
    print(len(X))
```

6.2.3 Plotting graph:

Plotting graph based on number of images for every class.

```
[ ]  class_labels = list(classes.keys())
     # features1 = np.stack(features,axis = 0).reshape(505,300*300*3)
     #print(sum(count_of_images))
     x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
     plt.xlabel('class labels')
     plt.ylabel('number/ of images')

     plt.margins(0.1)
     plt.xticks(x, class_labels, rotation = 45)
     plt.bar(class_labels, count_of_images)
```

```
<BarContainer object of 10 artists>
```




```
imshow(X[0])
```

6.2.4 One_Hot and Normalization:

Changing X to normalized_x by changing to float type and dividing with 255 and changing Y to label_encoder and then to categorical data.

```
[ ]  label_encoder = LabelEncoder()
     Y_integer_encoded = label_encoder.fit_transform(Y)
     Y_integer_encoded
```

```
array([4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 7, 7, 7, 7, 7, 7, 7, 7, 7,
       7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
       7, 7, 7, 7, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
       6, 6, 6, 6, 6, 6, 6, 6, 6, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
       9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
       9, 9, 9, 9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
       8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3])
```

```
[ ]  len(Y_integer_encoded)
```

```
505
```

```
[ ] Y_one_hot = to_categorical(Y_integer_encoded)
    Y_one_hot

array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)

[ ] Y_one_hot[0]

array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.], dtype=float32)

len(Y_one_hot)

505

[ ] X_normalized = X.astype(np.float64) / 255
    X_normalized[0]

array([[[0.05882353, 0.02745098, 0.42352941],
        [0.05882353, 0.02745098, 0.42352941],
        [0.05882353, 0.02745098, 0.42352941],
        ...,
        [0.05882353, 0.02745098, 0.42352941],
        [0.05882353, 0.02745098, 0.42352941],
        [0.05882353, 0.02745098, 0.42352941]],
```

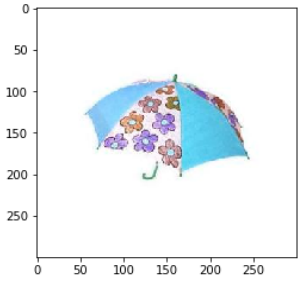
6.2.5 Splitting Data:

Splitting data as train data and test data with 75% and 25% respectively.

```
[ ] X_train, X_validation, Y_train, Y_validation = train_test_split(X_normalized, Y_one_hot, test_size = 0.25, random_state =

[ ] X_train.shape

(378, 300, 300, 3)
```

checking whether the image and class name matched or not

```
[ ] imshow(X_train[89])

<matplotlib.image.AxesImage at 0x7f9050e3d470>
```



6.2.6 Model Description:

- Sequential Model used to train the data.
- As the images are of 2D shape. So, we added Conv2D layers.
- Max Pooling is used to reduce spatial dimensions of the output volume.
- Activation parameter used to keep the code cleaner.

```
[ ] def classLabel(val):
      for key, value in classes.items():
        if(value == val):
          return key
```

```
[ ] classLabel(np.argmax(Y_train[89]))
```

```
'umbrella'
```

```
[ ] Y_train.shape
```

```
(378, 10)
```

```
[ ] model_cnn = Sequential()
    model_cnn.add(Conv2D(8, (3,3), activation='relu', input_shape=(300,300,3)))
    model_cnn.add(Conv2D(16, (3,3), activation='relu'))
    model_cnn.add(MaxPooling2D(pool_size=2, strides=2))
    model_cnn.add(Conv2D(32, (3,3), activation='relu'))

    model_cnn.add(MaxPooling2D(pool_size=2, strides=2))
    model_cnn.add(Flatten())
    model_cnn.add(Dense(10, activation='softmax'))
    model_cnn.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:66: The name tf.get_defa

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:541: The name tf.placeho
```

TEST CASES

7. TESTCASES

7.1 TESTCASES:-

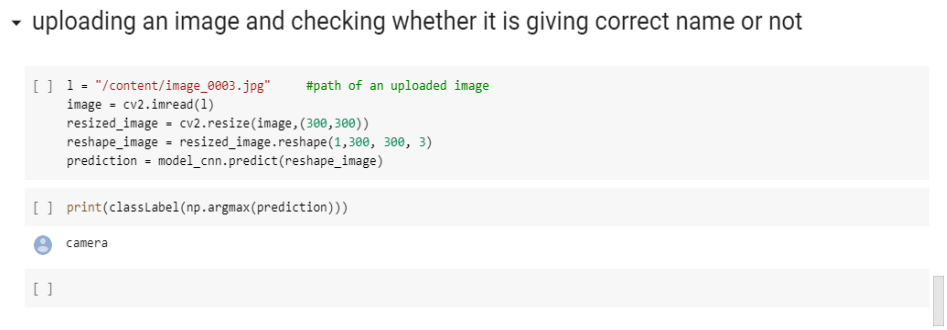
In software engineering, a **test case** is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Test cases underlie testing that is methodical rather than haphazard. A battery of test cases can be built to produce the desired coverage of the software being tested. Formally defined test cases allow the same tests to be run repeatedly against successive versions of the software, allowing for effective and consistent regression testing.

For applications or systems without formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class. In some schools of testing, test cases are not written at all but the activities and results are reported after the tests have been run.

In scenario testing, hypothetical stories are used to help the tester think through a complex problem or system. These scenarios are usually not written down in any detail. They can be as simple as a diagram for a testing environment or they could be a description written in prose. The ideal scenario test is a story that is motivating, credible, complex, and easy to evaluate. They are usually different from test cases in that test cases are single steps while scenarios cover a number of steps of the key.

7.1.1 Checking the result of given test case:

Uploaded a camera image, the path of it is declared in the name of 'l' as we observe below code. Resizing the image and sent to the predict function of a trained CNN model. As we predict the class label of the image by making as an argument to the ClassLabel function. The result is a "camera".



CONCLUSION

8. CONCLUSION

8.1 ACCURACY:

- As we fit the model to the trained data with a batch size of 64 and epochs 10.
- After the tenth epoch the accuracy observed is 95%.
- So, the result of any uploaded image will give a correct answer.

```
[21] model_cnn.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is deprecated. Please use tf.math.log instead.
```

```
[22] model_cnn.fit(X_train, Y_train, batch_size=64, epochs=10, verbose=1, validation_data=(X_validation,Y_validation))

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

Train on 378 samples, validate on 127 samples
Epoch 1/10
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:216: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

378/378 [==============================] - 43s 114ms/step - loss: 8.5400 - acc: 0.1058 - val_loss: 9.7708 - val_acc: 0.1181
Epoch 2/10
378/378 [==============================] - 41s 109ms/step - loss: 6.4469 - acc: 0.1138 - val_loss: 2.3853 - val_acc: 0.1890
Epoch 3/10
378/378 [==============================] - 41s 109ms/step - loss: 2.2012 - acc: 0.2249 - val_loss: 2.2331 - val_acc: 0.2126
Epoch 4/10
378/378 [==============================] - 41s 109ms/step - loss: 1.8274 - acc: 0.4735 - val_loss: 2.1487 - val_acc: 0.2992
Epoch 5/10
378/378 [==============================] - 41s 109ms/step - loss: 1.4433 - acc: 0.6270 - val_loss: 1.9067 - val_acc: 0.4173
Epoch 6/10
378/378 [==============================] - 41s 110ms/step - loss: 0.9448 - acc: 0.7487 - val_loss: 1.9330 - val_acc: 0.5039
Epoch 7/10
378/378 [==============================] - 42s 110ms/step - loss: 0.6410 - acc: 0.8333 - val_loss: 2.3295 - val_acc: 0.3937
Epoch 8/10
378/378 [==============================] - 42s 110ms/step - loss: 0.4138 - acc: 0.8757 - val_loss: 2.2367 - val_acc: 0.4803
Epoch 9/10
378/378 [==============================] - 41s 110ms/step - loss: 0.2452 - acc: 0.9444 - val_loss: 2.7591 - val_acc: 0.4331
Epoch 10/10
378/378 [==============================] - 41s 110ms/step - loss: 0.1584 - acc: 0.9524 - val_loss: 2.9061 - val_acc: 0.4173
<keras.callbacks.History at 0x7f6e6fb4f4e0>
```

BIBLIOGRAPHY

10. BIBLIOGRAPHY

- **https://www.ismll.uni-hildesheim.de/lehre/ml-07w/skript/ml-2up-01-linearregression.pdf**
- **https://www.geeksforgeeks.org/ml-linear-regression/**
- **http://www.statsoft.com/Textbook/Classification-and-Regression-Trees**
- **https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/**
- **https://www.geeksforgeeks.org/random-forest-regression-in-python/**
- **https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf**
- **https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d**
- **https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/**
- **Link to book by** Sebastian Raschka: **https://g.co/kgs/wuvoFs**
- **Link to book by** Aurelien Geron**:**https://g.co/kgs/o1PvuQ