



UNIVERSITY OF MISSOURI-KANSAS CITY

CS5542

Big Data Analytics and Applications
Spring 2017

Project: ODD – ONE -- OUT.

Third Increment Report

Borusu, Vijaya Lakshmi Meenakshi. 5

Sevana, Uday Kiran Dora. 35

Pallepati, Rakesh Reddy. 31

Sudulaguntla, Sai Sriharsha. 49

ODD ONE OUT

INDEX

1. Introduction
2. Project Objective
 - 2.1. Motivation
 - 2.2. Significance
 - 2.3. Features: Use Case/Scenario
3. Approach
 - 3.1. Schedules
 - 3.1.1. Data Sources
 - 3.1.2. Analytic Tools
 - 3.1.3. Analytical Tasks
 - 3.1.4. Expected Inputs/Outputs
 - 3.1.5. Algorithms
4. Application Specification
 - 4.1. System Specification (Big Data Analytics Server/Clients)
 - 4.1.1. Software Architecture
 - 4.1.2. Features, workflow, technology
 - 4.1.3. Existing Application/Services Used: Name, Description, URL
5. Model Implementation
6. Documentation
7. Project Management.
8. IEEE Paper.

1.Introduction

In the recent time, AI has advanced to a new heights. With the invention of Machine and Deep Learning the current technology is able to achieve a lot more than possible before.

From finger print scanning to face recognition and voice recognition the process of analysis has improved significantly and has made life easy for us. Not only do these features have obvious uses but they help in many ways previously not thought of, like using the technology to let cars see the road using image recognition, searching tissues for cancer cells and watching space for unknown patterns, all of these seem like finding the **odd one out** game we used to play in our childhood.

ODD ONE OUT(OOO) is a simple strategy but has staggering usage that can benefit us in many ways. This project of ours take the inspiration from the **odd one out** and builds the foundations for a much complex application by trying to perfect the algorithm of finding the **odd one out** among various images.

2.Project Objective

2.1 Motivation:

Image Analysis is a developing field which has had many applications in the real world scenario. Be it developing a AI bot to monitor our streets to keep us safe or monitoring space feed to find anomalies or to process mages of blood cells to find cancer cells. This is how image analysis and AI combined have taken the standard of monitoring to a new level. This project is our take on this developing technology. We try to implement basic features and procedures to develop an application with Deep learning or machine learning. We also try to answer the question of which is the best of these both.

2.2 Significance:

The significance of this application is that it is based on the image or video feed that is given as an instance. We not only extract the tags or annotations but also try to find how different the objects are from each other.

This can be achieved by the API, Machine Learning or Deep learning program. In the first phase, we use the clarifai API where image/video analysis is done. Then we try to extract more information using machine learning.

Here we train the model by giving enough training sets thus, queries can be answered with more efficiency. Finally in the deep learning we add the intelligence aspect so that even slight differences can be easily identified. So, this makes the application more intelligible.

2.3 Features: Use Case/Scenario

The basic use case is as follows. An image is given as an input and the annotations for the images are generated. These annotations are stored in a file where they are later processed.

In the mean while a set of predefined labels are given for similarity and this dataset is used for comparison for of the annotations, of which a few are excluded and the resulting annotations are the one which are like each other.

The excluded annotations are considered as odd one out. If there are no annotations that are excluded then the objects in the images are considered to be similar.

3.Approach

3.1 Schedules

3.1.1 Data Sources

Currently we are using the following sources for our image datasets for training and test validation purposes.

1. Caltech 256 Data Set. (http://www.vision.caltech.edu/Image_Datasets/Caltech256/)
2. canstockphoto.com(http://www.canstockphoto.com/images-photos/odd-one-out_2.html)
3. Personally captured images.

3.1.2 Analytic Tools

Microsoft Cognitive Services: Microsoft Cognitive Services lets build apps with powerful algorithms using just a few lines of code. They work across devices and platforms such as iOS, Android, and Windows, keep improving, and are easy to set up.

Cloud Vision API, Clarifi API.

3.1.3 Analytical Tasks

Image Edge Detection, K-Means variation among various attributes of datasets considered.

3.1.4 Expected Inputs/Outputs

For examples, considering the below image of Holland Soccer Supporters as input, the systems identifies and classifies the Not a soccer fan as the odd one out.

Example Input Data:



Expected Out Come: For explanation purposes, Identified Odd one out is Outlined in Red.



3.1.5 Algorithms

HOG(Histogram of Oriented Gradients) features extract algorithm: For detection of key Features in images.

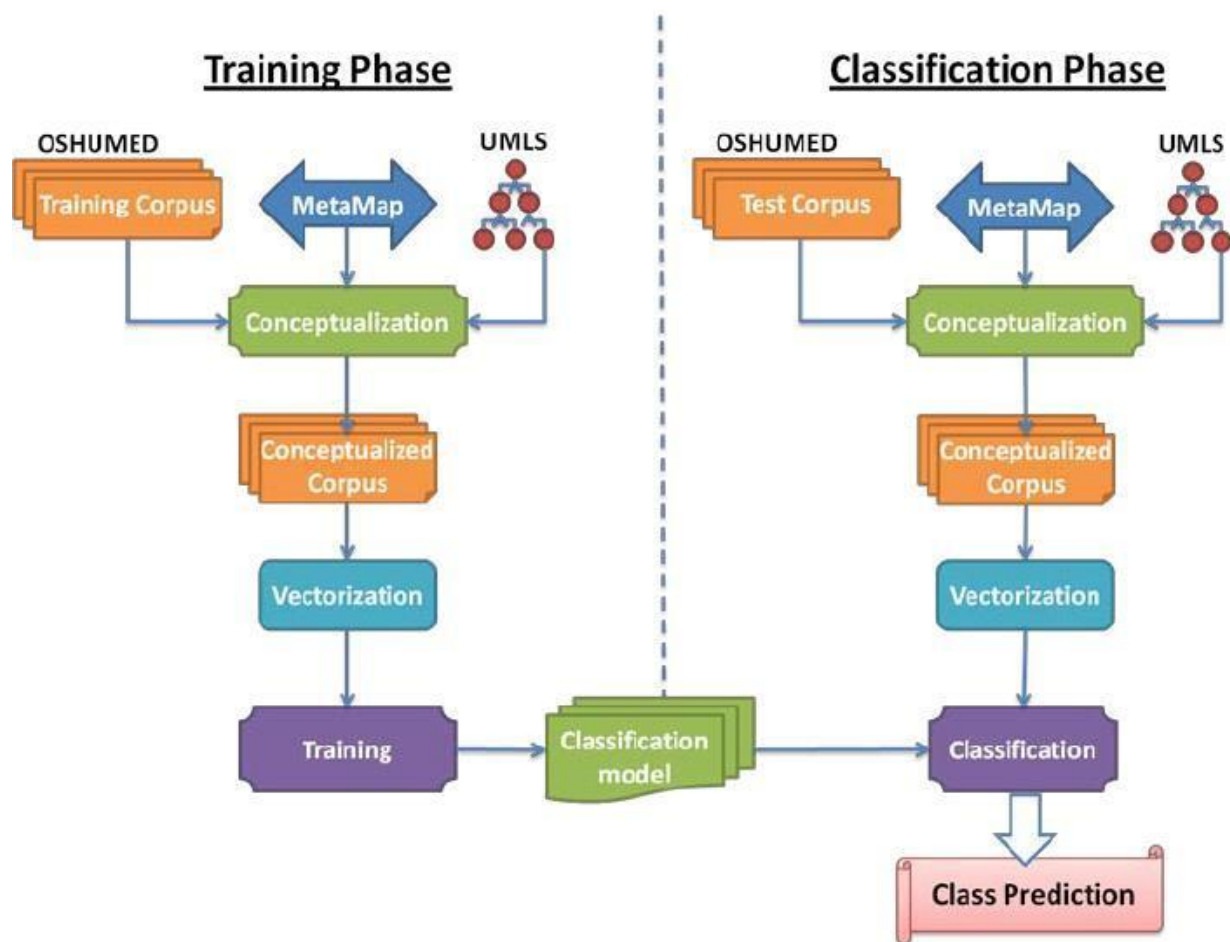
Support Vector Machine – an unsupervised machine learning Algorithm for Feature Vector generation

Clarifi API – for Image Classification Purposes.

4. Application Specification

4.1 System Specification

4.1.1 Software Architecture:

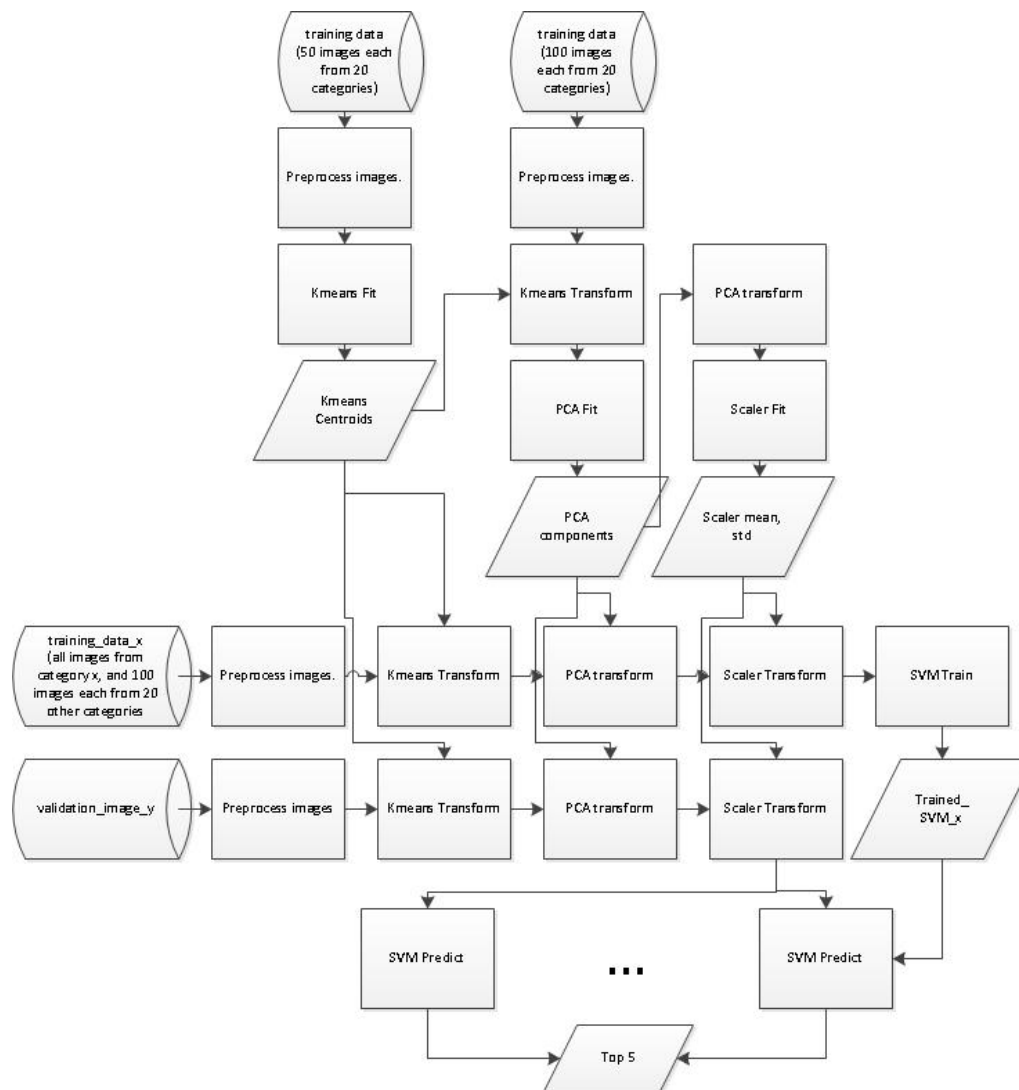


4.1.2 Features, workflow, technology

1. Activity Digaram (workflow, data, task)

The below image represents the activity Diagram of our Image Classification Application.

It represents both vertical and horizontal categorized processing techniques of both Training and Test Data Sets.



4.1.3 Existing Application/Services Used: Name, Description, URL

IntelliJ IDE Platform – a Java Integrated Development Environment.

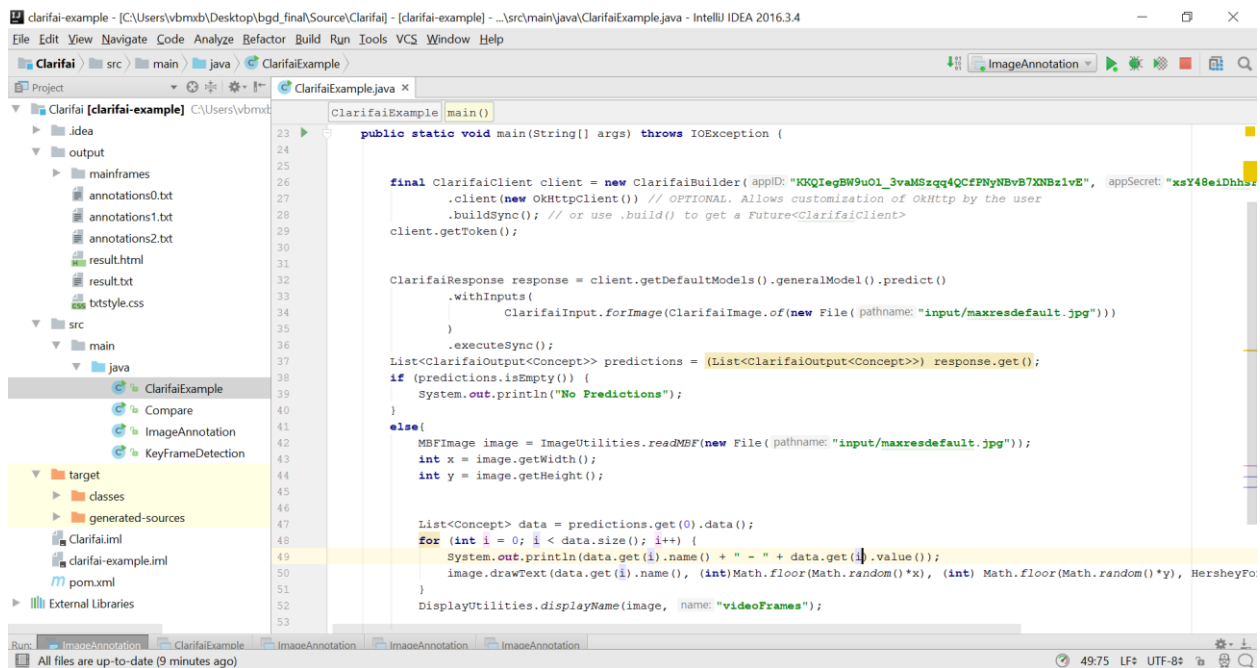
Clarifi API- Image and Video Recognition API.

<https://www.clarifai.com/api>

5.Implementation

1. In the clarifi api, preprocessing the data: As our first challenge, we took a set of images of
2. different sizes, hues, lighting conditions, et cetera. The algorithms we were about to use cannot tolerate such variance most of them expect a standard normalized feature input.
3. As a first stab at normalizing our data, we converted each color image to grayscale. Initially grayscale images were unchanged.
4. This would hopefully minimize the effect of lighting conditions (or color vs blackandwhite originals) on our downstream classifiers, as well as simplify the processing we'd have to do.
5. For example we'd only have to find centroids on single color values, rather than a threedimensional RGB matrix Kmeans: we intended to run Kmeans on our collection of patches as a sort of edge detector to find N_CLUSTERS common patterns.

6.Documentation: Clarifai Code:



```
23 public static void main(String[] args) throws IOException {
24
25
26     final ClarifaiClient client = new ClarifaiBuilder( appID: "KKQIegBN9u01_3vaMSzqq4QCfPNyNBvB7XNBz1vE", appSecret: "xsY48eiDhhs
27         .client(new OkHttpClient()) // OPTIONAL. Allows customization of OkHttpClient by the user
28         .buildSync(); // or use .build() to get a Future<ClarifaiClient>
29     client.getToken();
30
31
32     ClarifaiResponse response = client.getDefaultModels().generalModel().predict()
33         .withInputs(
34             ClarifaiInput.forImage(ClarifaiImage.of(new File( pathname: "input/maxresdefault.jpg" )))
35         )
36         .executeSync();
37     List<ClarifaiOutput<Concept>> predictions = (List<ClarifaiOutput<Concept>>) response.get();
38     if (predictions.isEmpty()) {
39         System.out.println("No Predictions");
40     }
41     else{
42         MBFImage image = ImageUtilities.readMBF(new File( pathname: "input/maxresdefault.jpg" ));
43         int x = image.getWidth();
44         int y = image.getHeight();
45
46
47         List<Concept> data = predictions.get(0).data();
48         for (int i = 0; i < data.size(); i++) {
49             System.out.println(data.get(i).name() + " - " + data.get(i).value());
50             image.drawText(data.get(i).name(), (int) Math.floor(Math.random()*x), (int) Math.floor(Math.random()*y), HersheyFo
51         }
52         DisplayUtilities.displayName(image, name: "videoFrames");
53     }
```

Result:

umbrella

Deep Learning : Inception Model

Image_Classification - [C:\Users\RANDOM\Desktop\ImageClassification\image_classification] - [image_classification] - ...src\main\scala\IPApp.scala - IntelliJ IDEA 2016.3.4

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

image_classification > src > main > scala > IPApp.scala

IPApp.scala x annot x SimpleServer.scala x IPSettings.scala x build.sbt x ImageUtils.scala x

```
25 val vocabulary = ImageUtils.vectorsToMat(model.clusterCenters)
26 val desc = ImageUtils.bowDescriptors(path, vocabulary)
27 val histogram = ImageUtils.matToVector(desc)
28
29 println("-- Histogram size : " + histogram.size)
30 println(histogram.toArray.mkString(" "))
31
32 val nbModel = RandomForestModel.load(sc, IPSettings.RANDOM_FOREST_PATH)
33 val p = nbModel.predict(histogram)
34 (s" + IMAGE_CATEGORIES(p.toInt))
35 }
36
37
38 def testImage(string: String):String = {
39   val conf = new SparkConf()
40   .setAppName(s"IPApp")
41   .setMaster("local[*]")
42   .set("spark.executor.memory", "6g")
43   .set("spark.driver.memory", "6g")
44   System.setProperty("hadoop.home.dir", "C:\\Users\\RANDOM\\Desktop\\hadoopwin\\winutils")
45   val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
46   val sc= SparkContext.getOrCreate(sparkConf)
47   val res = testImageClassification(sc, string)
48   val pw = new PrintWriter(new FileWriter("data3/annot", true))
49   pw.append(res+" \n")
50   pw.close()
51   printf(res+"\n");
52   def printArray(arr:Array[String], Index:Int)
53   {
54     if(Index<arr.length)
55     { print(arr[Index]+"")
56       printArray(arr, Index+1)
57     }
58   }
59 }
```

Compilation completed successfully in 4s 891ms (11 minutes ago)

UI - [C:\Users\RANDOM\Desktop\ImageClassification\UI] - _Index.html - WebStorm 2016.3.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

UI > index.html

Project > C:\Users\RANDOM\Desktop\image_classification\UI

index.html x oddout.html x annot.html x read.js x txtstyle.css x annot x custom.js x

Caltech_101

- test
 - birds
 - helicopter
 - motorbikes_side
 - pizza
 - train
 - umbrella
 - facebook_icon.png
 - animal.gif
- css
 - bootstrap.css
 - custom.css
 - thumbnail-gallery.css
- fonts
- js
 - bootstrap.js
 - custom.js
 - jquery.js
- annot.html
- index.html
- oddout.html
- read.js
- txtstyle.css

```
64 <div class="container">
65
66   <div class="row">
67
68     <div class="col-lg-12">
69       <h1 class="page-header">Image Class Prediction</h1>
70     </div>
71     <div class="row inputRow">
72       <div class="col-md-4 thumb">
73         <a class="thumbnail" href="#">
74           
75         </a>
76       </div>
77       <div class="col-md-1 text-center ver-centr">
78         <input class="vertical-center" id="clk" type="button" value="Predict" />
79       </div>
80       <div class="col-md-4">
81         <h3 id="txt_area"></h3>
82         
83       </div>
84     </div>
85
86     <h4>Select an Image:</h4>
87
88     <div class="col-lg-3 col-md-4 col-xs-6 thumb">
89       <a class="thumbnail" href="#">
90         
91       </a>
92     </div>
93     <div class="col-lg-3 col-md-4 col-xs-6 thumb">
94       <a class="thumbnail" href="#">
95
```

Platform and Plugin Updates
WebStorm is ready to [update](#).

Platform and Plugin Updates: WebStorm is ready to update. (37 minutes ago)

74:104 LF: UTF-8

Web Interface


Image classification exam x

localhost:3474/#

uday

Apache Spark - Image Classification

Image Class Prediction



Predict

bird

Select an Image:








Image classification exam x

localhost:3474/#

uday

Apache Spark - Image Classification





Image Class Prediction

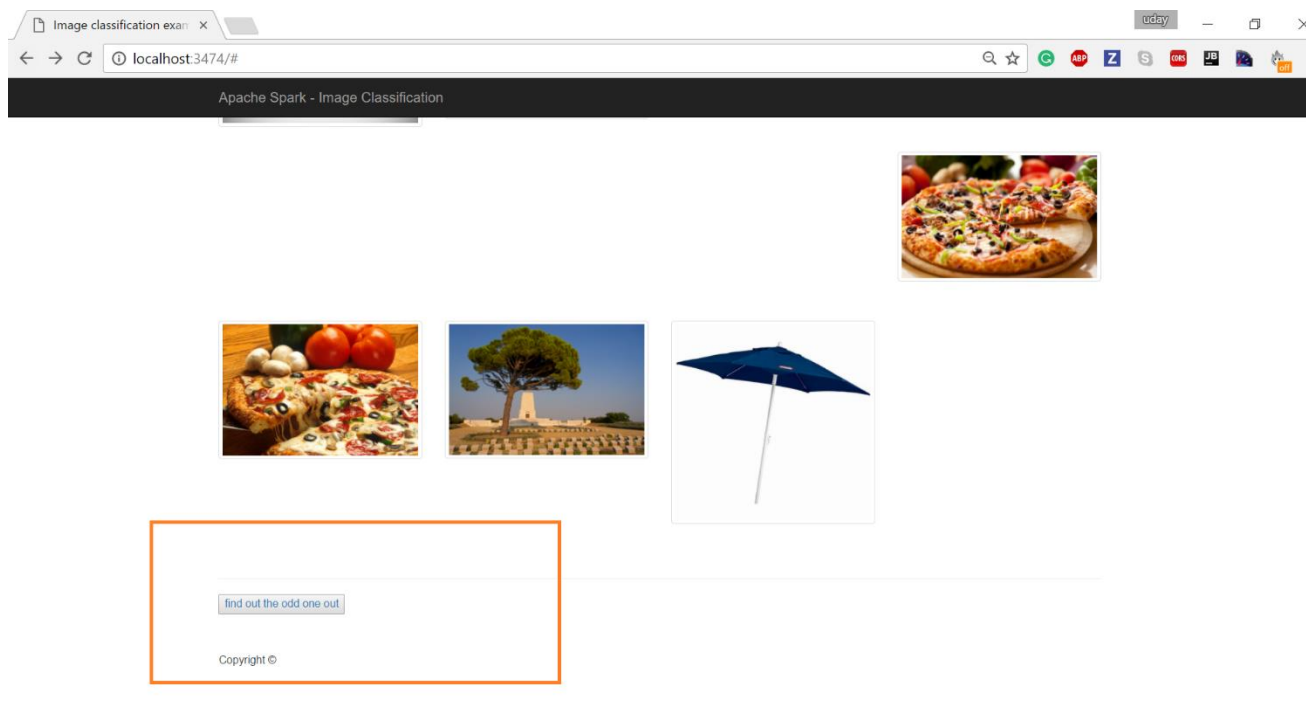
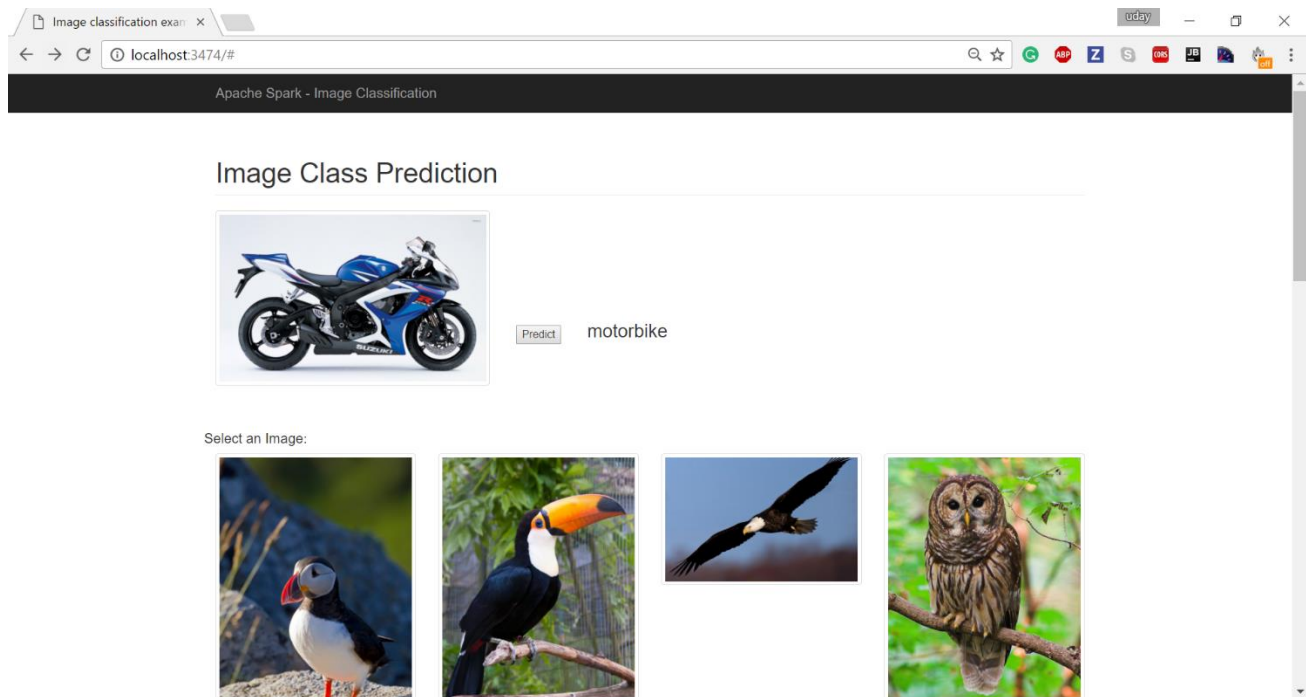


Predict

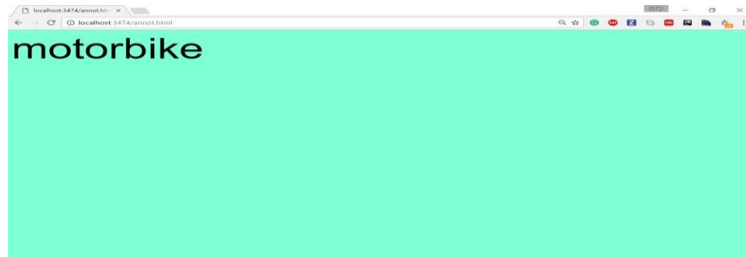
bird

Select an Image:





Result:



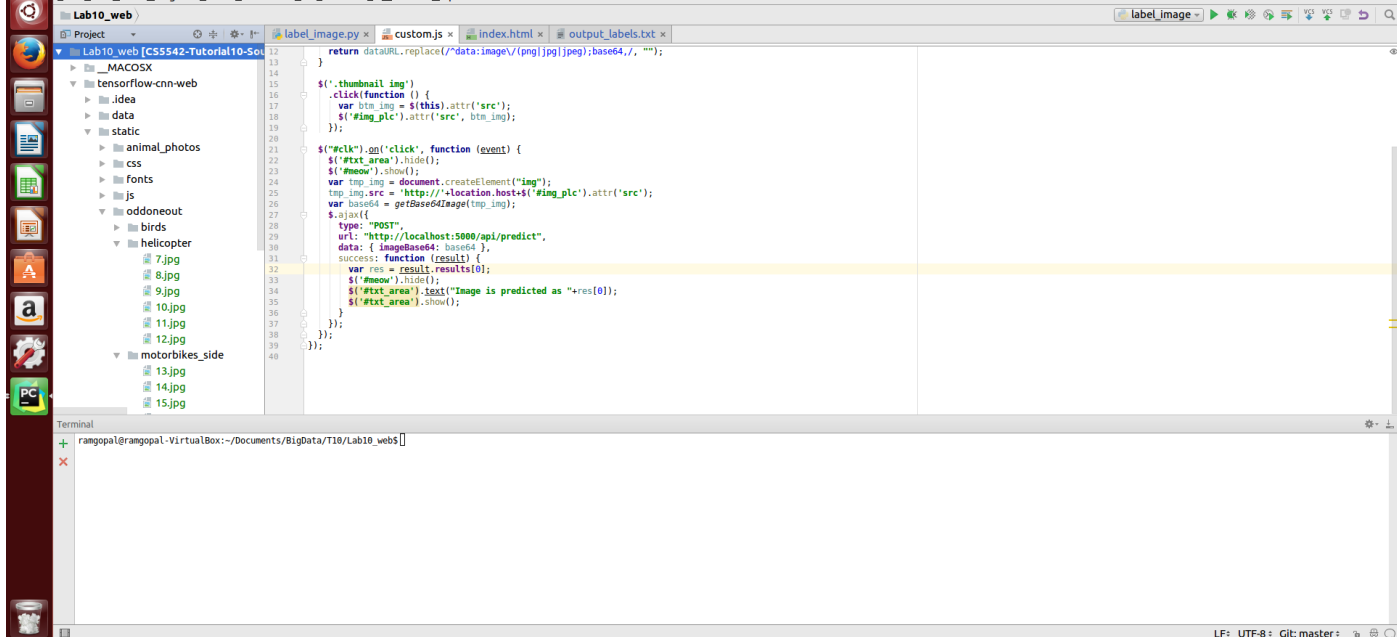
Deep Learning:

Ubuntu1404 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Lab10_web - [~/Documents/BigData/T10/Lab10_web] - tensorflow-cnn-web/static/js/custom.js - PyCharm 2016.3.2

File Edit View Navigate Code Refactor Run Tools VCS Window Help

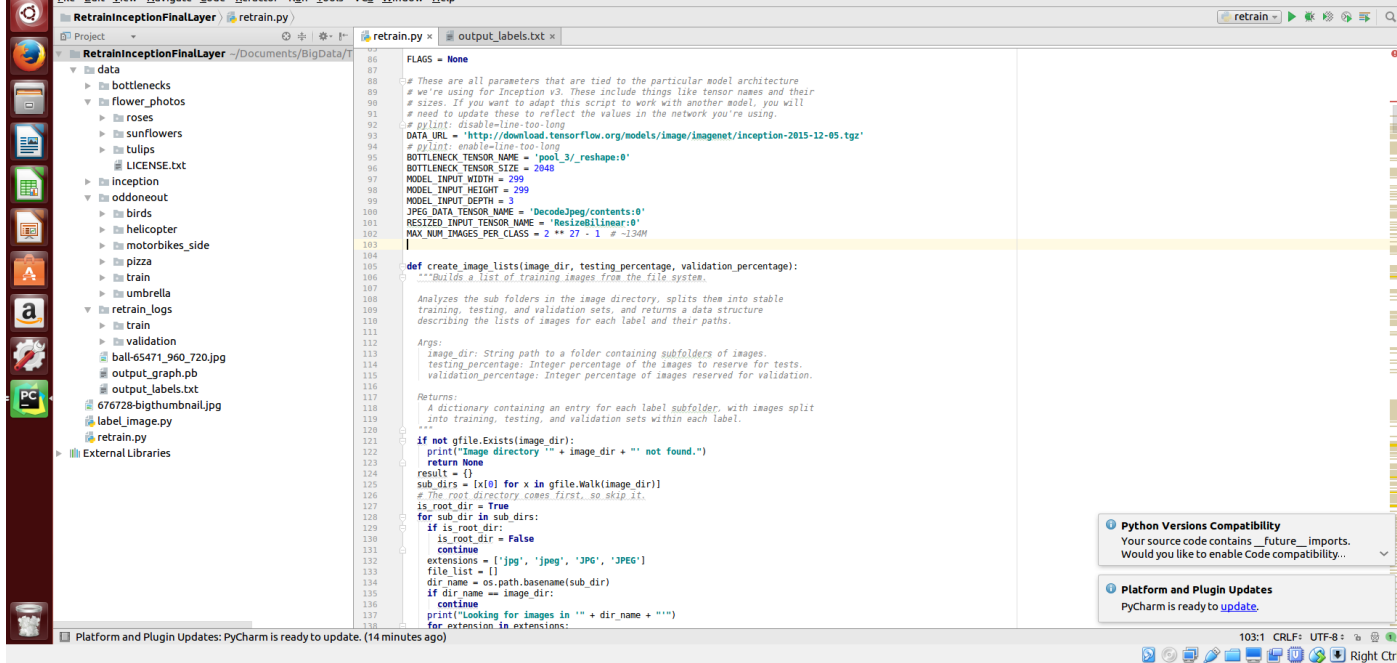


Ubuntu1404 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

RetrainInceptionFinalLayer - [~/Documents/BigData/T10/RetrainInceptionFinalLayer] - retrain.py - PyCharm 2016.3.2

File Edit View Navigate Code Refactor Run Tools VCS Window Help



Ubuntu1404 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

image classification example - Mozilla Firefox

127.0.0.1:5000/#

Tensorflow - Image Classification

Image Class Prediction








Image is predicted as helicopter

Select an Image:



Ubuntu1404 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Image classification example - Mozilla Firefox

127.0.0.1:5000/#

Tensorflow - Image Classification

Image Class Prediction





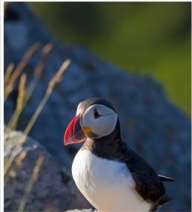
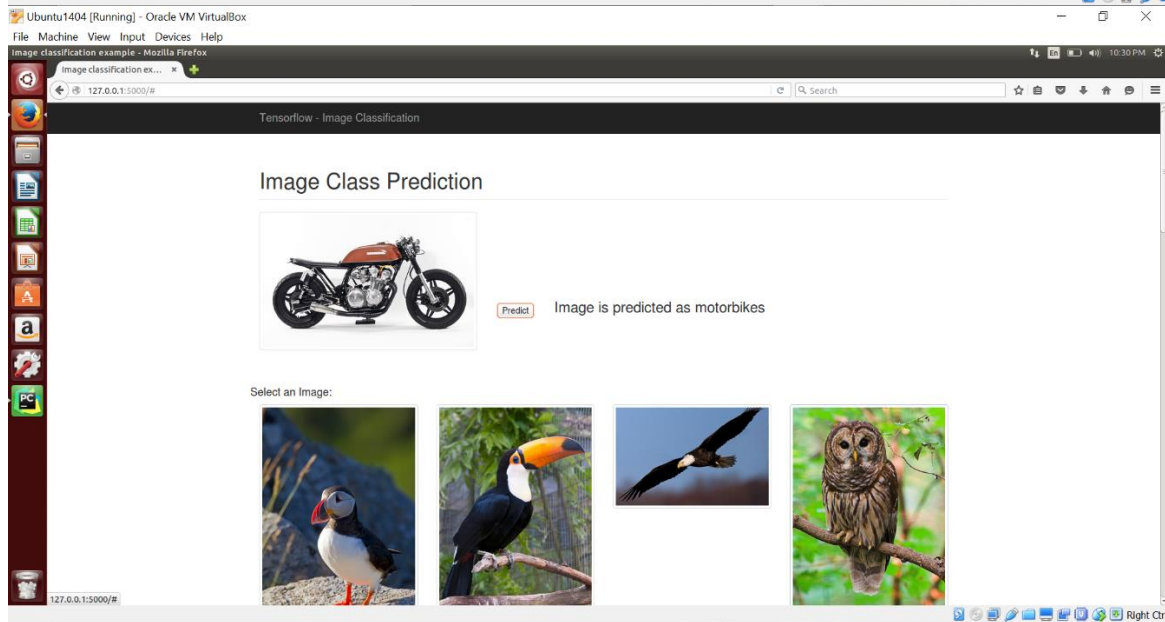
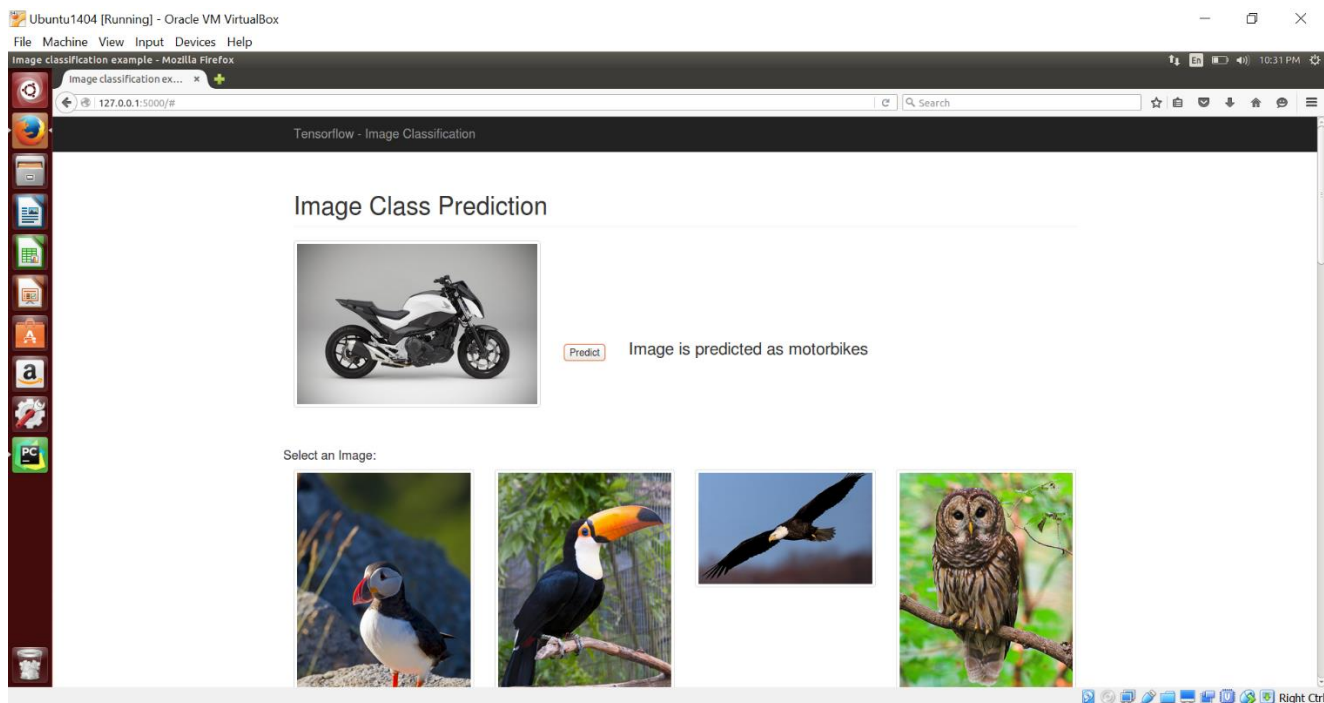


Image is predicted as motorbikes

Select an Image:





Result:

helicopter

9. Project Management:

Implementation status Report.

Work Completed	Description	Responsibility	Time Taken Dev Hours	Contributions
Clarifai, Google Conversion API.	Integration of Clarifai API in the model	Uday Meenakshi	10	50% Each
TensorFlow	Machine Learning Mdoel	Rakesh Harsha	10	50% Each
Scala Coding, Machine Learning	For Clarifai API	Rakesh Meenakshi	8	50% Each
Data Set & Training	Collection of Data Set	Harsha Uday	4	50% Each
Documnetation	Project Report, Project Management, IEEE Paper	Meenakshi Uday Rakesh Harsha	8	25% Each
Testing	Testing the Trained Model for Accuracy and Result	Meenakshi Uday Rakesh Harsha	4	25% Each

Issues and Concerns: Issues developed with implementation of functionalities are resolved with the help of TA Naga.

Project Management:

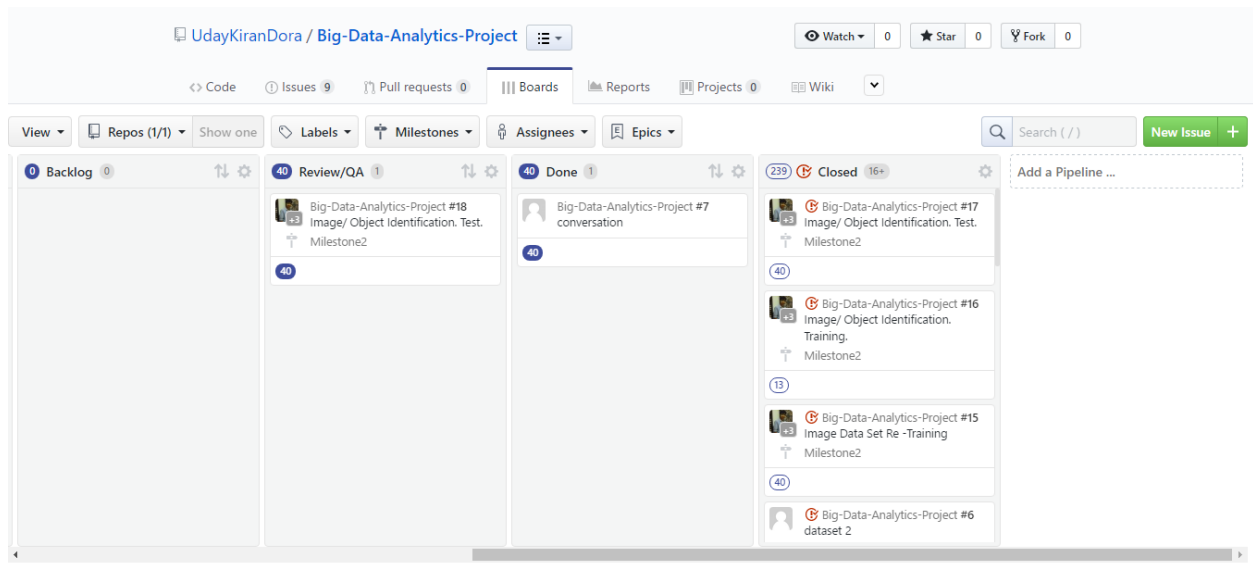
The entire project is developed in 3 iterations, which are covered in two milestones.

Total User Stories: 18.

All the Issues logged were Closed.

All the members Contributed Equally and Efficiently.

Kanban Board:



Milestone 2 Graph:

Milestone2



Final Kanban Board:

UdayKiranDora / Big-Data-Analytics-Project

Watch 0

Star 0

Fork 0

<> Code

Issues 9

Pull requests 0

Boards

Reports

Projects 0

Wiki

View

Repos (1/1)

Show one

Labels

Milestones

Assignees

Epics

In Progress 0

Backlog 0

Review/QA 0

Done 0

319

Closed

18+

Big-Data-Analytics-Project #7 conversation

40

Big-Data-Analytics-Project #18 Image/ Object Identification. Test. Milestone2

40

Big-Data-Analytics-Project #17 Image/ Object Identification. Test. Milestone2

40

Big-Data-Analytics-Project #16 Image/ Object Identification. Training. Milestone2

Image Identification; Classification, Visual Grouping, Recognition and Learning.

Borusu Vijaya Lakshmi Meenakshi,
University of Missouri Kansas City.
vbmxp@mail.umkc.edu

Pallepati Rakesh Reddy,
University of Missouri Kansas City.
Rpd54@mail.umkc.edu

Sevana Uday Kiran Dora,
University of Missouri Kansas City.
Uskc2@mail.umkc.edu

Sudulaguntla Sai Sriharsha,
University of Missouri Kansas City.
Ss42f@mail.umkc.edu

Abstract

We present an image recognition system, developed using end-to-end deep learning. We propose a CNN pre-training technique based on a novel auxiliary task called odd-one-out learning. In this task, the machine is asked to identify the unrelated or odd element from a set of otherwise input elements. We apply this technique to our learning model where we input sample images and ask the network to learn to predict the odd image among all the input images. The odd image is sampled such that all the input images are recognized and categorized accordingly with their corresponding annotation and the algorithm displays the annotation label of the odd image among all the input images.

Thus, the odd image's annotations are displayed as the output. All the odd images' annotation is displayed in the output. I.e. If the system identifies one odd image out of three input images it's corresponding annotation is displayed, if system identifies all the input images belongs to different categories then system displays their corresponding annotations as the odd-one-out.

Our machine learning model is developed such as it is implemented in the form of multi-stream convolution neural network in which end-to-end learning procedure is implemented

Post image classification, our model achieved 93% accuracy on the CALTECH 101 dataset.

I. INTRODUCTION.

Convolutional neural networks are currently the new state of the art machine learning frameworks. Accounting their success in handling larger datasets such as ImageNet, we have developed our CNN model to identify the odd image in set of input image, as models like these can be adapted as additional security purposes in web applications such as captcha. We developed our proposed odd-one-out such that the learning machine on Tensor Flow identifies all the input images and classifies them according to their category of similarity, and the odd input image annotation is displayed as the output. If all the input images belong to the same category, the learning machine categorizes them in a one category as there is no odd category present no odd-one-out annotation is displayed in the output.



Bonsai.



No odd Image.

II. RELATED WORK.

The most common techniques studied include autoencoders [1, 2], restricted Boltzmann machines [3],

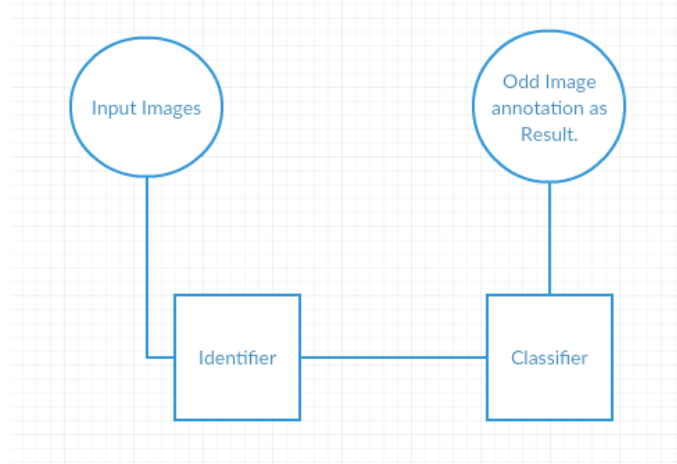
convolutional deep belief networks [4], LSTMs and recurrent neural networks.

III. PROPOSED WORK

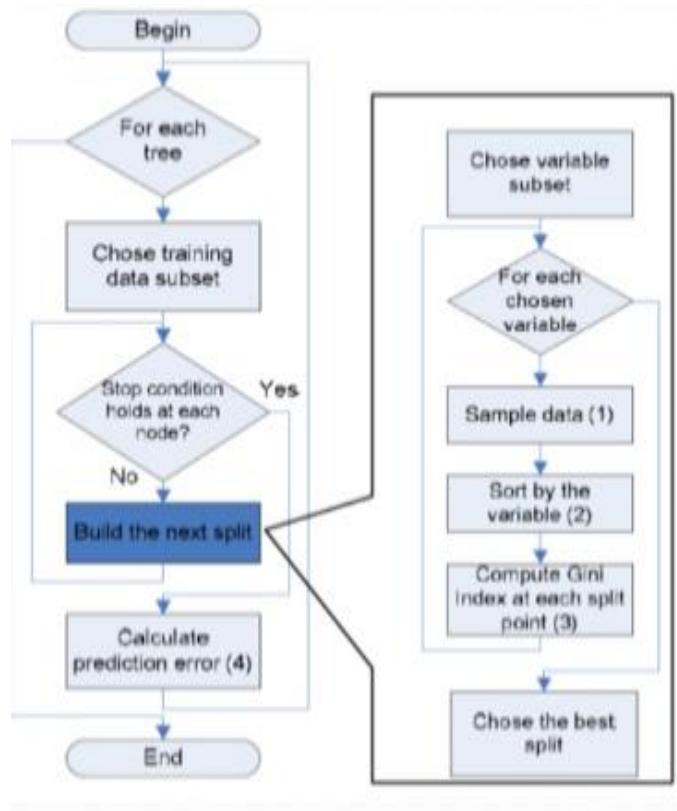
A. Proposed Models

The basic use case is as follows. An image is given as an input and the annotations for the images are generated. These annotations are stored in a file where they are later processed. In the mean while a set of predefined labels are given for similarity and this dataset is used for comparison for of the annotations, of which a few are excluded and the resulting annotations are the one which are like each other. The excluded annotations are considered as odd one out. If there are no annotations that are excluded then the objects in the images are similar.

B. UML Diagrams



Random Forest Algorithm:



C. Evaluations and Results:

Images from Caltech 101 Data are used to train the model, and tested with random test images. Model was accurate upto 70%. Top -1 Error was 24% and Top -5 Error is 18%.

V. DISCUSSION & LIMITATIONS

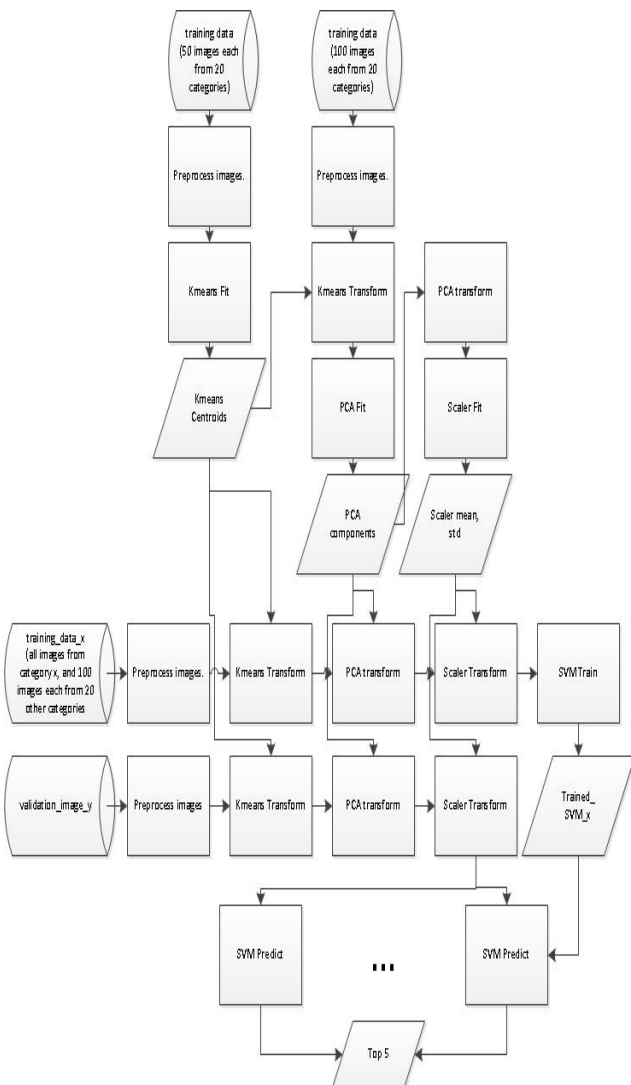
We observed that the top -5 and Top -1 error are minimized by training the model with larger datasets. Accuracy of the model is 70%, which can be improved with Residual Networks.

IV. IMPLEMENTATIONS AND EVALUATIONS

Performed K-Means Clustering and Random Forest Algorithm is implemented.

A. Software Architecture

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the



a) Results Table:

1	Results		
	Error rate	Top -1	Top -5
	Data Set: Caltech 101.	24%	18%

^a Sample of a Table footnote. (Table footnote)

^b.

VI. REFERENCES

[1] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics, 59(4-5):291–294, 1988. 1, 2 J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and helmholtz free energy. NIPS, 1994. 1

[3] G. E. Hinton and T. J. Sejnowski. Learning and relearning in boltzmann machines. Parallel distributed processing: Explorations in the microstructure of cognition, 1:282–317, 1986. 2

[4] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In NIPS, 2009. 2

[5] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetism Japan, p. 301, 19

