CoBoRo: An Interactive cook book powered by Spark

Group 3: Prudhvi Myneni, Mahidhar Varma, Pruthvi Raj C, Sudheer Gourishetty

1. Abstract

Future homes will all be equipped with robots where robots will be utilized in every possible way to make our lives simple. So in this project we present how we have programmed WowWee Robot to serve as a cookbook assistant. This interactive cookbook robot has multi-function like suggesting recipes, playing music, etc. For our architecture we have used a simple server client architecture. We have used apache spark as our background engine to process data in parallel as batches. Our main idea was to use machine learning algorithms available with apache spark to aid our robot provide better results and find new insights from the data without being explicitly programmed. Our implementation of the robot is different to the ones that are already available. So in this paper we talk about our approach and the implementation of the project.

2. Introduction

Cooking with friends makes cooking fun. So you would always want someone to be by your side to help you with cooking or narrate a recipe. This was our main motivation that led to program a robot which would become your cook mate. Not only providing recipes, narrating them, it would also play songs and interact with you just like any other friend.

The most significant feature of this robot is that it can recognize the raw ingredient suppose a vegetable by having a look at it. So once it recognizes the vegetable it finds suitable top rated recipes for the shown vegetables and starts narrating the recipe on the users go. It can start or stop the narration anytime based on the user's input.

3. Problem analysis and proposed solution

As the main requirement of our project is image recognition, we wanted it to be accurate. We have explored different machine learning algorithms that would give the desired accuracy. Image recognition has become quite challenging as we are dealing with a large data that has a variety of shapes and colors. So we wanted to use a machine learning algorithm that would best suite our requirements. Upon comparing the different machine learning algorithms logistic regression for classification provided reliable results.

Also we found that image recognition for fruits and vegetables required good feature extraction methods as vegetables and fruits have similar shapes. On evaluating the different feature extraction techniques we found that the FAST corner detection technique was more suitable for real-time applications.

3.1 Algorithms

For our analysis we searched for algorithms that was popular in image recognition and had support with Apache Spark. Finally we narrowed down our search to three algorithms, Logistic Regression, Decision Tree Learning and Naïve Bayes.

Logistic Regression is a regression model that is similar to linear regression but the dependent variable is categorical i.e. it can take on one of a limited number of possible values.

Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating the probabilities using a logistic function which produces a logistic curve which is limited to values between 0 and 1. In other words, Logistic Regression calculates the probability of particular outcomes. Although it is called logistic regression it mostly does classification.

Decision Tree learning algorithm uses a decision tree as a prediction model. This model maps the observations of an item to the items target value. The decision tree has a flow chart like structure where internal node denotes a test on an attribute, each branch represents an outcome of the test and each leaf node has the class label. This method approximates the discrete-valued target function where the learned function is represented by the decision tree.

Naïve Bayes Algorithm is a classification algorithm based on Bayes rules that assumes the attributes are all conditionally independent of one another. This value of assumption is made using a probabilistic model. This model is a popular method used for text categorization.

From our testing results with sample data we have decided to use Logistic Regression over the other two algorithms as it had higher accuracy.

3.2 Feature Extractors

To improve the accuracy of the image recognition we mainly focused on the popular edge detection methods available. We have considered ORB, FAST corner detection, SIFT SURF methods and use the one with best detection capabilities.

ORB stands for Oriented FAST and Rotated BRIEF which uses FAST key point detection and BRIEF visual descriptor.

FAST is short for Features from accelerated segment test which is a corner detection method. As the name suggests, this method is faster than many other feature extraction methods. Fast Corner detector is most suitable for real time image or video processing applications.

SIFT stands for Scale-invariant feature transform. SIFT is most robust and scalable feature extractor but not suitable for real time applications.

SURF is short for Speeded up Robust Features. It was inspired from SIFT and is several times faster than it. Although faster, SIFT outperforms it and more robust.

We chose FAST corner detection as our feature extractor in our project as it was able to accurately detect more key points than the other methods.

3.3 **Implementation of Algorithm and Feature Extractors**

For image classification in our project we first obtain important features from an input using FAST corner detection, which is a part of Opencv framework for key points extraction. All these key points and descriptors are extracted using FAST from a training data set and then clustered together into N centroids using k-means. So for any image all the descriptors are mapped together to a nearby cluster and a histogram vector is constructed using these clustered descriptors. Therefore an image can be represented by a histogram vector.

So the histogram of the image that has to be classified is compared with the histograms from the training set and classified accordingly.

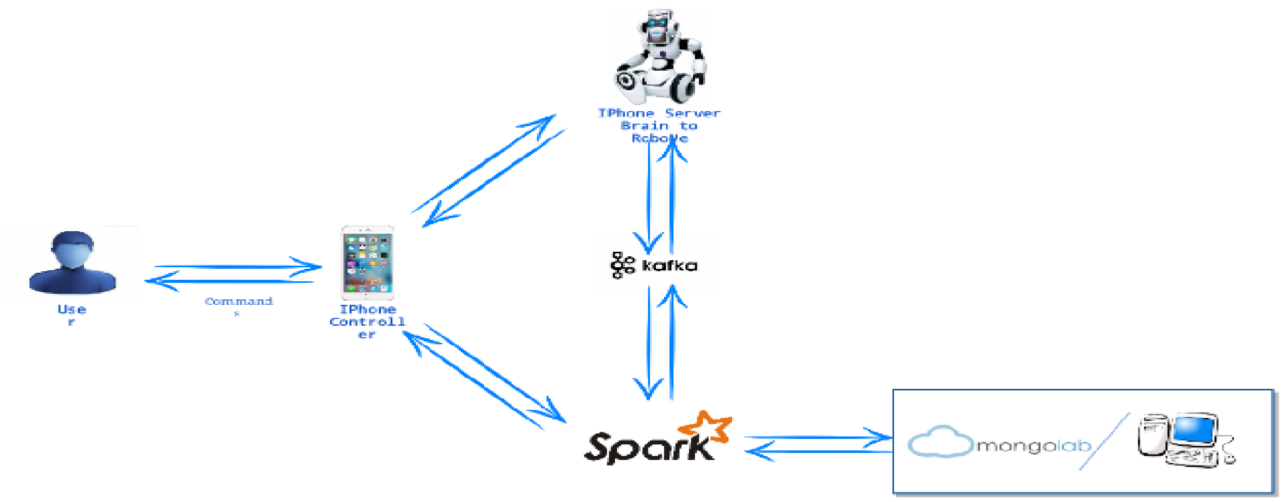


Fig 1. System Architecture.

4. System Architecture

Fig 1. Depicts the architecture that we implemented in our system. This architecture has two main components one is a mobile client and the other is the big data server. The user interacts with a mobile client via a simple UI design and can give simple movement commands or query for recipes and weather. The mobile client communicates with the big data server. The big data server is where the image classification and other processing occurs. The results from processing in the big data server are sent back to the mobile client for the user.

4.1 Design of Big Data Server

In our proposed software architecture the mobile client sends the users input to the spark server. Apache spark is the core of our server mainly for processing image data. We chose spark because it is one of the growing engines for big data processing. So as our project needed a fast batch processing server we opted for spark to be our best choice. As we are dealing with a lot of big data we needed data parallelism and distributing data across multiple computation programs. Spark uses RDD's for doing this. RDD's are an immutable representation of data which do operation at the memory layer and also they provide fault tolerance.

Fig 2. Shows the activity diagram of our big data server. First, spark uses a training data set to generate models. Based on these models spark provides classification to the inputs provided by the user via a mobile client.

Fig 3. Shows the sequence diagram of the big data server. In our architecture we used an IOS device to be our mobile client. And our server is actually another IOS device that acts as brain to the robot. This keeps communication with apache spark in the background. So the input from an IOS client is received on the app delegate function of the server which passes it to GCDSynchronize function which accepts any new connection and is used to write/read data to spark. After the transaction is complete the connection is disconnected.

The server expects an image from the mobile client which is then sent to spark for classification. The spark classifies the image and send the result back to the mobile client.

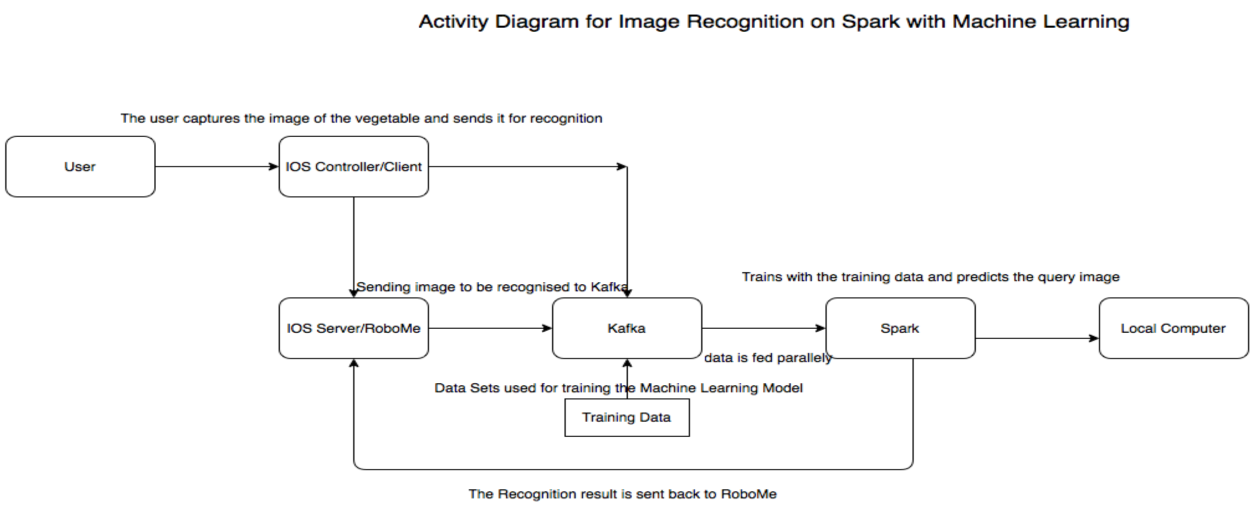

Activity Diagram for Image Recognition on Spark with Machine Learning
The user captures the image of the vegetable and sends it for recognition
User
IOS Controller/Client
Sending image to be recognised to Kafka
Trains with the training data and predicts the query image
IOS Server/RoboMe
Kafka
Spark
Local Computer
data is fed parallely
Data Sets used for training the Machine Learning Model
Training Data
The Recognition result is sent back to RoboMe

Fig 2. Activity Diagram of Big Data Server

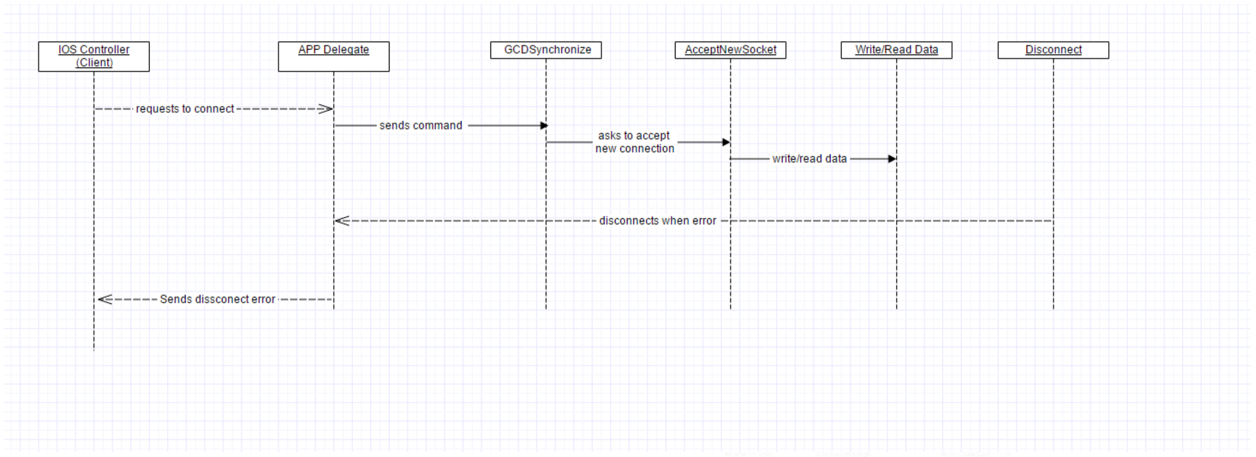
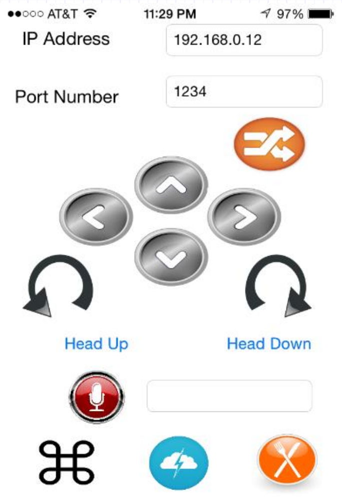


Fig 3. Sequence Diagram of Big Data Server.



4.2 Design of Mobile Client

The mobile client is capable of capturing the image of vegetable that is to be recognized and then send the image data or any other user’s queries to the mobile big data server. The UI was built using Xcode. The UI also includes features like movement control, voice to text feature.

Fig 4. Shows a screenshot of the UI.

Fig 4. UI of mobile client.

A second mobile device is used as a server. This server is attached to the RoboMe robot and is responsible for the movement that the user sends using the mobile client and moreover the facial recognition and the capturing of image data that will be sent to the spark is done here.

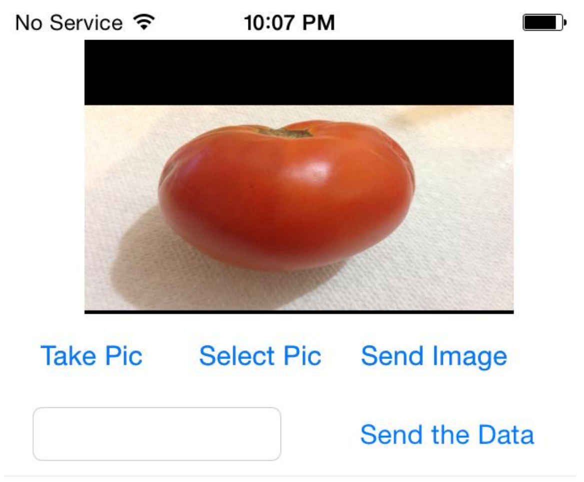


Fig 5. Shows the UI for image capturing.

Fig 6. Shows the activity diagram for the mobile client. Fig 7. Shows the sequence diagram.

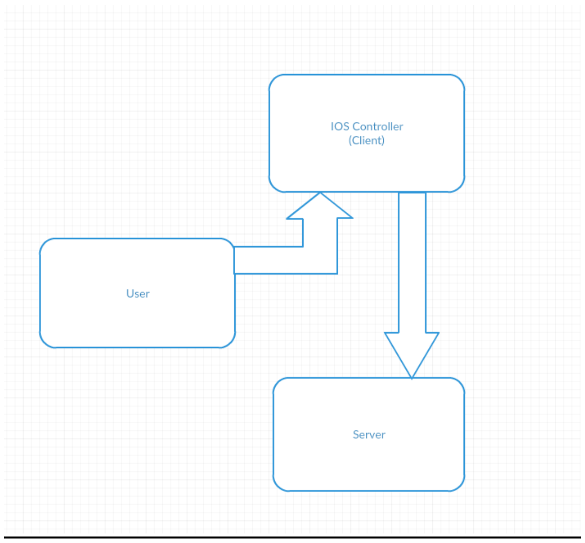


Fig 6. Activity diagram of mobile client.

So in this the mobile client takes an input from the user. As soon as the controller receives an input matching the input function that is associated to a parameter, the App delegate function in the IOS client sends it to the server to perform specific action.

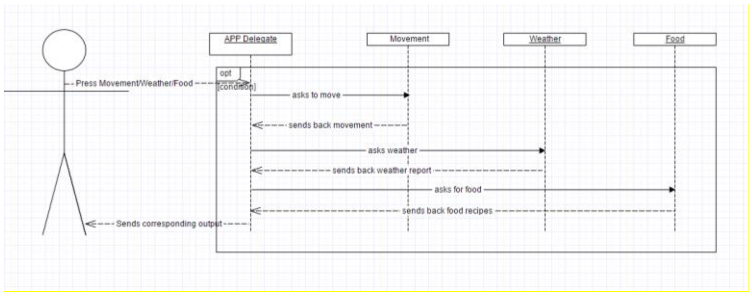


Fig 7. Sequence diagram of the mobile client

4.3 Existing open source projects/services used

For our project we used mainly three types of services each relating to a separate function.

OpenCV framework [2], is an open source computer vision framework that is popular for image processing. This framework allows us to detect key features of an image using SURF or SIFT detectors.

Yummly API [3], this is a food recipe API that we are using to get the recipes. So this API take any input related to food and gives a list of the top recipes based on user ratings provided on their website. Example, if you give eggplant as an input to the API it gives a list of the top recipes related to eggplant. You could give multiple parameters as its input also, it will filter the recipes based on these inputs. We found this API to be amongst the top rated and used API for food recipes.

Kairos Face Detection [4], is a face and emotion recognition service that we have used in our mobile client. This API service first trains a particular person’s face and then it is capable of recognizing him/her. This API also provides other features like emotion detection and crowd analytics.

Open Weather API [5], it is a simple weather forecasting service provided by openweathermap. It returns a JSON output regarding a particular location.

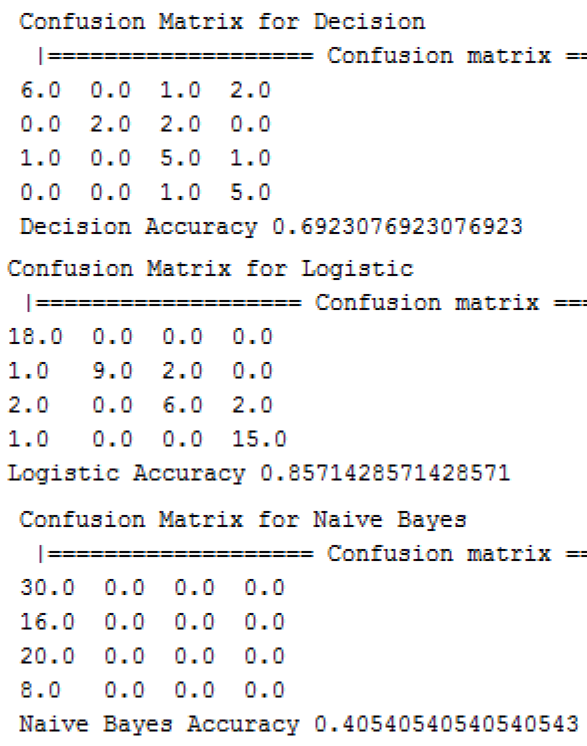
Speech to text [6], we used dragon naturally speaking SDK to add speech recognition capabilities to our project.

5. **Evaluation Results**

We have chosen a well-defined and distributed image dataset of 600 images with 8 classes for analyzing the algorithms. We have used this data set and ran each of the machine learning algorithm separately to check its precision and confusion matrix.

The precision and the confusion matrix of the three algorithms namely Logistic Regression, Naïve Bayes and Decision Tree algorithms are compared in the fig. 8. It can be clearly seen that the Logistic regression has outdone other algorithms.

Similarly we have tested for the best feature extractor method that would meet the desired speed and accuracy. We have evaluated ORB, SIFT, SURF and FAST feature extractors and FAST feature extractor seemed to be the best out of the other methods. The fig. 9 and 10 shows the key points plotted on the image. These key points are detected by these methods. It can be clearly seen that the FAST was able to identify the shape of the object and also more accurate than the other methods. Apart from accuracy, FAST was also faster than the other methods.



```
Confusion Matrix for Decision
 |=================== Confusion matrix ===============
6.0  0.0  1.0  2.0
0.0  2.0  2.0  0.0
1.0  0.0  5.0  1.0
0.0  0.0  1.0  5.0
Decision Accuracy 0.6923076923076923
Confusion Matrix for Logistic
 |=================== Confusion matrix ===============
18.0  0.0  0.0  0.0
1.0   9.0  2.0  0.0
2.0   0.0  6.0  2.0
1.0   0.0  0.0  15.0
Logistic Accuracy 0.8571428571428571
Confusion Matrix for Naive Bayes
 |=================== Confusion matrix ===============
30.0  0.0  0.0  0.0
16.0  0.0  0.0  0.0
20.0  0.0  0.0  0.0
8.0   0.0  0.0  0.0
Naive Bayes Accuracy 0.40540540540540543
```

Fig 8. Accuracy and Confusion Matrix of Logistic regression, Decision Tree learning and Naïve Bayes

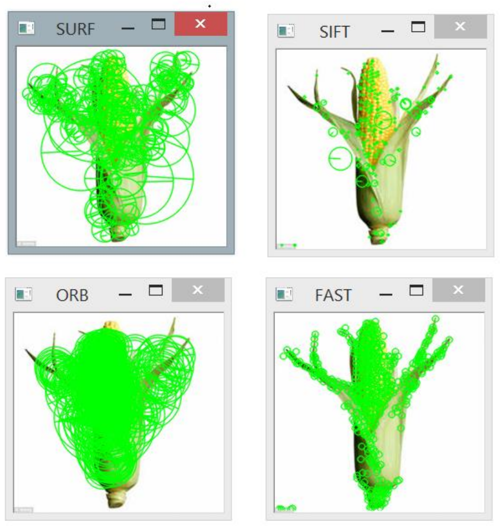


Fig 9. Key point Detection on Corn by ORB, SIFT, SURF and FAST.

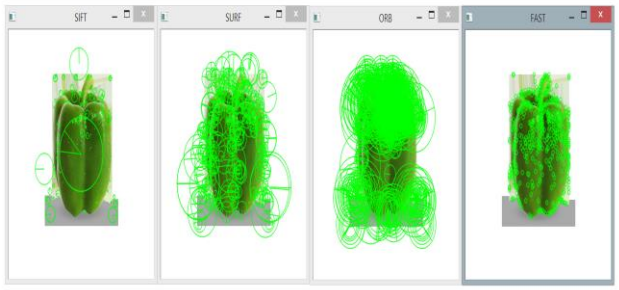
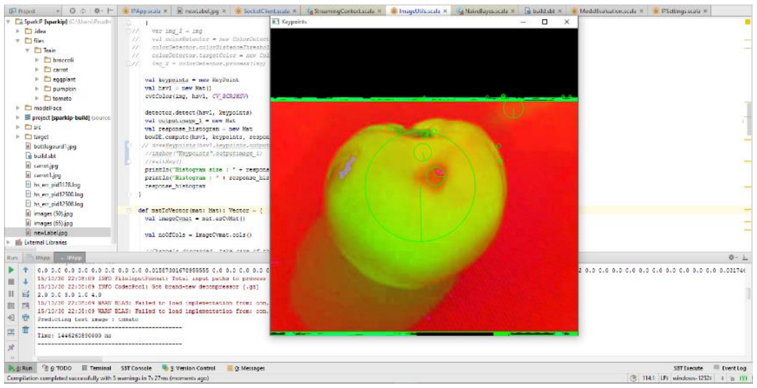


Fig 10. Key point Detection on Green Peppers by ORB, SIFT, SURF and FAST.



```
  }
//    var img_2 = img
//    val colorDetector = new ColorDetect
//    colorDetector.colorDistanceThreshol
//    colorDetector.targetColor = new Col
//    img_2 = colorDetector.process(img)

    val keypoints = new KeyPoint
    val hsv1 = new Mat()
    cvtColor(img, hsv1, CV_BGR2HSV)

    detector.detect(hsv1, keypoints)
    val outputimage_1 = new Mat
    val response_histogram = new Mat
    bowDE.compute(hsv1, keypoints, respon
   // drawKeypoints(hsv1,keypoints,output
    //imshow("Keypoints",outputimage_1)
    //waitKey()
    println("Histogram size : " + respons
    println("Histogram : " + response_his
    response_histogram
  }

  def matToVector(mat: Mat): Vector = {
    val imageCvmat = mat.asCvMat()

    val noOfCols = imageCvmat.cols()
```

```
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.015873016785555 0.0 0.0 0.0 0.0 0
15/10/30 22:08:09 INFO FileInputFormat: Total input paths to process
15/10/30 22:08:09 INFO CodecPool: Got brand-new decompressor [.gz]
2.0 0.0 3.0 1.0 4.0
15/10/30 22:08:09 WARN BLAS: Failed to load implementation from: com.
15/10/30 22:08:09 WARN BLAS: Failed to load implementation from: com.
Predicting test image : tomato
----------------------------------------
Time: 1446260850000 ms
----------------------------------------
```

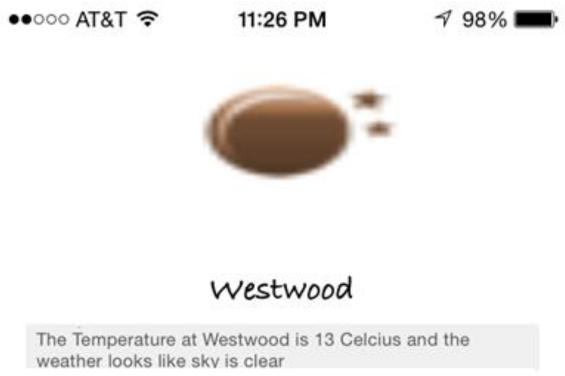
Compilation completed successfully with 5 warnings in 7s 27ms (moments ago)

Fig 12. Shows the image classification in spark in IntelliJ IDE

5. End Results

For the training set used in our image recognition we have collected most of our images using the advanced search in google images and a few other image dataset from [7].

Weather, the user queries for a particular location based on zip code and it would result in an image showing the weather and the location name.

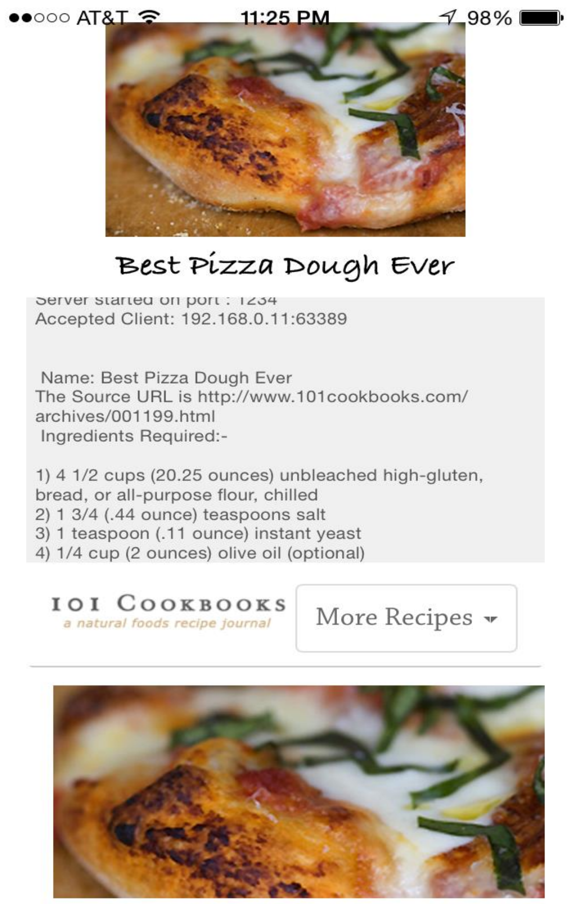


Westwood

The Temperature at Westwood is 13 Celcius and the weather looks like sky is clear

Fig.11 shows the weather results

Recipes, the user can query for recipes either by text, voice or through images. If the user selects image he has to first capture the image of the vegetable which will be sent to spark for image classification and the result will be sent to the yummly api to get recipes.



Best Pizza Dough Ever

```
Server started on port : 1234
Accepted Client: 192.168.0.11:63389


 Name: Best Pizza Dough Ever
The Source URL is http://www.101cookbooks.com/
archives/001199.html
 Ingredients Required:-

1) 4 1/2 cups (20.25 ounces) unbleached high-gluten,
bread, or all-purpose flour, chilled
2) 1 3/4 (.44 ounce) teaspoons salt
3) 1 teaspoon (.11 ounce) instant yeast
4) 1/4 cup (2 ounces) olive oil (optional)
```

Fig. 9 Shows the recipe when a user inputs through text

6. Related Work

[8], this paper shows classification of fruits using computer vision and SVM. Their classification was based on a kernel support vector machine algorithm as it was able to provide a desirable accuracy. Similar to our project they have removed the background white color of the image and calculated color histogram, texture and shape and then they used principle component analysis to reduce the feature space. But in our project we used k-means to cluster features for a specific features that are nearby.

7. Future Work

Providing a better user interface, suppose a live image recognition without capturing the image and sending it. Applying better image classification methods.

8. References

[1] http://cs229.stanford.edu/proj2011/SchmittMcCoyObjectClassificationAndLocalizationUsingSURFDescriptors.pdf

[2] http://docs.opencv.org/2.4/modules/core/doc/intro.html

[3] Http://api.yummly.com/v1/api/recipes?_app_id=appid&_app_key=appkey&your_search_parameters

[4]https://www.kairos.com/face-recognition-api

[5]https://www.kairos.com/face-recognition-api

[6]http://www.nuance.com/for-developers/dragon/client-sdk/index.htm

[7] https://www.cs.toronto.edu/~kriz/cifar.html

[8]http://www.ncbi.nlm.nih.gov/pubmed/23112727