VIVEKANAND EDUCATION SOCIETY'S
INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Project Report on

SeeFood – Food that you can see

In partial fulfillment of the Fourth Year (Semester–VIII), Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai

Academic Year 2017-2018

Project Mentor
Mrs. Priya R. L.

Submitted by
Himanshu Rawlani D17A - 64
Jayesh Saita D17A - 66
Vignesh Zambre D17A - 78

(2017-18)

VIVEKANAND EDUCATION SOCIETY'S
INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

CERTIFICATE of Approval

This is to certify that _____ of Fourth Year Computer Engineering studying under the University of Mumbai has satisfactorily presented the project on "***SeeFood - Food that you can see***" as a part of the coursework of PROJECT-II for Semester-VIII under the guidance of ***Mrs. Priya R. L.*** in the year 2017-2018.

_____	_____	_____
Date	Internal Examiner	External Examiner
_____	_____	_____
Project Mentor	Head of the Department	Principal
Priya R. L.	Dr. Mrs. Nupur Giri	Dr. J. M. Nair

ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society's Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Mrs. Priya R.L** (Project Guide) for her kind help and valuable advice during the development of project synopsis and for her guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair ,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement at several times.

Computer Engineering Department

COURSE OUTCOMES FOR B.E PROJECT

Learners will be to:-

Course Outcome	Description of the Course Outcome
CO 1	Able to apply the relevant engineering concepts, knowledge and skills towards the project.
CO2	Able to identify, formulate and interpret the various relevant research papers and to determine the problem.
CO 3	Able to apply the engineering concepts towards designing solution for the problem.
CO 4	Able to interpret the data and datasets to be utilized.
CO 5	Able to create, select and apply appropriate technologies, techniques, resources and tools for the project.
CO 6	Able to apply ethical, professional policies and principles towards societal, environmental, safety and cultural benefit.
CO 7	Able to function effectively as an individual, and as a member of a team, allocating roles with clear lines of responsibility and accountability.
CO 8	Able to write effective reports, design documents and make effective presentations.
CO 9	Able to apply engineering and management principles to the project as a team member.
CO 10	Able to apply the project domain knowledge to sharpen one's competency.
CO 11	Able to develop professional, presentational, balanced and structured approach towards project development.
CO 12	Able to adopt skills, languages, environment and platforms for creating innovative solutions for the project.

Index

Chapter 1

INTRODUCTION

1.1 Introduction to the project

People get easily tired of eating the same things again and again. They then resort to restaurant / junk food which affects their health as the vegetables and oil used by them are of suboptimal quality.

The proposed system suggests recipes according to the vegetables available to the user. User scans the available vegetables and android application displays a list of possible recipes.

The application is personalized, which means the recipes displayed are filtered according to the user's preferences and health conditions. For example, the proposed application does not display a spicy recipe to a user for elderly people. This makes our application more health friendly.

1.2 Motivation for the project

There are daily situations in our lives where we are really confused about what dish to prepare. We are tired of eating the same dishes daily. We usually need quick suggestion of what is possible to make with the available vegetables. Thus the proposed system helps in avoid such problems. It quickly displays a list of possible recipes.

Another interesting aspect of the proposed project is using Deep Learning approach. Deep Learning is currently trending in the field of Artificial Intelligence(AI) and we want to learn and use it in real life application. Also multiple object detection is one of the challenging and interesting problems in deep learning. The system not only detect vegetables properly, but also discard similar looking objects that are not vegetables.

1.3 Drawback of the existing system

- Current System only displays calorie content of the food
- Current system does not recommend whether to eat this food or not i.e it does not take user's health preferences into consideration (example: if food has high calorie content, then the user should avoid it)

1.4 Problem Definition

The main challenge in the proposed system is to perform multiple object detection. It is cumbersome to click one photo of a vegetable/fruit at a time. The multiple vegetables can be in multiple environments (inside refrigerator, placed on kitchen platform etc)

After detecting the vegetables, the system helps to generate a list of recipes using those detected vegetables. These recipes are filtered according to the user's health preferences.

All of the above things should be done quickly. User should not wait for a long amount of time to get the results.

1.5 Relevance of the project

Our application is for any person who wants to try out new recipes. The main advantage is that the person does not need to buy any additional ingredients. All the suggested recipes are possible using the available ingredients.

1.6 Methodology used

The proposed system uses supervised learning for training the network with images of vegetables and fruits.The actual algorithm for recognition will be based on convolutional networks which includes Tiny YOLO. Among R-CNN,Fast R-CNN, Faster R-CNN and Tiny YOLO the requirement of speedy detection was satisfied by Tiny YOLO model. Prototyping methodology is used.

Initially, we created a prototype which only provided a list of detected fruits and vegetables. Along with this, we created a prototype for fetching recipes. Finally, we integrated these into one application and focused on increasing the accuracy of prediction by increasing the number of epochs.

Chapter 2

LITERATURE SURVEY

2.1 Research Papers

1. Histograms of Oriented Gradients for Human Detection (Jun 2005) [1]

Abstract:

Studying the question of feature sets for robust visual object recognition, by adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient based descriptors, results show experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. The influence of each stage of the computation on performance is studied, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database.

Inference:

The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. Idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. Using locally normalized histogram of gradient orientations features similar to SIFT descriptors in a dense overlapping grid gives very good results for person detection, reducing false positive rates by more than an order of magnitude relative to the best Haar wavelet based detector. First in feature extraction HOG uses Sobel filters which are either diagonal or rectangular depending on the type of edges in the pictures.

2. ImageNet Classification with Deep Convolutional Neural Networks - AlexNet (Nov 2012) [2]

Abstract :

Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high dimensional patterns such as handwritten characters, with minimal preprocessing. Convolutional neural networks that are specifically designed deal with the variability of 2d shapes, are shown to outperform all other techniques. Real-life document recognition systems are composed of multiple modules including field extraction, segmentation,

recognition, and language modeling. A new learning paradigm, called Graph Transformation Networks (GTN), allows such multi-module systems to be trained globally using Gradient based methods so as to minimize an overall performance measure.

Inference:

AlexNet uses 8 main layers in which the first five layers are the convolutional layers and the last three layers are fully connected layers. There are certain internal layers that include pooling and activation functions. Pooling decreases the size of input(i.e. Output from previous layer) and activation converts the net inputs to output of a neural node and adds nonlinearity. Since the computation are heavy in this neural network, it is divided into two separate streams and are executed parallely. The first layer takes input of 224x224x3 image size and number of filters size is 96 with filter size 11x11x3. The compressed output 55x55x48 per GPU. The activation unit used here is Rectified Linear Unit (RELU). The next 2 to 5 layers uses sub sampling window of 2x2 and the max pooling is the max of 4 values. This is done to reduce the number of computation and controls overfitting. The sixth layer is fully connected layer with input matrix to be 13x13x128. It uses graph transformation network. It is a acyclic graph in which every edge depicts certain amount of pixels is segmented by selecting the internal nodes. This marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4%.

3.Real Time object detection - SVM (2012) [13]

Abstract :

In this paper, a mobile cooking recipe recommendation system employing object recognition for food ingredients such as vegetables and meats is proposed. The proposed system carries out object recognition on food ingredients in a real-time way on an Android-based smartphone, and recommends cooking recipes related to the recognized food ingredients. By only pointing a built-in camera on a mobile device to food ingredients, the user can obtain a recipe list instantly. As an object recognition method, we adopt bag-of-features with SURF and color histogram extracted from multiple images as image features and linear SVM with the one-vs-rest strategy as a classifier.

Inference :

They used bag-of-features and SURF with color histogram extracted from multiple images as image features. Along with this, linear SVM with one-vs-rest strategy was used a classifier. They achieved a recognition rate of 83.93%.

3. Visualizing and Understanding Convolutional Networks - ZF Net (2013) [3]

Abstract:

Large Convolutional Network models have recently demonstrated impressive classification performance on the ImageNet benchmark (Krizhevsky et al., 2012). However there is no clear understanding of why they perform so well, or how they might be improved. In this paper we address both issues. We introduce a novel visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier. Used in a diagnostic role, these visualizations allow us to find model architectures that outperform Krizhevsky et al. on the ImageNet classification benchmark. We also perform an ablation study to discover the performance contribution from different model layers. We show our ImageNet model generalizes well to other datasets: when the softmax classifier is retrained, it convincingly beats the current state-of-the-art results on Caltech-101 and Caltech-256 datasets.

Inference:

ZF Net is very similar architecture to AlexNet, except for a few minor modifications. AlexNet trained on 15 million images, while ZF Net trained on only 1.3 million images. Instead of using 11x11 sized filters in the first layer (which is what AlexNet implemented), ZF Net used filters of size 7x7 and a decreased stride value. The reasoning behind this modification is that a smaller filter size in the first conv layer helps retain a lot of original pixel information in the input volume. A filtering of size 11x11 proved to be skipping a lot of relevant information, especially as this is the first conv layer. As the network grows, we also see a rise in the number of filters used. Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent. Developed a visualization technique named Deconvolutional Network, which helps to examine different feature activations and their relation to the input space. Called “deconvnet” because it maps features to pixels (the opposite of what a convolutional layer does).

4. Very Deep Convolutional Networks For Large-Scale Image Recognition - VGGNet (2014) [4]

Abstract:

In this work the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting is investigated. The main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3 × 3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet

Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. This paper also shows that the representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

Inference:

The use of only 3x3 sized filters is quite different from AlexNet's 11x11 filters in the first layer and ZF Net's 7x7 filters. The authors' reasoning is that the combination of two 3x3 conv layers has an effective receptive field of 5x5. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes. One of the benefits is a decrease in the number of parameters. Also, with two conv layers, we're able to use two ReLU layers instead of one. 3 conv layers back to back have an effective receptive field of 7x7. As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network. Interesting to notice that the number of filters doubles after each maxpool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth. Worked well on both image classification and localization tasks. The authors used a form of localization as regression (see page 10 of the paper for all details). Built model with the Caffe toolbox. Used scale jittering as one data augmentation technique during training. Used ReLU layers after each conv layer and trained with batch gradient descent. Trained on 4 Nvidia Titan Black GPUs for two to three weeks.

5. Rich feature hierarchies for accurate object detection and semantic segmentation - RCNN (Oct 2014) [5]

Abstract:

Object detection performance, as measured on the canonical PASCAL VOC dataset, has plateaued in the last few years. The best-performing methods are complex ensemble systems that typically combine multiple low-level image features with high-level context. In this paper, a simple and scalable detection algorithm is proposed that improves mean average precision (mAP) by more than 30% relative to the previous best result on VOC 2012—achieving a mAP of 53.3%. Our approach combines two key insights:

(1) one can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects and

(2) when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.

Since we combine region proposals with CNNs, we call our method R-CNN: Regions with CNN features. We also compare R-CNN to OverFeat, a recently proposed sliding-window detector based on a similar CNN architecture. We find that R-CNN outperforms OverFeat by a large margin on the 200-class ILSVRC2013 detection dataset.

Inference:

The last decade of progress on various visual recognition tasks has been based considerably on the use of SIFT and HOG, but performance on canonical visual recognition task like the PASCAL VOC dataset has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods. RCNN object detection system consists of 3 modules: The first generates category independent regional proposals, the second module is large convolution neural network that extracts a fixed length feature vector from each regions and the third module is a set of class specific linear SVMs. R-CNN achieves a mAP of 31.4%, which is significantly ahead of the second-best result of 4.3% from OverFeat.

6. Deep Residual Networks for Image Recognition (Dec 2015) [6]

Abstract:

Deeper neural networks are more difficult to train. This paper presents a residual learning framework to ease the training of networks that are substantially deeper than those used previously. This paper explicitly reformulates the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. This paper provides comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers. The depth of representations is of central importance for many visual recognition tasks. Solely due to the extremely deep representations, this method obtains a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

Inference:

In the previous deep networks,the main keystone of success was backpropagation but as the number of layers increases the accuracy gets saturated and error is not propagated.Hence,the higher the number of layers the more degradation in accuracy.This degradation mostly occurs due to vanishing and exploding gradient problems. ResNets tweak the mathematical formula for a deep neural network. They tweak the layers' equations to introduce identity connections between them. The identity function is simply id(x) = x; given an input x it returns the same value x as output. A layer in a traditional neural network learns to calculate a function y = f(x). A residual neural network layer approximately calculates y = f(x) + id(x) = f(x) + x. Identity connections enable the layers to learn incremental, or residual, representations. The layers can start as the identity function and gradually transform to be more complex. Such evolution occurs if the parameters for the f(x) part begin at or near zero.Thus,the degradation,if found, is stopped by the input which was fed to the previous layers.This network provides a way to increase the number of layers without degrading the accuracy.

7. Fast R-CNN (Sep 2015) [7]

Abstract:

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9x faster than R-CNN, is 213x faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3x faster, tests 10x faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License.

Inference:

Improvements were made to the original model because of 3 main problems. Training took multiple stages (ConvNets to SVMs to bounding box regressors), was computationally expensive, and was extremely slow (RCNN took 53 seconds per image). Fast R-CNN was able to solve the problem of speed by basically sharing computation of the conv layers between different proposals and swapping the order of generating region proposals and running the CNN. In this model, the image is first fed through a ConvNet, features of the region proposals are obtained from the last feature map of the ConvNet, and lastly we have our fully connected layers as well as our regression and classification heads.

8. Faster R-CNN (Jan 2016) [8]

Abstract:

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet and Fast R-CNN have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. Further RPN and Fast R-CNN are merged into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. This detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets.

Inference:

Faster R-CNN combats the somewhat complex training pipeline that both R-CNN and Fast R-CNN exhibited. The authors insert a region proposal network (RPN) after the last convolutional layer. This network is able to just look at the last convolutional feature map and produce region proposals from that. From that stage, the same pipeline as R-CNN is used (ROI pooling, FC, and then classification and regression heads). Being able to determine that a specific object is in an image is one thing, but being able to determine that object's exact location is a huge jump in knowledge for the computer..

9. YOLOv2 - You Only Look Once(2016)[9]

Abstract:

Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict

false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Inference :

YOLO uses bounding box predictor for getting the coordinates of object for classification.Unlike the RCNN models which use RPN, which require multiple passes on the image, the BBox predictor divides the image into grids and generates the boxes in one go, thus, the name ‘only look once’. Also, what we have implemented requires real time detection, which has to be fast, which is not possible in RCNN as these are very slow. As YOLO works with image only once it is substantially faster than all models of RCNN.

Chapter 3

REQUIREMENTS OF PROPOSED SYSTEM

3.1 Functional Requirements

3.1.a Register and Login

The user has to login in order to use the preferences section as the system would be storing those using firebase. But, user can also use the app without logging in, consequently, missing the preferences functionality.

3.1.b Scan vegetables and fruits

User can scan the fruits and vegetables, the YOLO detector will carry out detection, and the list will be displayed to the user. These detected fruits and vegetables will be used to give possible recipes.

3.1.c Filtering Recipes

All the possible recipes will not be displayed to the user. All the possible recipes are filtered according to information provided by user (age, weight, preferences etc) and finally list of filtered recipes is displayed to the user.

3.1.d Open and read recipe

From the list of filtered dishes, user can select any one of the dishes to read the recipe.

3.1.e Share the recipe

If user likes a particular dish, he/she can share it with his/her friends. The recipe can be shared on various platforms like facebook, twitter, whatsapp etc

3.1.f Add recipe to favorites

User has the option to mark a dish as favorite. The consequent recipe will be saved offline on the user's phone and the user can see it any time.

3.1.g History

User can see the previously viewed recipes in the current session. These are not stored persistently. This feature is made available so that the user does not has to click the picture of the vegetables again to see previously viewed recipe.

3.2 Non Functional Requirements

3.2.a Simplistic User Interface

The target audience for this app is all age groups. To help old aged people easily use the app, the app should have simplistic user interface. The UI should not be cluttered and the meaning of various icons should be understandable at first glance.

3.2.b Quick retrieval of recipes

Given the user has proper network connections, the recipes should be retrieved quickly. User should not wait for a long time to see the list of recipes.

3.2.c Matching correct recipes with scanned ingredients

Proper matching of recipes and vegetables should be done.User would be displayed dishes which have the best overlap of ingredients, it contains, with the list of vegetables detected. Its is also possible that there are no such dishes which match the given list of vegetables, in this case, the dish list is sorted according to highest number of common ingredients to the lowest number.

3.2.d Dataset collection

Available images and their annotations were downloaded from ImageNet and Google's Open Images Dataset. Google's Open Images dataset[14] has a relational database which mapped annotations to corresponding image urls and many other mappings such as test, training and validation sets. Python script was used to scrape the required images.

3.3 Constraints

- Cost - The cost of GPUs used for training the model is high (12k - 13k).
- The cost of training models on Cloud based GPUs is similar.
- Android phone with moderate specs (2-3 GB of RAM) is required for good performance.

3.4 Hardware and Software Requirements

3.4.a Hardware

- Smartphone

This the platform on which the application is to be deployed , thus, is important for debugging and testing.

- Nvidia GPU

Training of Deep neural networks is computationally intensive; this calculation, if done on a CPU, would take a large amount to train. Also, the tensorflow framework is optimized for calculation on GPUs and it takes less time to train on such processing units.

3.4.b Software

- Ubuntu OS

  This is the main application development environment.
- Android OS

3.5 Techniques utilised for the proposed system

- **Object Detection**

One of the main functionality is to detect all the raw fruits and vegetables provided, which requires a multiple object detector. Thus, instead of using a separate detector the YOLO has incorporated the object predictor in its model.

- **Bounding Box regression**

This regression is used for determining whether the enclosed box contains an object or not.

- **Fully Connected classification network**

The network is trained using 13 classes of fruits and vegetables. The classification layer is also incorporated in the YOLO model and are represented by the last three convolutional layers.

- **Tiny YOLO**

The main YOLO model is too heavy for deploying on a mobile or a handheld device as it lacks the required processing power. Thus, a smaller version which would have lesser layers but it would be much faster, is used**.**

3.6 Tools utilised for the proposed system

3.6.a API

- Tensorflow

- Matplotlib
- Numpy

3.6.b Software and IDE

- Google Colaboratory
- Sublime Text
- Android Studio

3.7 Algorithms used in existing system

- Histogram of Oriented Gradients
- Neural Networks
- Support Vector Machines
- Random Forest and Gradient Boosted Trees
- Logistic regression

Chapter 4

PROPOSED DESIGN

4.1 Conceptual Diagram

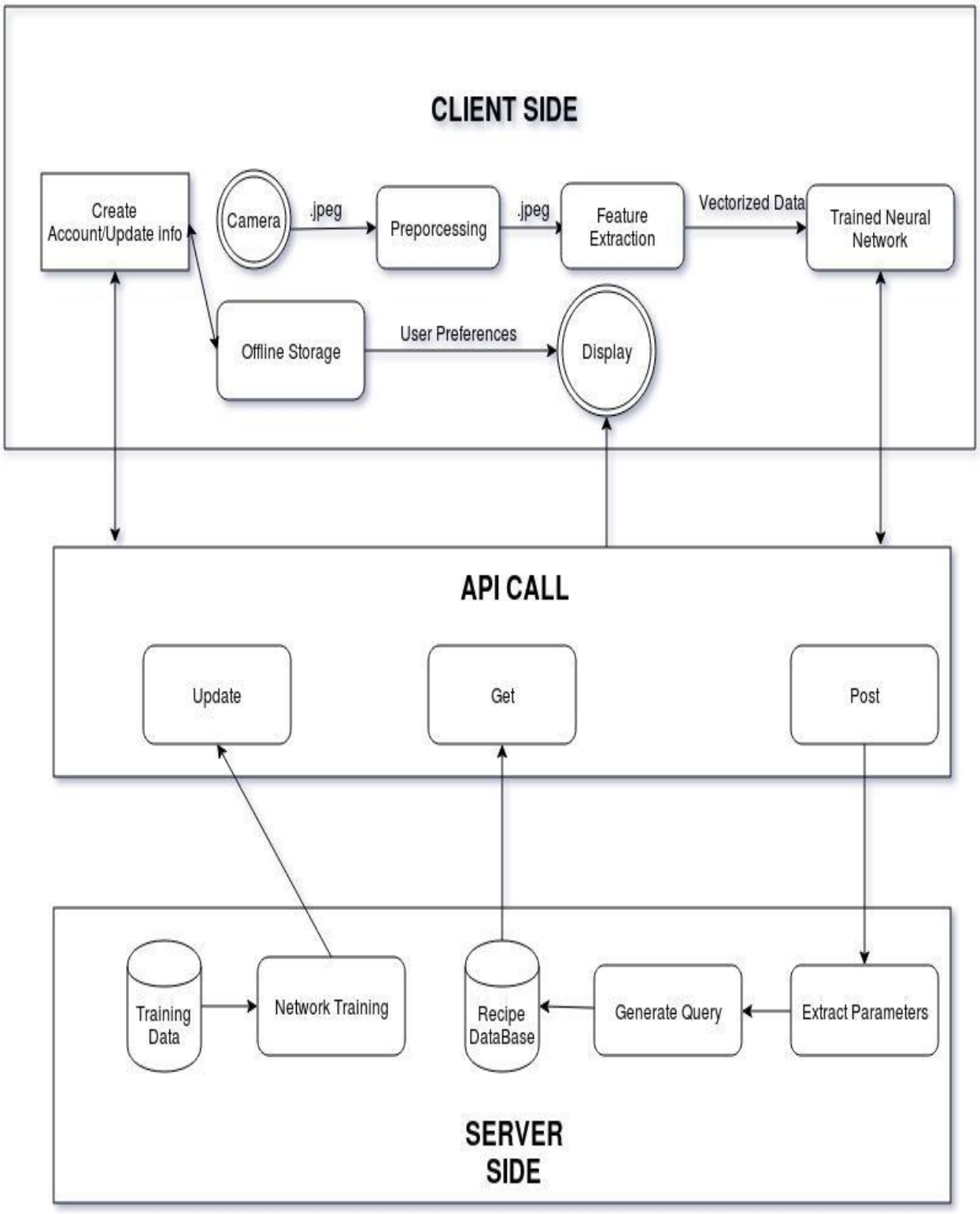


Fig.1 Conceptual Diagram

4.2 Block Diagram

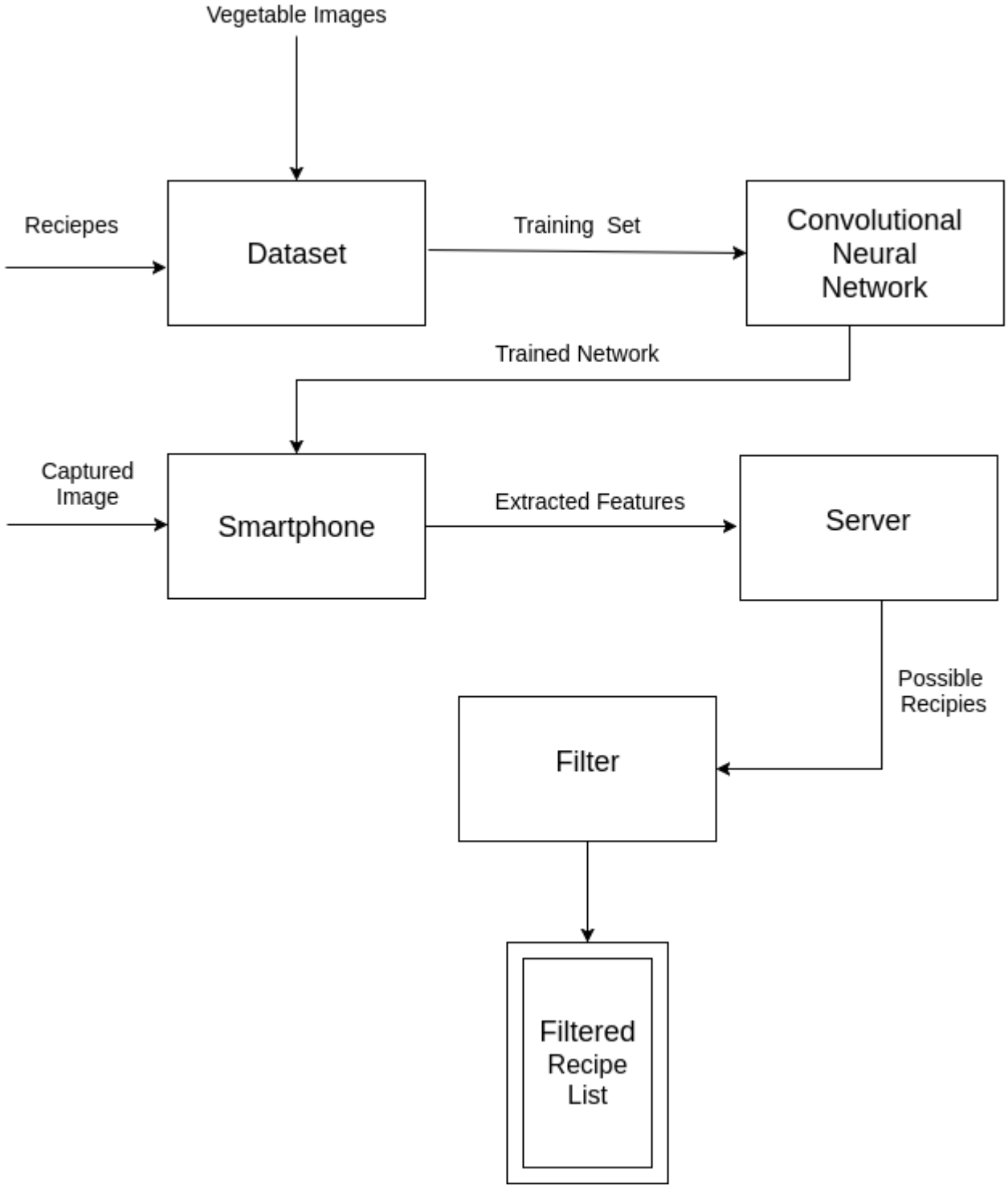


Fig.2 Block Diagram

4.1.a Training Module :

Dataset :[14]

This includes the images as well as annotations of vegetables and fruits that the neural network has to be trained on. Also, the neural networks requires certain formatting in the annotations file, this is also done in this block.

Tiny YOLO:

The dataset is used for training this model upto certain number of epochs.The training is done, parallely, on a GPU provided for free by Google via the collaboratory.

4.1.b Image Prediction :

Smart Phone :

The trained Tiny YOLO model will be stored along with the Android application. This provides offline fruits and vegetables detection, also, the model, which is .pb or a protobuf file, has a size around 60-80 mb, which is pretty minimalistic.

The camera is used to scan the fruits and vegetables for detection. The list generated can be edited by marking or unmarking the check boxes for the available fruits and vegetables.All the detection is done parallely, thus, is faster.

Server :

The Yummly's API is used for searching the recipes corresponding to the detected vegetables. The system append the list of detected items to the search query, which then sent to the Yummly's database for querying. After the query has been processed the server replies with the list of dishes along with various other parameters.

4.1.c Filters :

Filtered Recipe list :

When the user logs in he/she can provide his/her preferences as to what type of food he/she does not like or does not prefer. Thus, this module filters the dishes list, provided by the API, and returns a list which is user friendly.

4.3 Modular Diagram

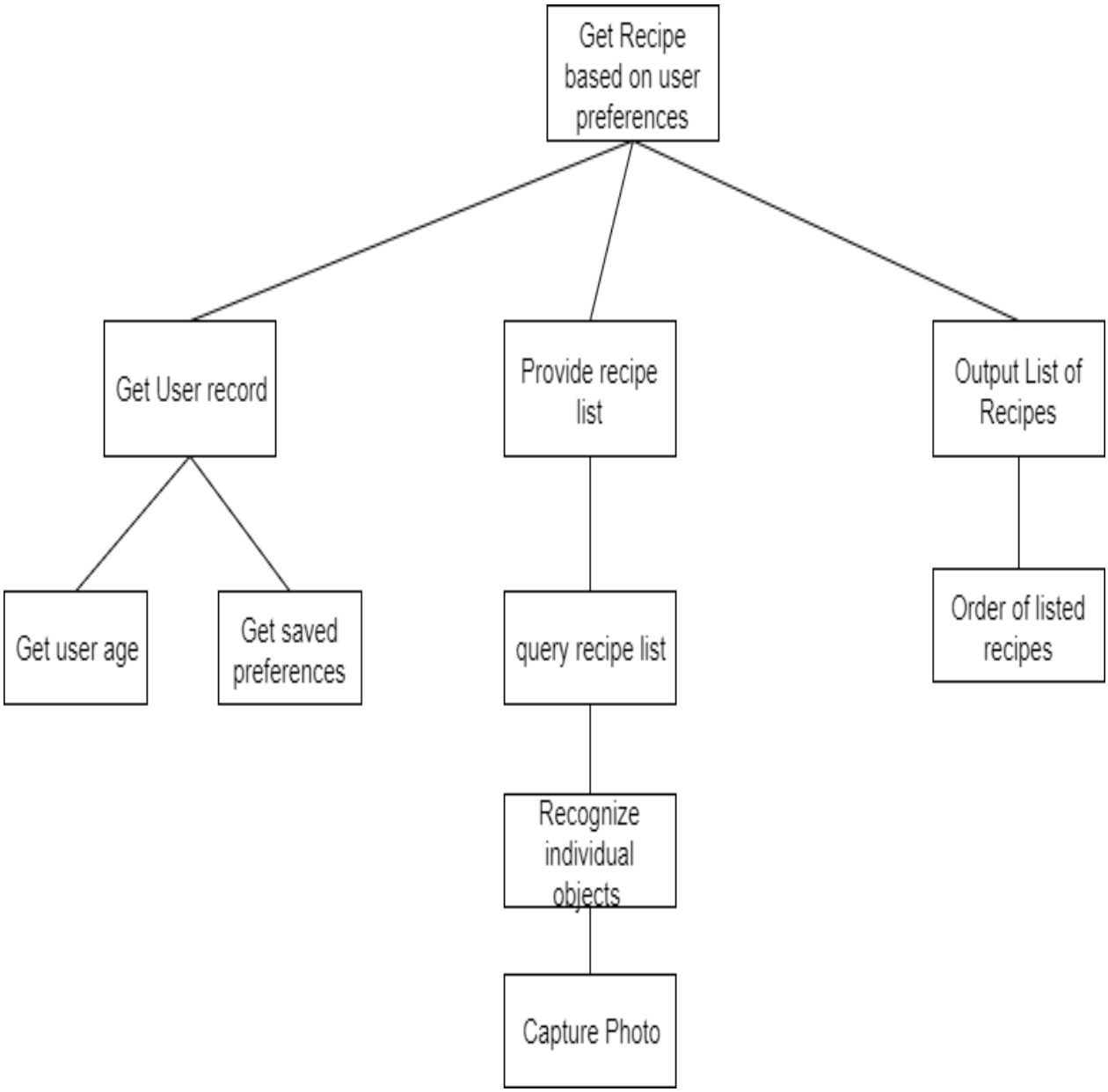


Fig.3 Modular Diagram

4.4 Design of System

4.4.a Data Flow Diagram

DFD Level 0:

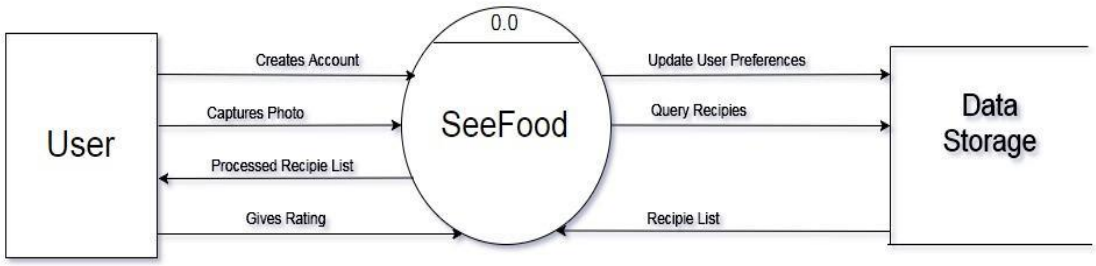


Fig.4 DFD level 0

DFD Level 1:

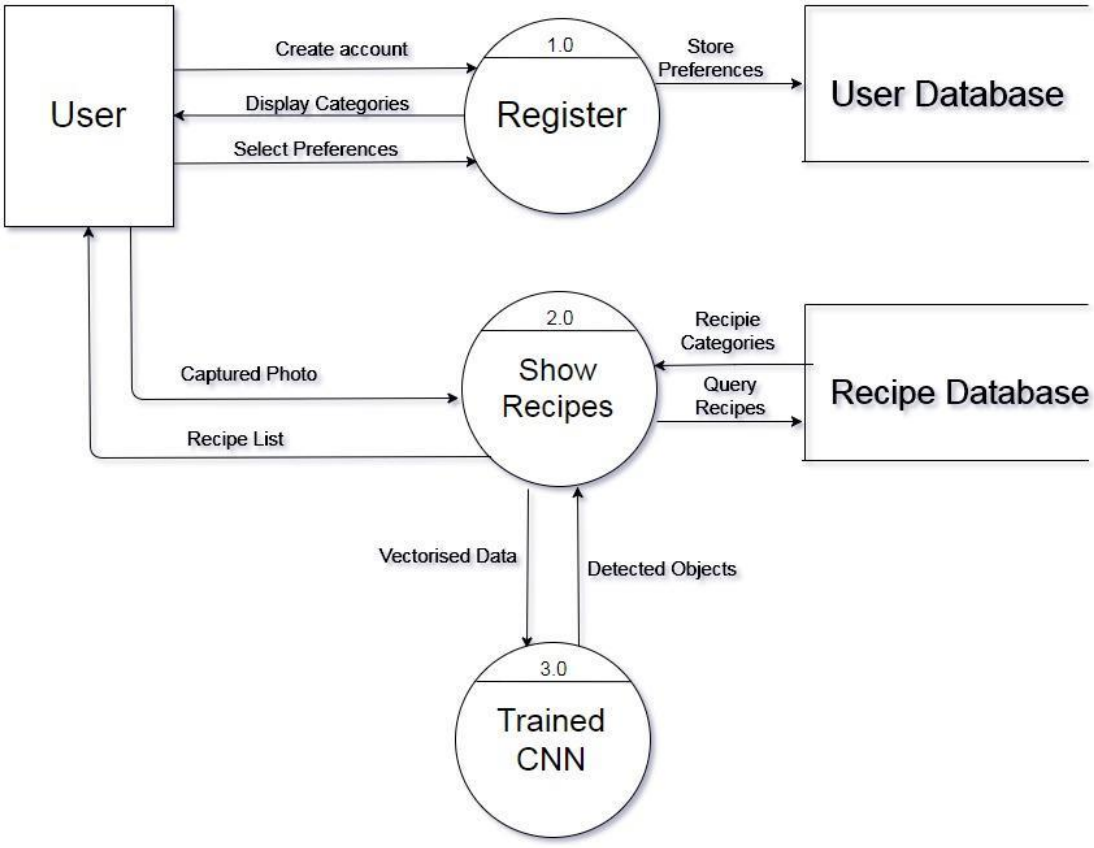


Fig.5 DFD level 1

DFD Level 2:

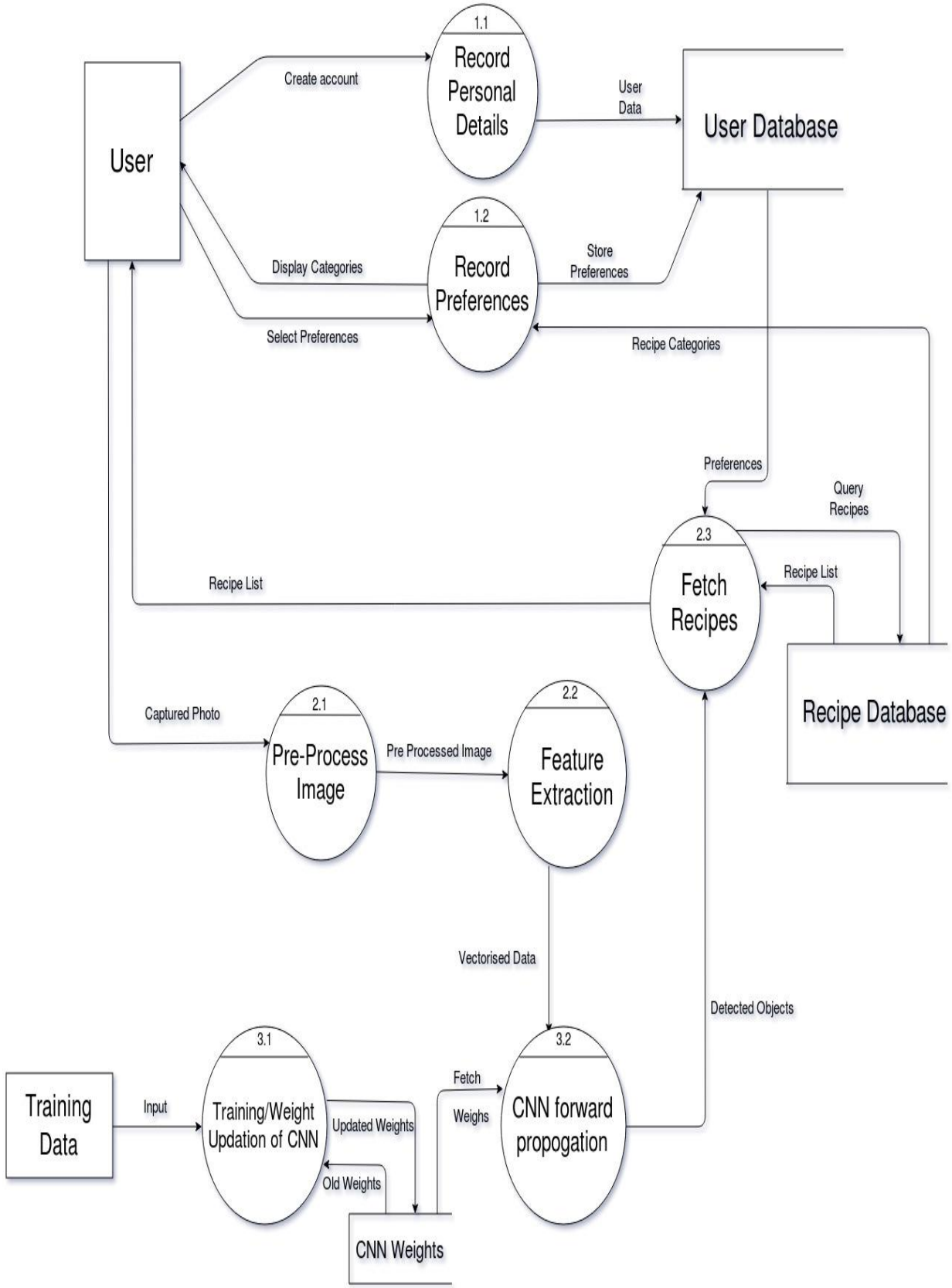


Fig.6 DFD level 2

4.4.b System Architecture Diagram

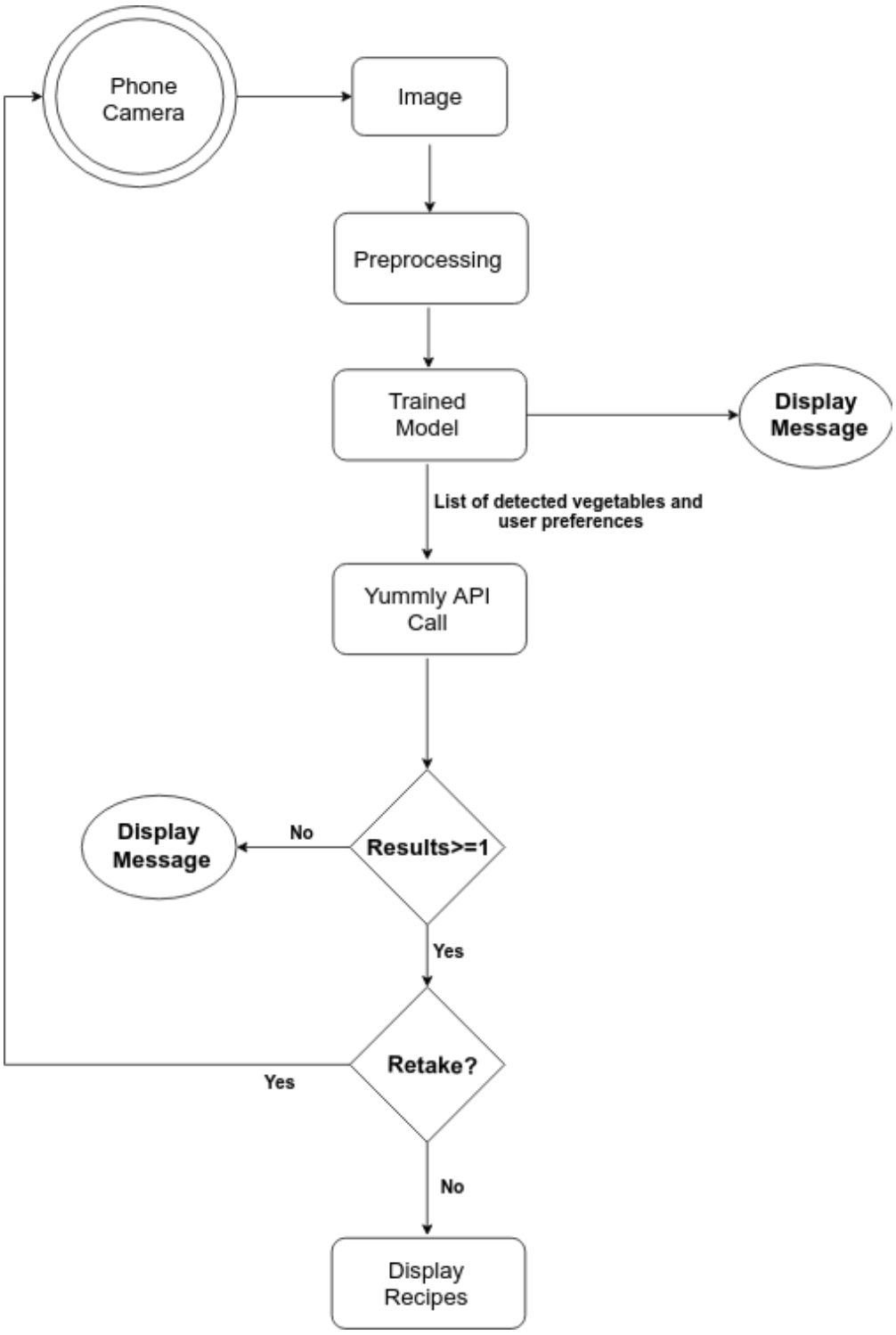


Fig.7 System Architecture Diagram

4.4.c Activity Diagram

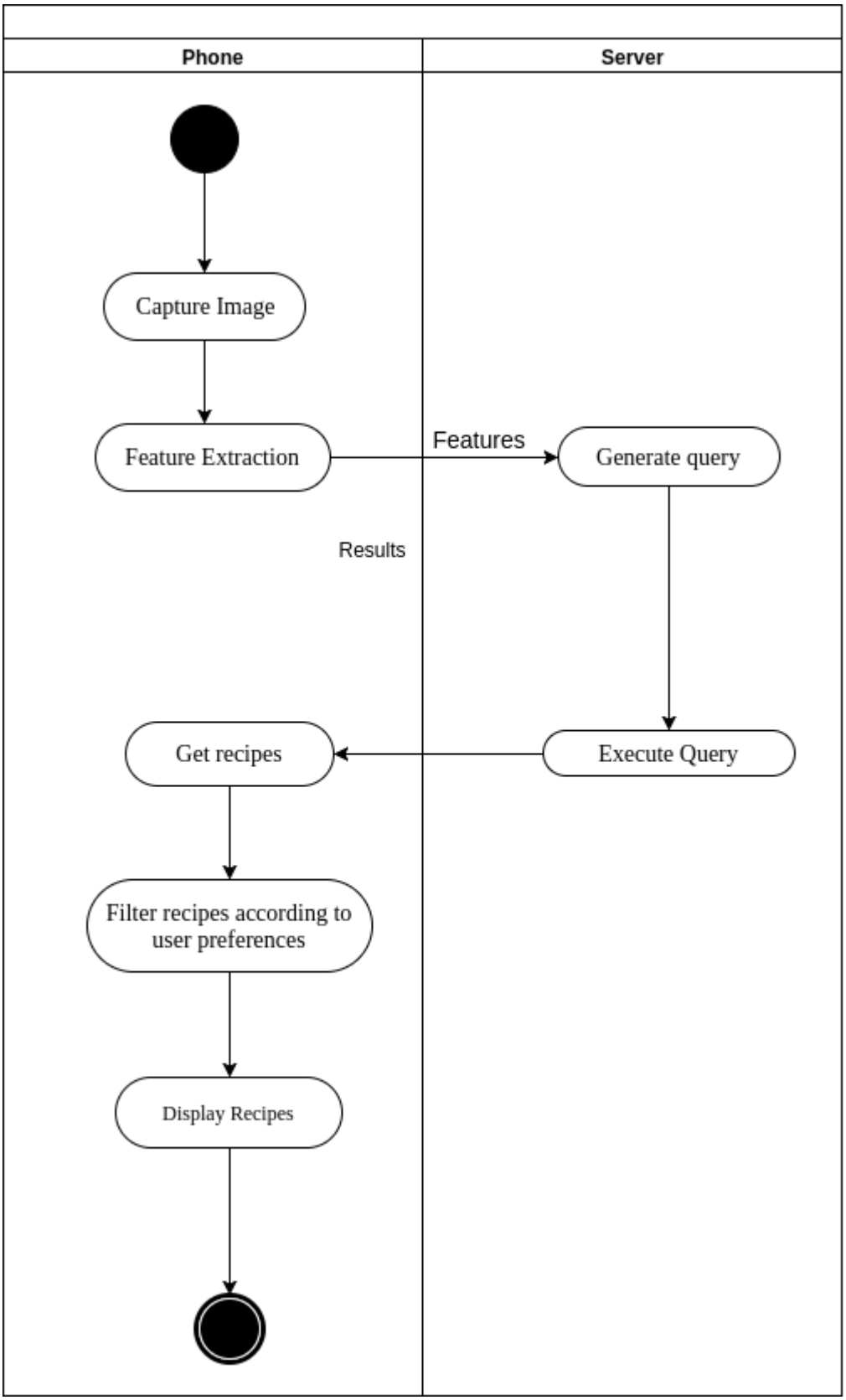


Fig.8 Activity diagram

4.4.d ER Diagram

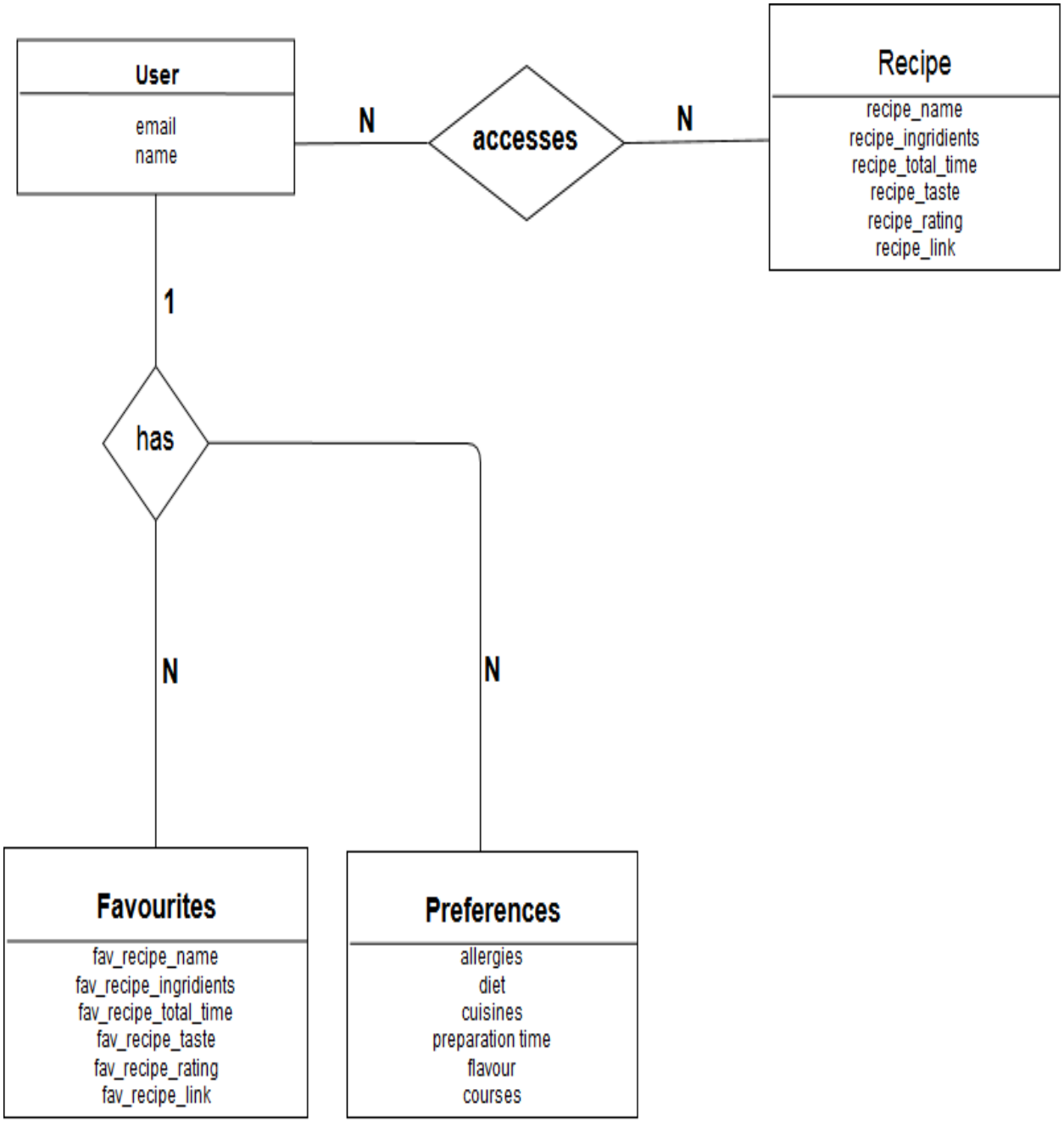


Fig 9: ER Diagram

4.4.e Gantt Chart

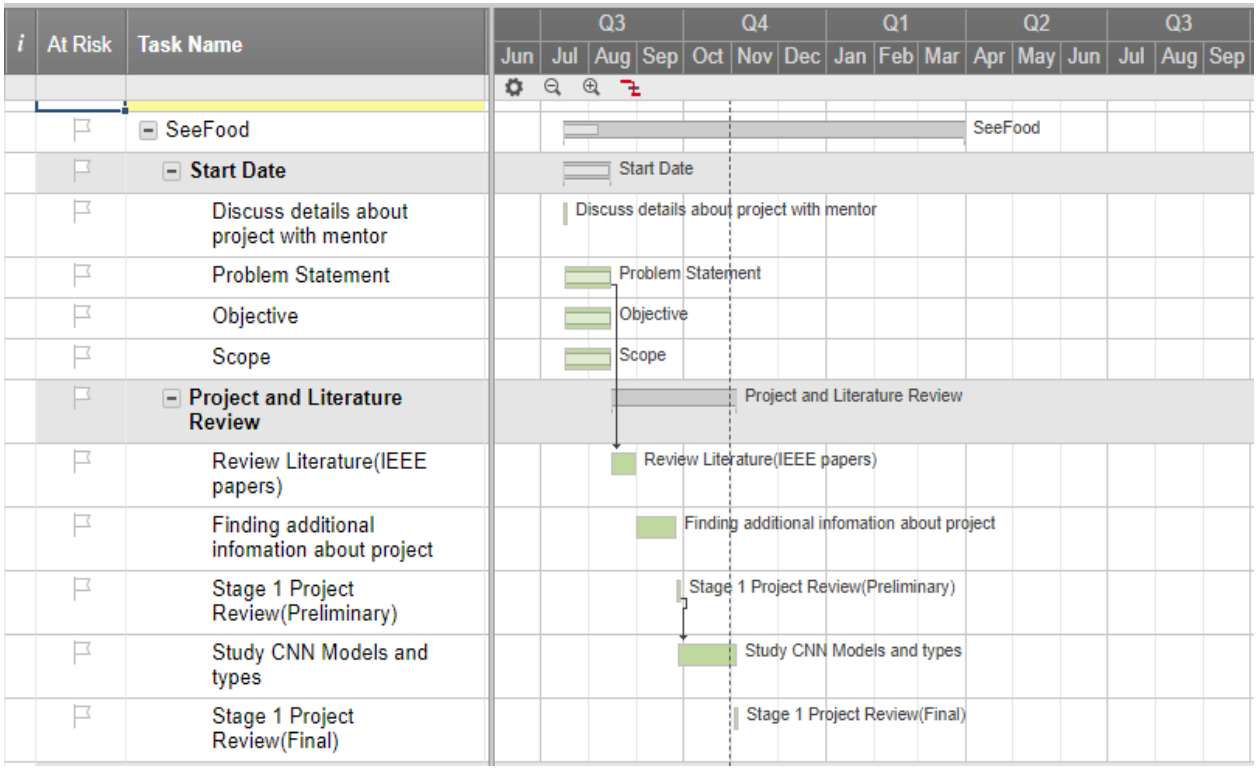


Fig.10.a Gantt Chart (1)

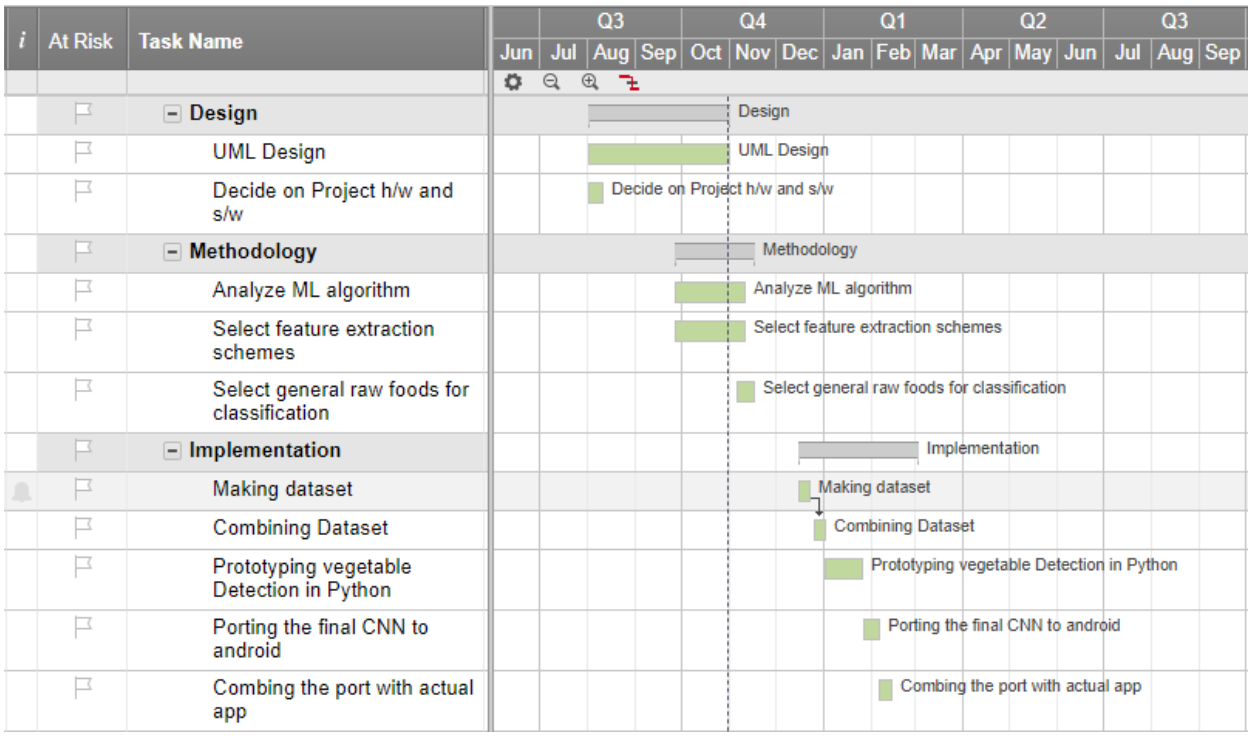


Fig.10.b Gantt Chart (2)

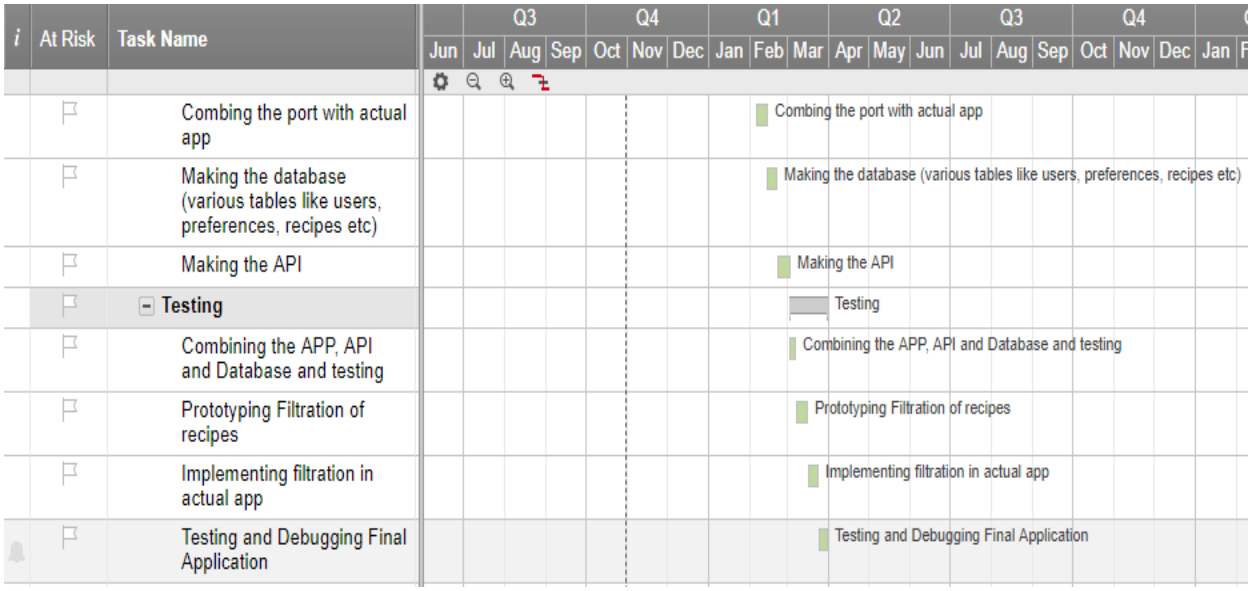


Fig.10.c Gantt Chart (3)

Chapter 5

IMPLEMENTATION DETAILS

5.1 Tiny YOLO

Tiny YOLO, is a smaller version of YOLO, which has 15 layers over the 24 layers of the original YOLO.

Transfer learning has been performed on Tiny-YOLO model, which is pre-trained on COCO dataset as the base model, on our own dataset[14] of vegetables and fruits which has 13 different classes.

Originally, Tiny-YOLO is written using DarkNet which is written in C++ (by the author). Here we have used DarkFlow which is a port of this C++ code to Tensorflow. Thus, instead of DarkNet we have used the DarkFlow library.

Google Colaboratory has been used to train the model. It is an online Jupyter notebook environment that provides free GPU (Tesla K80) processing for 12 hrs at a stretch.

For loading the images, annotations and script, a github repository is created to store all the required files. Images with their annotations have been downloaded from ImageNet and Google's Open Images dataset. The annotations were already generated and system does not annotate any image.

The repository is cloned inside Google collab and trained the model for 150 epochs. After training, the .pb file is generated for weights and used it for offline prediction on the android application

Below is the description of Tiny YOLO model used in the system.

Layer	Kernel	Stride	Shape
Input Convolution	3×3	1	416,416,3
MaxPooling	2×2	2	416,416,16
Convolution	3×3	1	208,208,32
MaxPooling	2×2	2	104,104,32
Convolution	3×3	1	104,104,64
MaxPooling	2×2	2	52,52,64
Convolution	3×3	1	52,52,128
MaxPooling	2×2	2	26,26,128
Convolution	3×3	1	26,26,256

MaxPooling	2×2	2	13,13,256
Convolution	3×3	1	13,13,512
MaxPooling	2×2	1	13,13,512
Convolution	3×3	1	13,13,1024
Convolution	3×3	1	13,13,1024
Convolution	1×1	1	13,13,90

Table. 1 : Tiny YOLO model

This algorithm is divided into two main parts, those are :

a) Object detector/Predictor
b) Object Classifier

The Object detector predicts the bounding boxes ((x,y) coordinates of the corners of a box of an predicted object) of possible objects and creates the list. This list of coordinates is used by the classifier to grab the area, bounded by those coordinates on the downscaled image, to perform classification into defined classes.

Tiny YOLO divides up the image into a grid of 13 by 13 cells; Each of these cells is responsible for predicting 5 bounding boxes. A bounding box describes the rectangle that encloses an object. YOLO also outputs a *confidence score* that tells us how certain it is that the predicted bounding box actually encloses some object. This score doesn't say anything about what kind of object is in the box, just if the shape of the box is any good. One predictor is assigned to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. IOU is intersection over union, which means how much portion of predicted box intersects with the ground truth.

The value of IOU over which the predicted object is considered as an actual object is called as a threshold. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall. At the i.

The last three convolutional layers are nothing but fully connected layers as used in classification. The last layer has 13*13*90 as matrix parameters, where 13*13 represents the grids in which the image was cut into and '90' denotes the output set which contains data regarding 5 bounding boxes for 18 data items which are :

1) 4 - bounding box coordinates
2) 1 - confidence score
3) 13 - defined classes

The model is pretrained on MS COCO dataset and is trained on fruits and vegetables dataset for 150 epochs. Throughout the training batch size is taken to be 32, learning rate as and 1e-5 Adam optimizer for convergence on a dataset of 4823 images.

YOLO model hyperparameters

- H = 13
- W = 13
- Box = 5
- Classes = 13
- Scales = [1.0, 5.0, 1.0, 1.0]

Training hyperparameters

- Learning rate = 1e-05
- Batch size = 32
- Epoch number = 150
- Optimiser = Adam
- Dataset size = 4823 images [14]
- GPU = Tesla K80

5.2 Comparative Analysis of Algorithms

Existing System	SeeFood
In[7], [8] For bounding boxes selective search is used which takes 2 and 0.2 secs per image respectively.	For bounding boxes, bounding box regression is used which takes 25 milliseconds per image
In[7], [8] Training is divided into two parts which increases the training time	Training is carried out in one single session, thus, is faster.
In[12] Sliding window is used for traversing the	The image is divided in 13* 13 grid and 5

input image	bounding boxes are predicted for each grid cell
Yummly API takes text as input thus cumbersome for user to input the vegetable names if list is long	The user only has to scan the vegetables,the system performs detection all by itself
The error rates of [7], [8] are 33.1%	The error rate is 10.73%

Table 2: Comparative Analysis of Algorithms

5.3 Determination of Efficiency, Accuracy, Effectiveness

The running time of other object detection system: fast RCNN scanning takes 1-2 seconds depending on how objects are placed (on average about 1.5s), and Faster R-CNN with VGG-16 takes 220ms on 2000 proposals (or 223ms if using SVD on fully-connected layers [2]). This system with Tiny YOLO takes in total 25ms for both bounding box regression and detection. With the convolutional features shared, the bounding box alone only takes 10ms computing the additional layers.

Detection results of Tiny YOLO on MS COCO test set using different settings of anchors: Training data contains 4823 images with the default setting of using 3 scales and 3 aspect ratios results in 89.17% mAP. By default 3 scales and 3 aspect ratios are used. Using just 3 scales with 1 aspect ratio (89.17%) is as good as using 3 scales with 3 aspect ratios on this dataset, suggesting that scales and aspect ratios are not disentangled dimensions for the detection accuracy.

Algorithm	**Test time per image(secs)**	**Speed Up**	**mAP (VOC)**
R-CNN [6]	50	1x	66%
Fast R-CNN [7]	2	25x	66.9%
Faster R-CNN [8]	0.2	250x	66.9%
YOLOv2 [9]	0.025	2000x	78.6%
Tiny YOLO	0.025	2000x	89.17%

Table 3.a : Algorithm Analysis

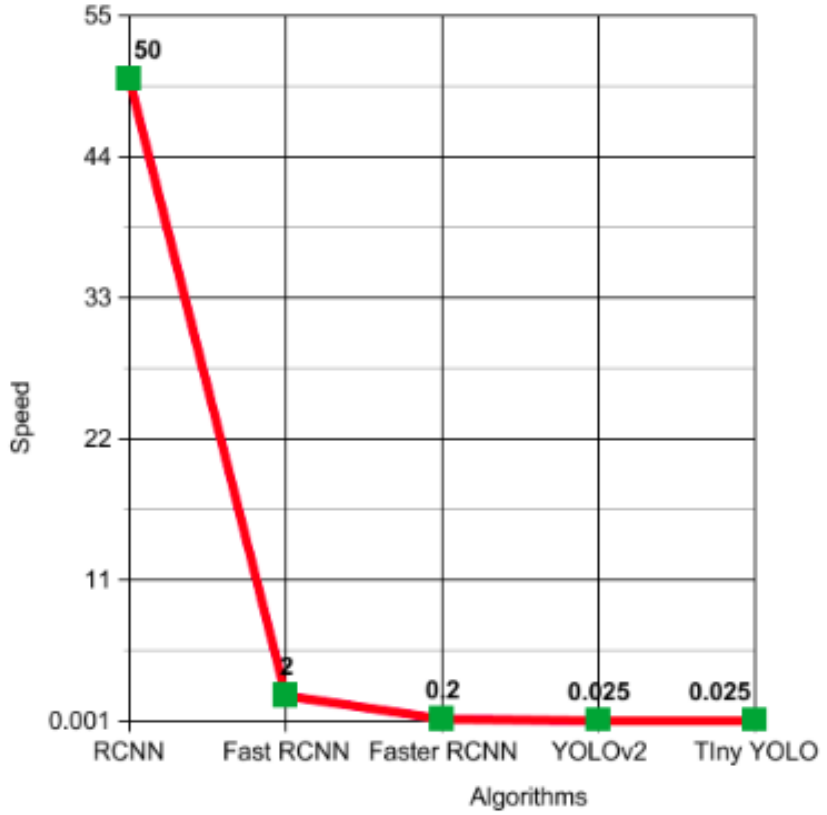


Table 3.b : Speed vs Algorithms

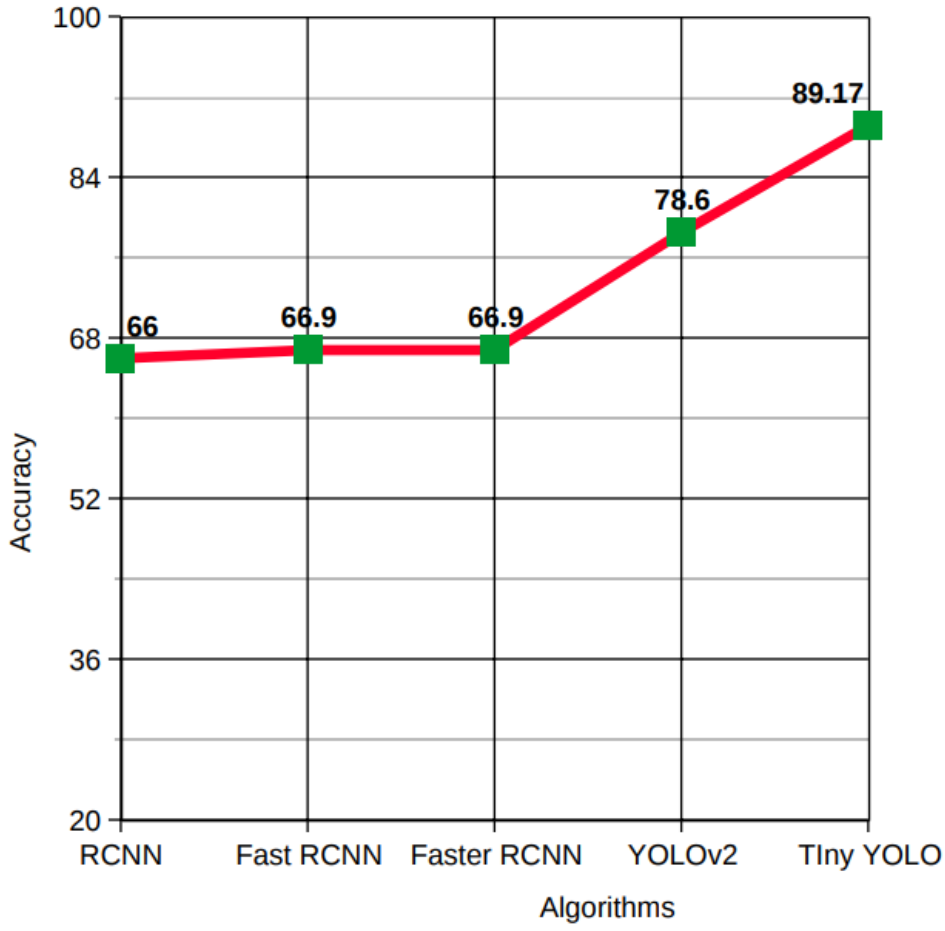


Table 3.c : Accuracy vs Algorithms

Chapter 6

TESTING

6.1 Unit testing

Following are the results of unit testing :

Test Case No.	Test Case	Description	Input	Expected Output	Actual Output	Pass/ Fail
1	Sign Up	Validation of user email id and password	Email, password	Entered details are valid	Entered details are valid	Pass
2	Preferences	Validating the updates on user preferences on fire base	Allergies, courses, cuisines, diet and flavor preferences	Dishes which follow the preferences	Dishes output by the API call follow the preferences	Pass
3	Fruits and vegetables detection	Checking whether correct items are detected or not	Real time scanning of vegetables and fruits	Detection of available fruits and vegetables	Available fruits and vegetables are detected	Pass
4	API call	Checking whether the API returns dishes with respect to input parameters	List of detected fruits and vegetables as well as user preferences	Filtered list of dishes	Filtered list of dishes	Pass

Table 4: Unit testing results

6.2 Integration Testing

The output of the trained model of Tiny YOLO is used for detection of fruits and vegetables. The detected list of vegetables and fruits is easily sent via the API to the server to

fetch the recipes. Also, the recipes follow the user preferences if he/she has any. Thus, the system is successfully able to send the list of detected items and get a recipe list.

6.3 User Acceptance testing

The android application is tested on multiple users of different ages to understand the user acceptance of the application. Within a few minutes, the users were able to use to application easily. With a simple UI design, the users were easily able to understand the purpose and the usage of the application. The users were able to explore the features of the app by themselves.

Chapter 7

RESULT ANALYSIS

Screenshots :

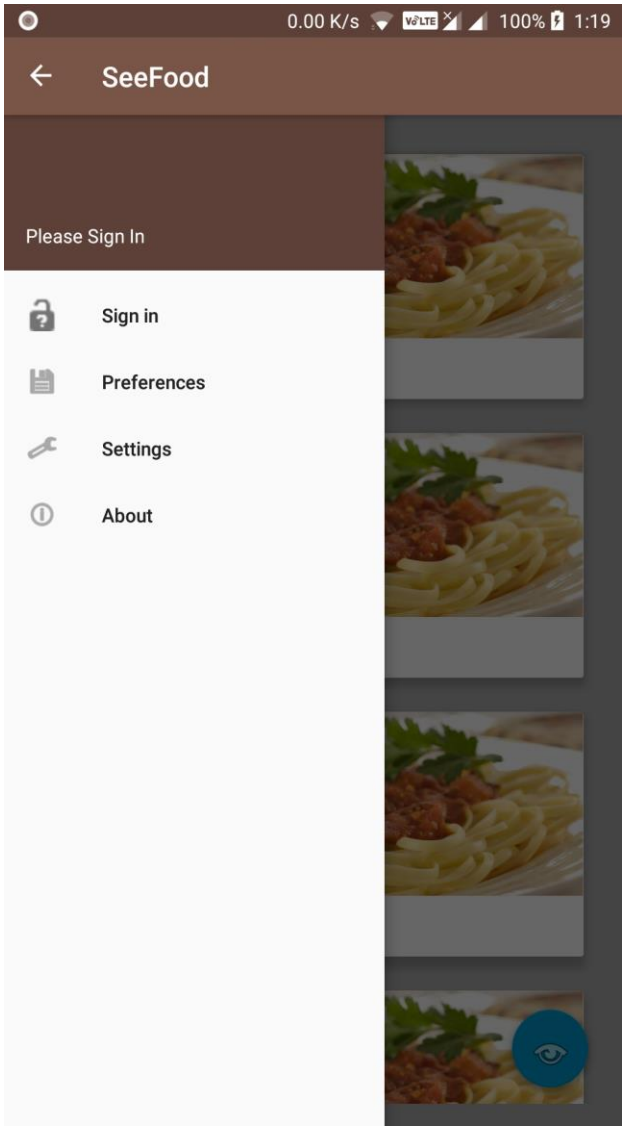
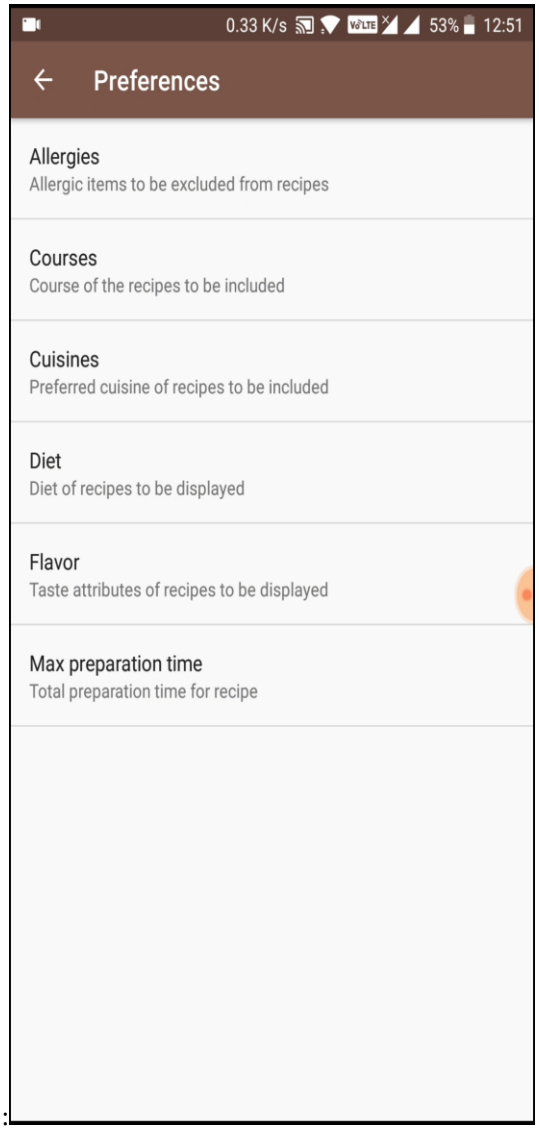


Fig.11 : Navigation Bar



:

Fig. 12 : Option for filtering recipes that are being displayed

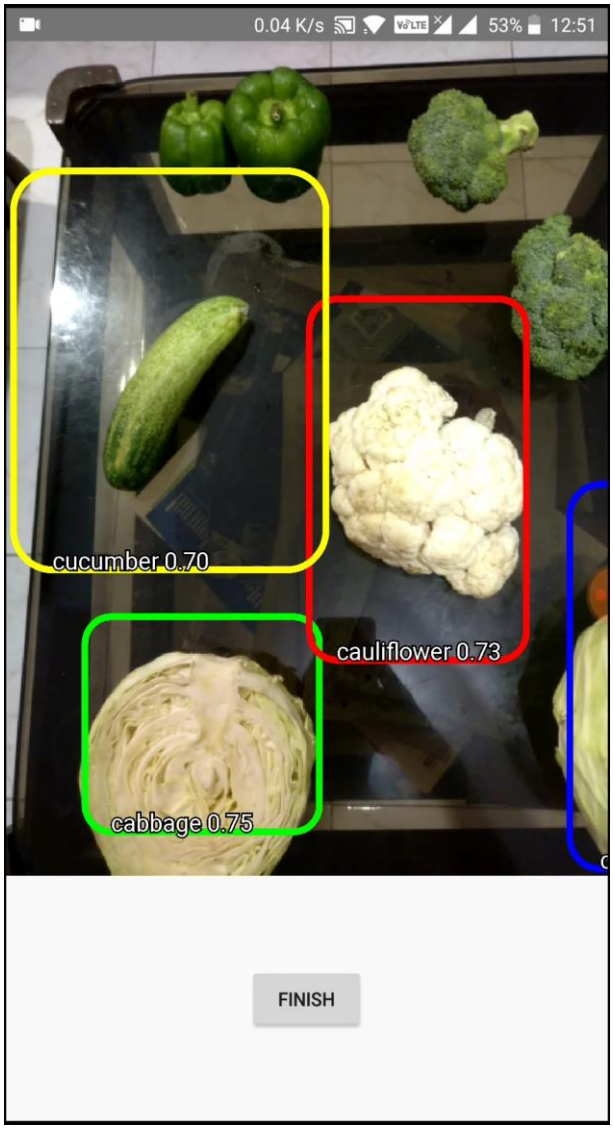


Fig. 13 : Detection Screen

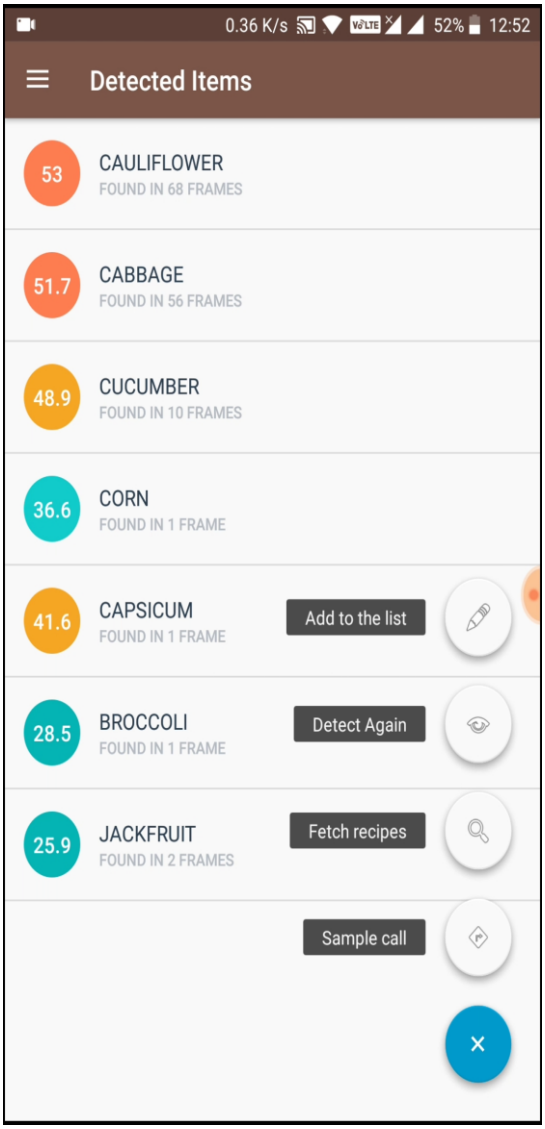


Fig. 14 : Detected Fruits and vegetables list

Fig. 16 : Fetched Dishes (Horizontal Scroll) list

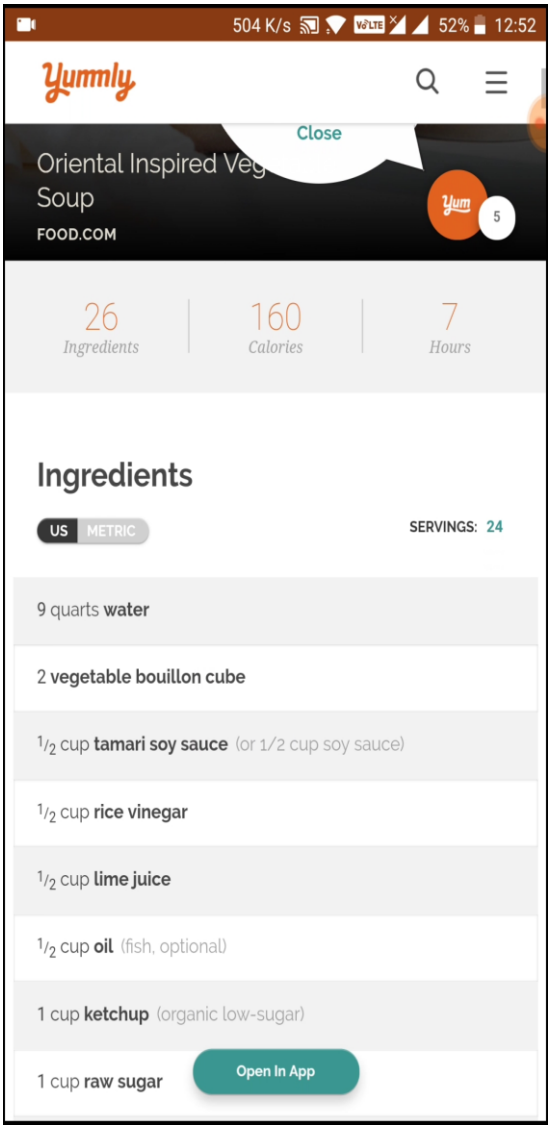


Fig. 17 : Recipe

Chapter 8

CONCLUSION

8.1 Limitations

- Less number of defined classes
- Less accuracy due to less training time
- Small Dataset due to less annotated images found
- Limited API calls (30000)

8.2 Conclusion

Recent advances in object detection are driven by the success of region proposal methods and region-based convolutional neural networks (RCNNs). Faster R-CNN, achieves near real-time rates using very deep networks, when ignoring the time spent on region proposals. Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems. Selective Search, one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks, Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation. Edge Boxes currently provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

This computational bottleneck was eliminated when YOLO was first proposed which introduced bounding box regression. This essentially reduced the test time inference because the convolutional layer did not have to traverse the whole image multiple times at the object detection module but only use cropped images at the classification layer. This bounding box predictor does not generate large number of proposals but generates 5 bounding boxes per grid cell and these 5 boxes are scaled with given number of aspect ratios and scales.

After training for 150 epochs, we achieved a Mean Average Precision (mAP) of 89.17%. This can be improved by training for more number of epochs and on more number of images.

Thus, the system implements Tiny-YOLO to detect as well as classify the vegetables and fruits from the user input images. The recipe suggestions are filtered depending upon the user preferences. Thus, this application negates the user's dependency on food franchises as he/she can prepare recipes with his/her own raw materials and if possible learn new recipes. Thus the proposed system uses state-of-art techniques to use machines in more intuitive manner.

Our application uses Tiny YOLO to perform multiple object detection and classification over the legacy methods of IP i.e. HOG[16] and of CNNs i.e the R-CNN and SPPnet[17]. It

provides quicker(same as Fast-RCNN better than R-CNN) and accurate results due to its revolutionary region of proposals technique.

8.3 Future Scope

- Increase the number of supported classes
- Train the model for more epochs
- Increase the images in dataset as well as create annotations if not present
- Multi platform support
- Create own database of recipes

REFERENCES

References :

[1] Navneet Dalal and Bill Triggs, ”Histograms of Oriented Gradients for Human Detection”, IEEE Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference, 2005

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” in Neural Information Processing Systems (NIPS), 2012.

[3]Matthew D. Zeiler, Rob Fergus, ”Visualizing and Understanding Convolutional Networks”, https://arxiv.org/pdf/1311.2901.pdf, 2013

[4] Karen Simonyan & Andrew Zisserman, ”Very Deep Convolutional Neural networks for large scale image recognition” at ICLR 2015

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. IJCV, 59:167–181, 2004.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,”Deep Residual Learning for Image Recognition”, cs.CV, 2015

[7] Fast R-CNN, Ross Girshick Microsoft Research, 2015, arXiv:1504.08083.

[8] Shaoquing Ren, Kaiming He, Ross Ghrishik, Jian Sun, ”Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks” in IEEE Explore, 2017

[9] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster,stronger. arXiv preprint, 1612, 2016.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. Proc. of the IEEE, 1998.

[11] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” International Journal of Computer Vision (IJCV), 2013.

[12] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation”. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[13] Takuma Maruyama, Yoshiyuki Kawano, Keiji Yanai, “Real-time mobile recipe recommendation system using food ingredient recognition” , ACM digital library, 2012.

[14] Krasin I., Duerig T., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K. OpenImages: A public dataset for large-scale multi-label and multi-class image classification, 2017. Available from https://github.com/openimages

[15] Recipes: yummly.com

[16] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, 2015.

[17]https://en.wikipedia.org/wiki/YUV#Y%E2%80%B2UV420sp_(NV21)_to_RGB_conversion_(Android)

APPENDIX

a. List of Figures

b. List of Tables

PAPERS

PAPER 1 :

Deep Learning based approach to suggest recipes

Himanshu Rawlani, Jayesh Saita, Vignesh Zambre, Priya R. L.
Vivekanand Education Society's Institute of Technology, Mumbai

Abstract: This paper proposes an application that suggests recipes based on an image, clicked by the user, which contains multiple vegetables or fruits. This image can be captured in various environments, lighting conditions and from different angles. To detect multiple vegetables a state-of-art Convolutional Neural Network (CNN) called Faster-RCNN is used.

Faster-RCNN is deployed on an android application which in turn interacts with server to fetch the possible recipes. The server retrieves the recipes from database, filters it and then returns the list to the client application.

Keywords: Object Detection, Region Proposals, Convolutional Neural Network

I. INTRODUCTION

There have been many ideas on the internet about detecting or recognizing what the given food is and what are its components. There are very few machine learning classification applications which differentiate between vegetables and fruits. Hence, the paper proposes an application, which uses machine learning, to perform multiple object detection to find out which fruits/vegetables are there in the captured photo and what recipe can be made using those fruits/vegetables with certain basic assumptions that the user has common ingredients such as salt, milk, spices, etc. in order to make the recipe.

The closest example of application with respect to proposed system is 'Calorie Mama'. This application took the finished food as an input and the output was the number of calories. But, it did not perform well as the users pointed out that the food detection was not consistent and detected dishes sometimes gave different calorie values. Thus it was not accurate.

II. LITERATURE SURVEY

Since 1990, there is a heavy use of Convolutional Neural Network(CNN)[1,8] in several real-time applications. The different variations of CNNs[2,5,6,7] were used for image classification with a high rate of accuracy. It has been observed[2,5] that Region based Convolutional Neural Networks (RCNN) are slow because it performs a CNN forward propagation for each object proposal without sharing any of the computations.

In RCNN, training is done in a multi stage pipeline fashion which results towards expensive space and time requirement. Fast RCNN [5] solves this by taking input entire image and set of object proposals (generated using selective search [3,4]). It shares the computation of conv layers (VGGNet [6]) between different object proposals.

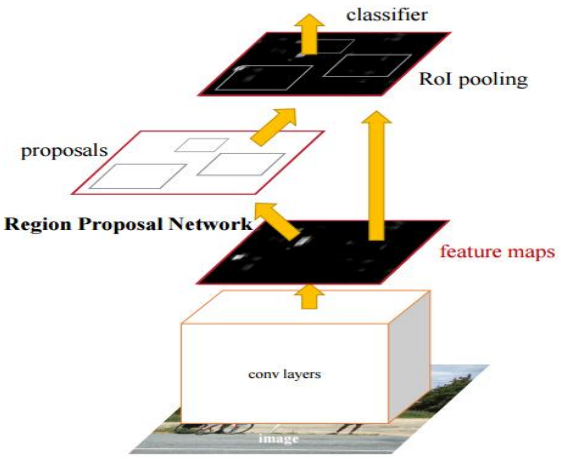


Figure 1: Faster-RCNN Architecture.[7]

In Faster-RCNN[7], the above approach of Fast-RCNN is even improved further. Here, the idea of region proposals networks (RPN) as shown in Figure 1, is introduced.. The authors construct an RPN by adding a few additional convolutional layers on top of Fast-RCNN convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid and thus is a kind of fully convolutional network (FCN)[8] . It can be trained end-to-end specifically for the task for generating detection proposals. This approach takes 0.2 seconds per image and achieves an efficiency of 66.9% and is 250x faster than Fast-RCNN.

The fact that CNNs perform so well on image data is because this neural network architecture makes an assumption that the input data is an image i.e. convolution neural networks only capture local "spatial" patterns in the data and hence are great at finding patterns.

III. PROPOSED SYSTEM

The objective of proposed application is to suggest food recipes on the basis of raw materials (vegetables and fruits) recognized in the photographs taken. The system is based on multiple object detection (multiple vegetables/fruits) and it fetches recipes based on the ingredients detected in the image. The proposed system architecture is depicted in Figure 2.

FIDS30, ImageNet datasets[9] will be used for vegetables/fruit recognition and kaggle dataset[10] as recipe database on server side.

The hyperparameters for CNN to be used are same as that mentioned in [7]. FIDS30 and ImageNet datasets will be used for training CNN on variety of fruits and vegetables. Training is carried

out on a GPU prior to application deployment. The learned weights of the CNN will be deployed along with the application and these weights will be used in inference part i.e. object detection part.

A. SYSTEM ARCHITECTURE

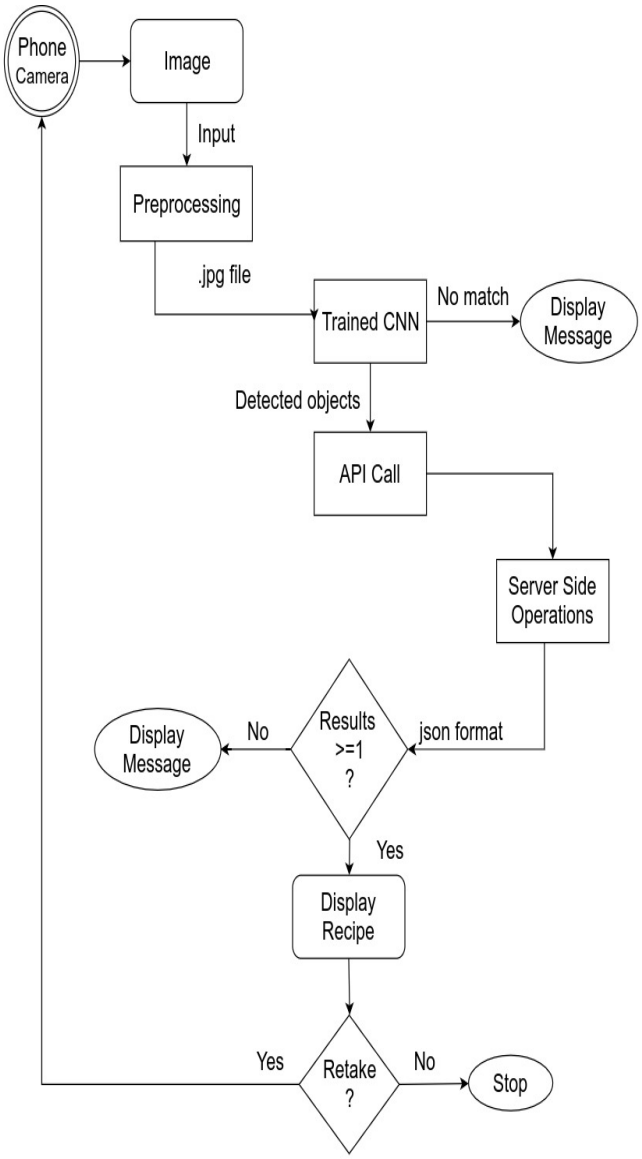


Figure 2: System Architecture

For the first time when user opens the app, he/she will be prompted to register in the system using Google/Facebook sign in APIs or user can enter any other email and password of their choice. After successful registration, user will be asked basic questions about their The taste preferences (eg. sweet or spicy), their age and weight (for better diet recommendation), location (for regional preferences).

The home screen of the app will list various recipes that match user preferences and those that are popular (i.e. is being viewed) by other users in that region.

Pressing the camera button on home screen will launch the in-app camera functionality. This in-app camera captures the image and performs the required preprocessing required for input to the trained CNN.

Detecting objects in the image and listing out the detected vegetables (if any) will be done on user's device. This is the inference part and is carried out by the Trained CNN deployed within the application.

This design decision is made so as to reduce the network load of uploading complete image to the servers and then performing object detection. It will also reduce overall execution time of vegetable/fruit detection.

The list of detected vegetables is sent to the server in POST request. The CNN is trained to ensure less false positives i.e. the system knows that no vegetables or fruits are found and it displays appropriate message to user advising to improve lighting condition or hold camera at a particular angle.

After server side processing (explained next) the system lists out the recipes in easy to read and responsive layout. Users can see the list of detected vegetables and have the option to remove one or more from the list. After that, the application reloads the recipes with the updated list of vegetables.

B. STRUCTURE OF SERVER SIDE

The list of detected vegetables will be sent to the server. The server will return the possible recipes which are filtered according to user preferences. The server does the following operations

A. Fetching possible recipes
B. Executing query to extract user preferences
C. Performing filtration of recipes
D. Displaying recipes

i. THE API

An android application cannot directly access a database residing at a server. It needs some intermediary medium which the app can request for the recipes, and through which the recipes are received by the application. An Application Package Interface (API) does this job. API is a collection of PHP (or any server side scripting language) files. Each file does a single operation like read, write, delete etc. The basic operations are GET, POST, PUT etc.

The POST operation requests the server for recipes by passing the list of detected vegetables. The GET operation is similar to POST, but in that the parameters (list of vegetables) are appended in the url.

Whereas in POST, the parameters are added in the body of the packet. So it is easy and secure way of passing many parameters. The PUT operation is used to write into the database. In this case, PUT operation is not needed as the application will not cause the database to change (i.e new recipe added). The database is predefined according to the recipe dataset and it will not change unless new recipes are added (by admin).

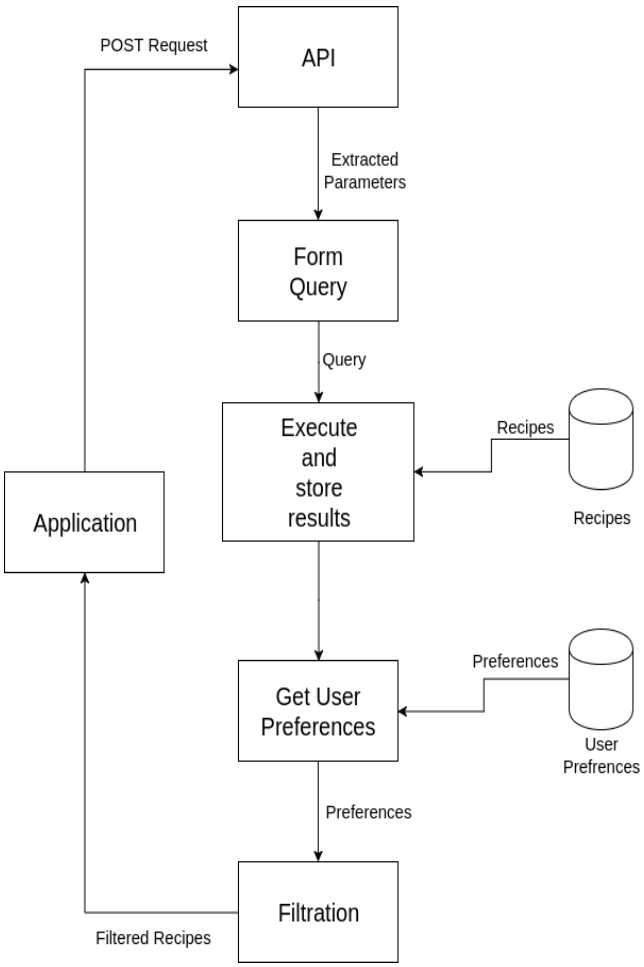


Fig 3: Server Side Operations

ii. FETCHING POSSIBLE RECIPES

The application will pass the list of parameters in a POST request. The API file handling POST request will receive this request. The API will first extract all the parameters. Then it will form a SQL query with these parameters and it will be executed.

iii. EXTRACTING PREFERENCES

When the user registers with the application, information like age, location, weight and other preferences from the user is retrieved. Before the filtration process, these user preferences are retrieved from the database.

iv. FILTERING RECIPES

Next, the filtration of recipes is done by taking into consideration the above extracted preferences. Various factors considered here are as follows:

a. Age - If an elder person is using this app, then a recipe which has high oil content and/or spicy will not be shown to him/her. Similarly, if a young person is using the app, then more tasty dishes will be shown.
b. Location - We will put the recipes at the top of the list that are most suitable for that particular region.
c. Weight - If the person using the app is fat, then the dishes which are fatty will not be shown to the user.

v. DISPLAYING RECIPES

After filtration of the recipes, the server now sends back the recipe list to the application. In this case, no extra API is required to send back the result. The server just prints the result in Javascript Object Notation (JSON) format. The application reads this JSON and parses it and displays the result accordingly.

IV. CONCLUSION AND FUTURE WORK

Thus, the system implements Faster R-CNN to detect as well as classify the vegetables and fruits from the user input image. The detected vector of vegetables is sent to the server for retrieving the recipe suggestions. These recipe suggestions are filtered depending upon the user profile. This filtered list is then sent to the client device. Thus, this application negates the user's dependency on food franchises as he/she can prepare recipes with his/her own raw materials and if possible learn new recipes. Thus the proposed system uses state-of-art techniques to use machines in more intuitive manner.

We need to ensure that there are enough recipes for users to try out. One way to do that is to provide functionality to users to upload their own recipes and allow other users to rate and add comments to it. Regularising and content monitoring system shall also be added to ensure relevant and unique content is uploaded.

V. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. You Only Look Once: Unified, Real-Time Object Detection.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in Neural Information Processing Systems (NIPS), 2012.

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. IJCV, 59:167–181, 2004.

[4] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," International Journal of Computer Vision (IJCV), 2013.

[5] Fast R-CNN, Ross Girshick Microsoft Research, 2015, arXiv:1504.08083.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015

[7] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, 2015.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation". IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[9] Datasets: Epicurious - Vegetable image dataset http://image-net.org/explore?wnid=n07707451, Fruit image dataset http://www.vicos.si/Downloads/FIDS30

[10] Recipe: Recipes with Rating and Nutrition:https://www.kaggle.com/hugodarwood/epi recipes, Nutrition facts from foods around the world: https://www.kaggle.com/openfoodfacts/world-food-facts

Plagiarism Report:

Plagiarism Scan Report

Summary	
Report Genrated Date	27 Oct, 2017
Plagiarism Status	**100% Unique**
Total Words	996
Total Characters	6055
Any Ignore Url Used	

Content Checked For Plagiarism:

Deep Learning based approach to suggest recipes
Himanshu Rawlani, Jayesh Saita, Vignesh Zambre, Priya R. L.
Vivekanand Education Society's Institute o Technology, Mumbai

Plagiarism Scan Report

Summary	
Report Genrated Date	27 Oct, 2017
Plagiarism Status	**100% Unique**
Total Words	992
Total Characters	6048
Any Ignore Url Used	

Content Checked For Plagiarism:

Figure 4: Max-Pooling done on window

We do max pooling with all o the stack o our iltered images and in every case get a smaller set o ltered images.

Plagiarism Scan Report

Summary	
Report Genrated Date	27 Oct, 2017
Plagiarism Status	**100% Unique**
Total Words	583
Total Characters	3358
Any Ignore Url Used	

Content Checked For Plagiarism:

"+/-" means partial invariance.
⅓ means one o the colors is invariant to given property.

Class: D17 A/B/C

Group No.: 51

Project Evaluation Sheet 2017 - 18

Title of Project: Seefood – Food that you can see

Group Members: Himanshu Rawlani (D17A 64), Jayesh Saita (D17A, 66), Vignesh Zambre (D17A, 78)

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg &Mgmt principles (3)	Life - long learning (3)	Professional Skills (5)	Innovative Approach (5)	Total Marks (50)
Review of Project Stage 1	3	3	2	2	3	1	2	2	2	2	2	2	2	2	30

Comments:

Name & Signature Reviewer1

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg &Mgmt principles (3)	Life - long learning (3)	Professional Skills (5)	Innovative Approach (5)	Total Marks (50)
Review of Project Stage 1	3	3	2	2	3	1	2	2	2	2	2	2	2	2	30

Comments:

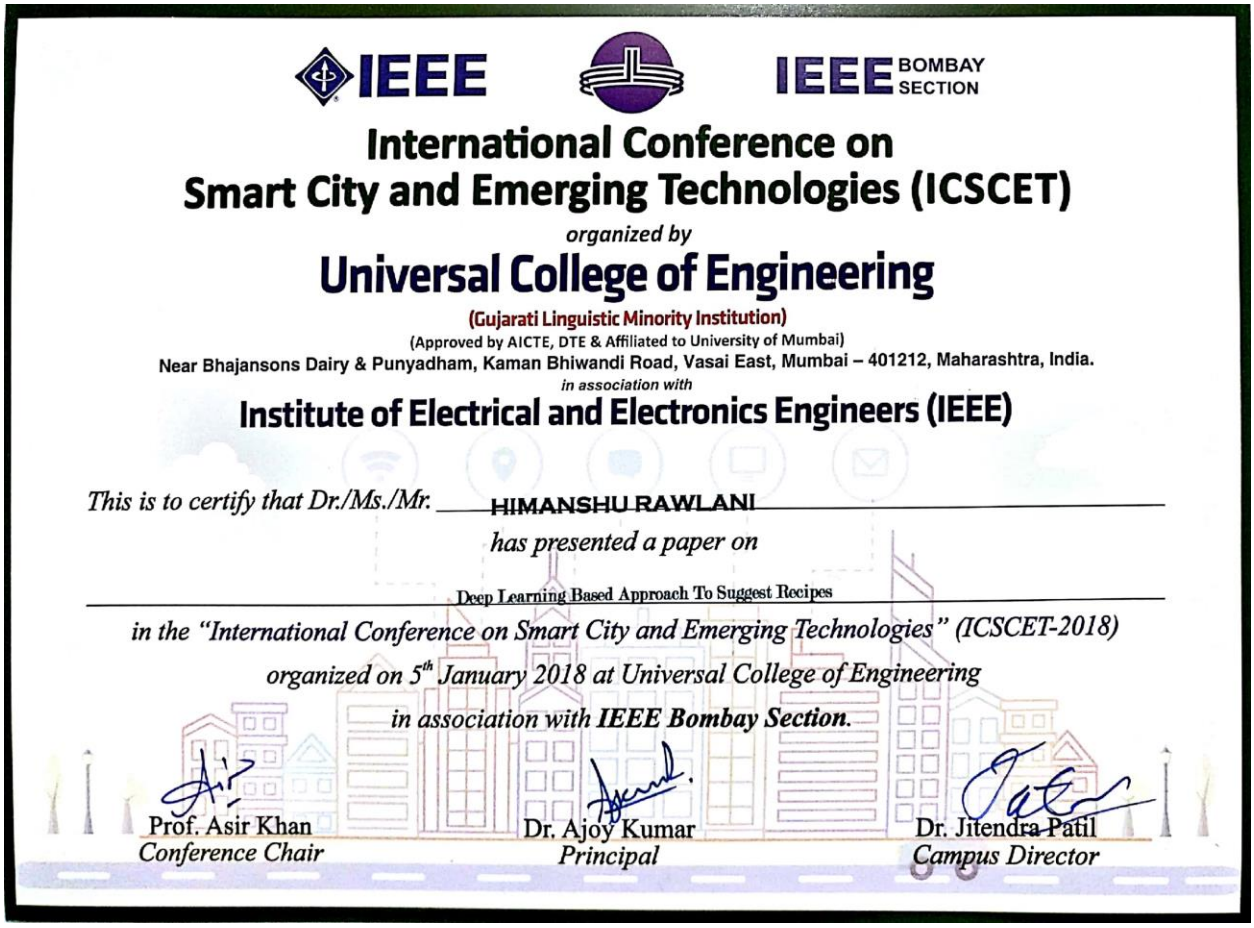
→ Not clear about actual methodology.

* Sound knowledge of MM.

Date: 26th September 2017

Name & Signature Reviewer2

IEEE conference certificates:



IEEE IEEE BOMBAY SECTION

International Conference on
Smart City and Emerging Technologies (ICSCET)

organized by

Universal College of Engineering

(Gujarati Linguistic Minority Institution)
(Approved by AICTE, DTE & Affiliated to University of Mumbai)
Near Bhajansons Dairy & Punyadham, Kaman Bhiwandi Road, Vasai East, Mumbai – 401212, Maharashtra, India.

in association with

Institute of Electrical and Electronics Engineers (IEEE)

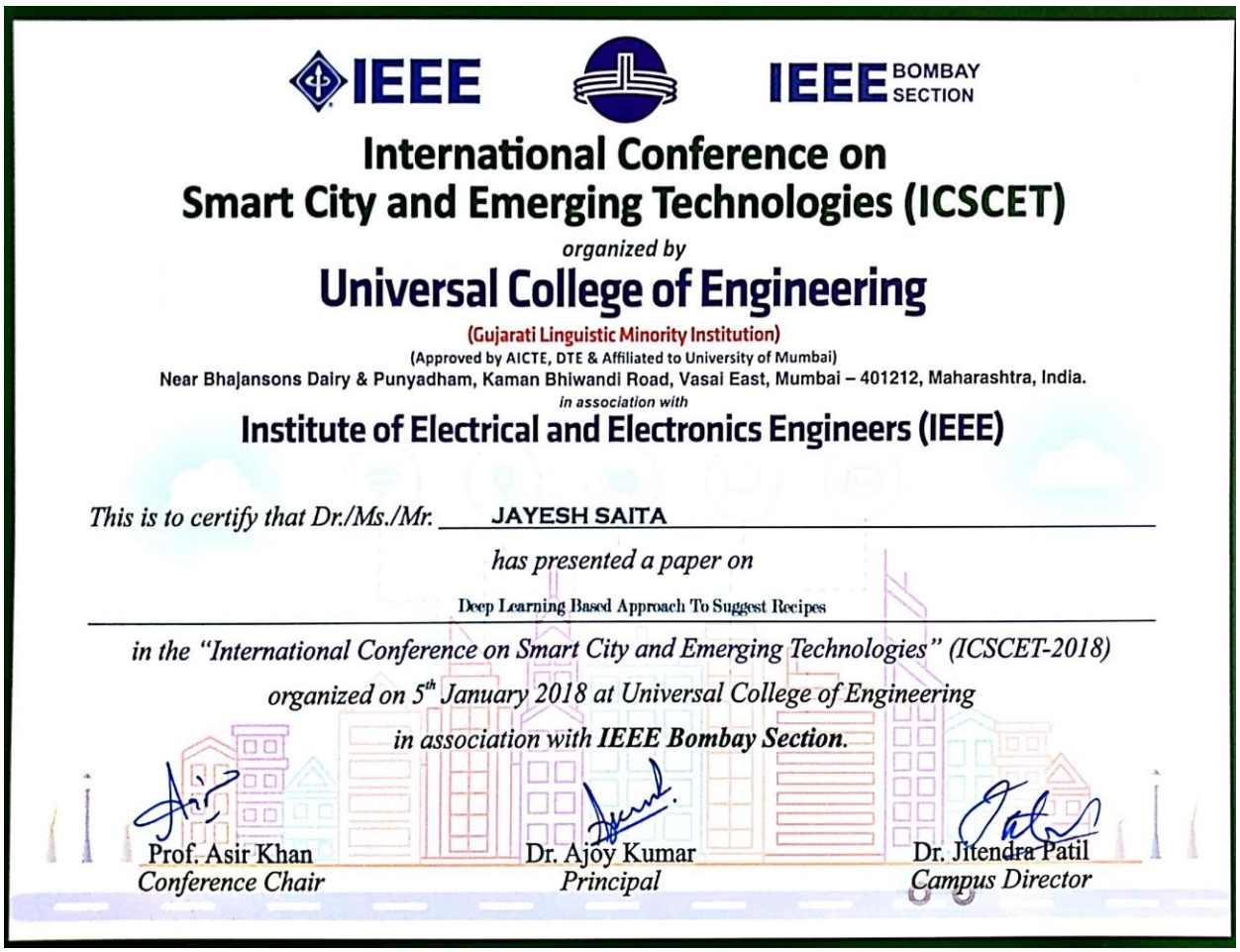
This is to certify that Dr./Ms./Mr. HIMANSHU RAWLANI

has presented a paper on

Deep Learning Based Approach To Suggest Recipes

in the "International Conference on Smart City and Emerging Technologies" (ICSCET-2018)
organized on 5th January 2018 at Universal College of Engineering
*in association with **IEEE Bombay Section.***

Prof. Asir Khan, *Conference Chair* | Dr. Ajoy Kumar, *Principal* | Dr. Jitendra Patil, *Campus Director*



IEEE IEEE BOMBAY SECTION

International Conference on
Smart City and Emerging Technologies (ICSCET)

organized by

Universal College of Engineering

(Gujarati Linguistic Minority Institution)
(Approved by AICTE, DTE & Affiliated to University of Mumbai)
Near Bhajansons Dairy & Punyadham, Kaman Bhiwandi Road, Vasai East, Mumbai – 401212, Maharashtra, India.

in association with

Institute of Electrical and Electronics Engineers (IEEE)

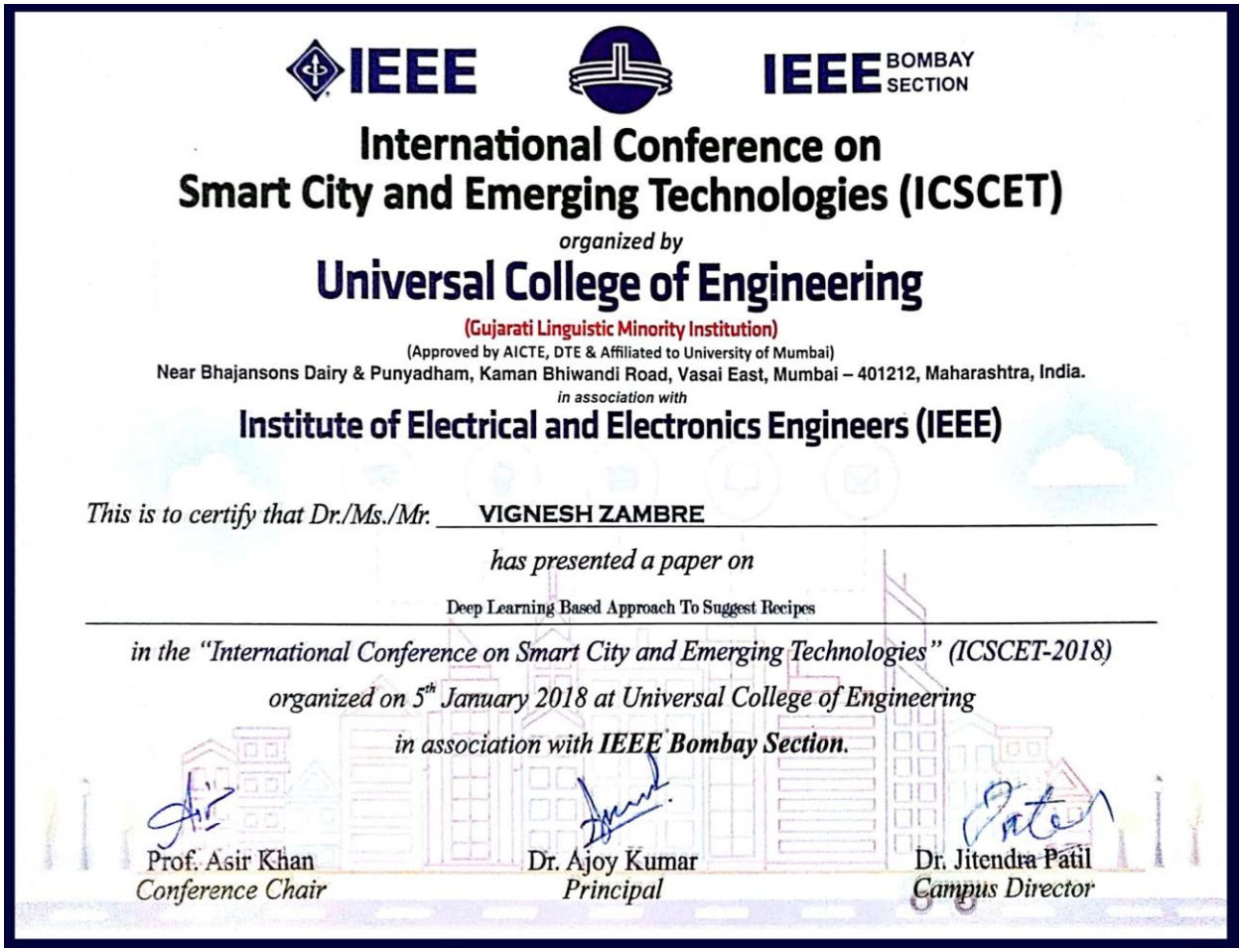
This is to certify that Dr./Ms./Mr. JAYESH SAITA

has presented a paper on

Deep Learning Based Approach To Suggest Recipes

in the "International Conference on Smart City and Emerging Technologies" (ICSCET-2018)
organized on 5th January 2018 at Universal College of Engineering
*in association with **IEEE Bombay Section.***

Prof. Asir Khan, *Conference Chair* | Dr. Ajoy Kumar, *Principal* | Dr. Jitendra Patil, *Campus Director*



IEEE IEEE BOMBAY SECTION

International Conference on Smart City and Emerging Technologies (ICSCET)

organized by

Universal College of Engineering

(Gujarati Linguistic Minority Institution)

(Approved by AICTE, DTE & Affiliated to University of Mumbai)

Near Bhajansons Dairy & Punyadham, Kaman Bhiwandi Road, Vasai East, Mumbai – 401212, Maharashtra, India.

in association with

Institute of Electrical and Electronics Engineers (IEEE)

This is to certify that Dr./Ms./Mr. VIGNESH ZAMBRE

has presented a paper on

Deep Learning Based Approach To Suggest Recipes

in the "International Conference on Smart City and Emerging Technologies" (ICSCET-2018)

organized on 5th January 2018 at Universal College of Engineering

in association with ***IEEE Bombay Section.***

Prof. Asir Khan
Conference Chair

Dr. Ajoy Kumar
Principal

Dr. Jitendra Patil
Campus Director

PAPER 2 :

SeeFood - Recipe Suggestion using Deep Learning

Himanshu Rawlani, Jayesh Saita, Vignesh Zambre, Priya R. L.
Vivekanand Education Society's Institute of Technology, Mumbai

Abstract: This paper describes an android application that suggests recipes by scanning raw vegetables and fruits using smartphone camera. The user points the camera at the vegetables and the app sends the list of detected vegetables and user preferences to recipe API. Recipe API also supports user preferences like Diet, Allergies, Cuisines etc. For object detection Tiny YOLO convolutional neural network is used. It is the smaller version of state-of-the-art model You Only Look Once (YOLO)[1]. It has 15 layers so that it can be deployed on smartphones with real time inferencing capability. The dataset is formed using images taken from Imagenet and Google Open Images dataset. After retraining the model on our dataset we got a Mean Average Precision(mAP) of 89.17%.

The project is available at GitHub - https://github.com/himanshurawlani/SeeFood

Keywords: Object Detection, Convolutional Neural Network, Tiny YOLO, Recipes

I. INTRODUCTION

There have been many ideas on the internet about detecting or recognizing what the given dish is using smartphone camera and predicting what are its nutrients. There are very few machine learning classification applications which differentiate between vegetables and fruits. This paper proposes an application, which uses deep learning, to perform multiple object detection to find out which fruits/vegetables are there in a given frame from camera preview and what recipe can be made using those fruits/vegetables with certain basic assumptions that the user has common ingredients such as salt, milk, spices, etc. in order to make the recipe. Recipes are provided by taking into consideration user's diet preferences.

The closest example of application with respect to proposed system is 'Calorie Mama'. This application does the reverse process of our proposed system, it takes the prepared dish image as an input and the output is the number of calories in the dish.

II. LITERATURE SURVEY

Since 1990, there has been heavy use of Convolutional Neural Network (CNN) [1, 7] in several real-time applications. The different variations of CNNs [2, 5, 6 and 7] were used for image classification with a high rate of accuracy. It has been observed [2, 4] that Region based Convolutional Neural Networks (RCNN) are slow because it performs a CNN forward propagation for each object proposal without sharing any of the computations. Also these networks repurpose classifiers to perform object detection.

In RCNN, training is done in a multi stage pipeline fashion which results towards expensive space and time requirement. Fast RCNN [5] solves this by taking input entire image and set of object proposals (generated using selective search [3, 4]). It shares the computation of convolutional layers (VGGNet [6]) between different object proposals.

In Faster-RCNN [7], the above approach of Fast-RCNN is even improved further.. The authors construct an RPN by adding a few additional convolutional layers on top of Fast-RCNN convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid and thus is a kind of fully convolutional network (FCN) [8]. It can be trained end-to-end specifically for the task for generating detection proposals. This approach takes 0.2 seconds per image and achieves an efficiency of 66.9% and is 250x faster than Fast-RCNN.

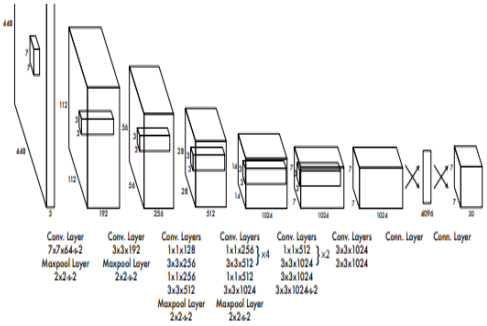


Fig. 1. YOLO Architecture [1]

In YOLO, instead of using RPNs, it divides the whole image in grids and runs object prediction on those grids. Now the predicted objects are fed as an input to the classifier in terms of bounding boxes. This algorithm is much faster than Faster RCNN as the input image is processed only once.

III. PROPOSED SYSTEM

Real time food ingredient recognition [1] has been done before in 2012. They used bag-of-features and SURF [2] with color histogram extracted from multiple images as image features. Along with this, linear SVM with one-vs-rest strategy was used as a classifier. They achieved a recognition rate of 83.93%.

The proposed system is an android application that uses Tiny-YOLO, the smaller and fast version of YOLO. It was made specifically for embedded devices with low computational power. It provides real time inferencing capability on smartphones. Apart from fetching recipes, recipe

API also filters the recipes according to preferences like Allergies, Diets, Cuisines etc.

Apart from these core features, the user can also add recipes to favorites, view recently viewed recipes and share the recipes with others. Basic system architecture is depicted in figure **2**.

A. System Architecture

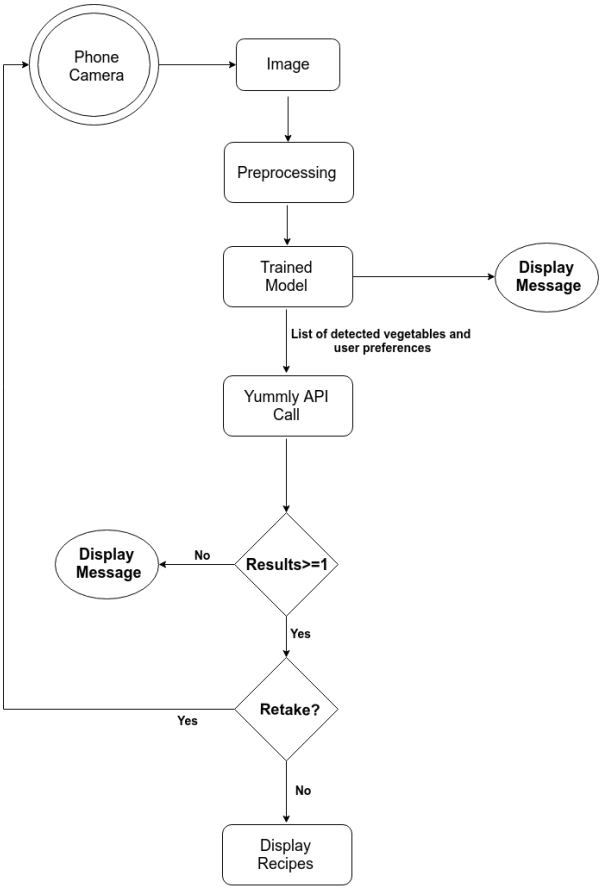


Fig.2 : SeeFood Architecture

When user opens the app, he/she sees home screen with multiple cards displaying recent recipes. The navigation drawer shows user a list of options that user can use to set preferences or settings as per their needs.

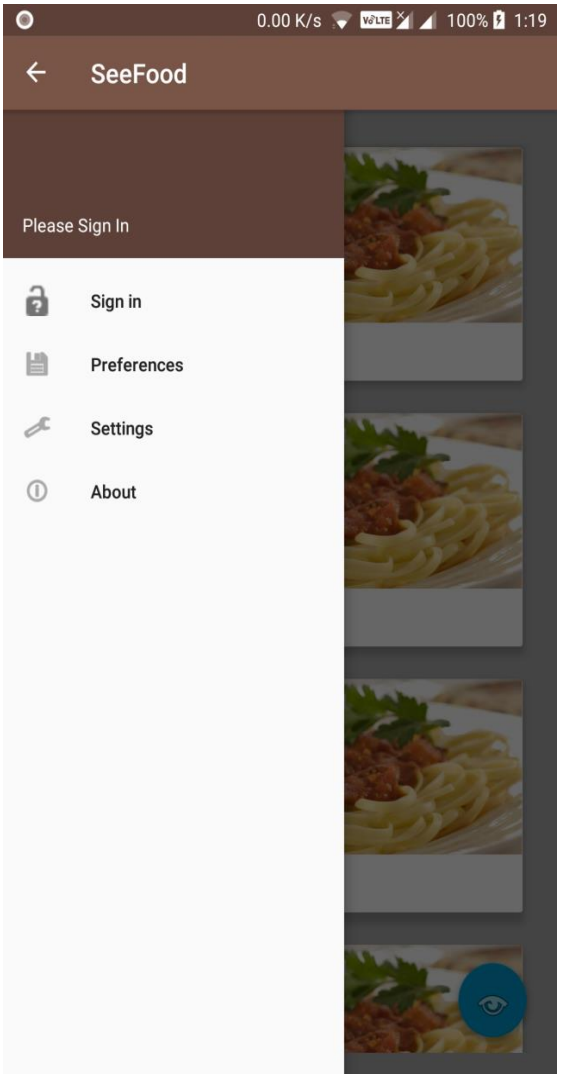


Fig.3 : Navigation bar

On clicking 'Sign In', user is be prompted to register in the system using Google/Facebook/Twitter sign in or user can enter any other email and password of their choice. After successful registration, user can access the 'Preferences' section. By default, no options are selected. The user can select the desired options. These preferences are considered when displaying the recipes to user.

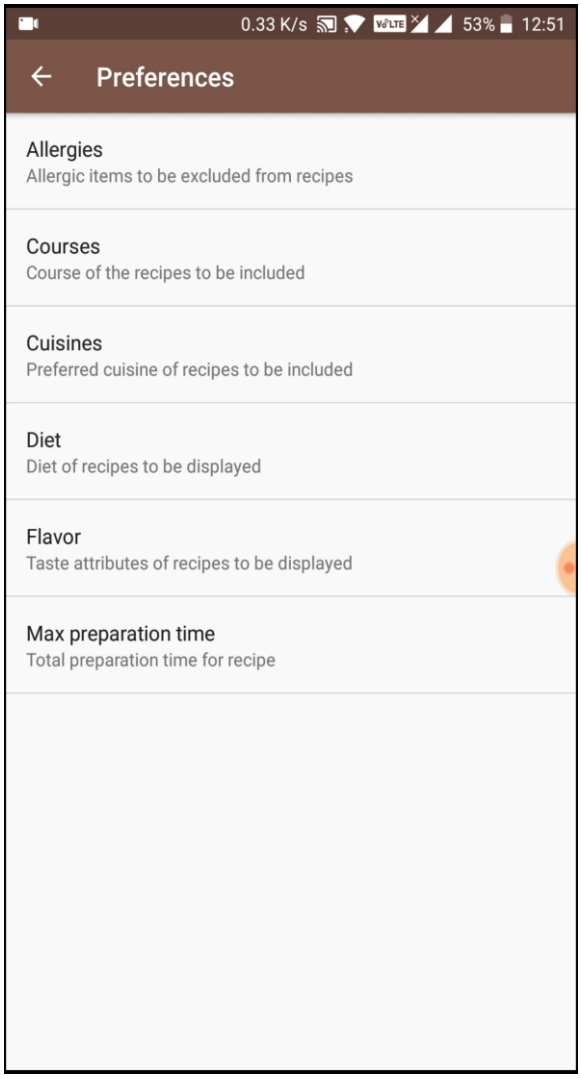


Fig.4 : User recipe preferences

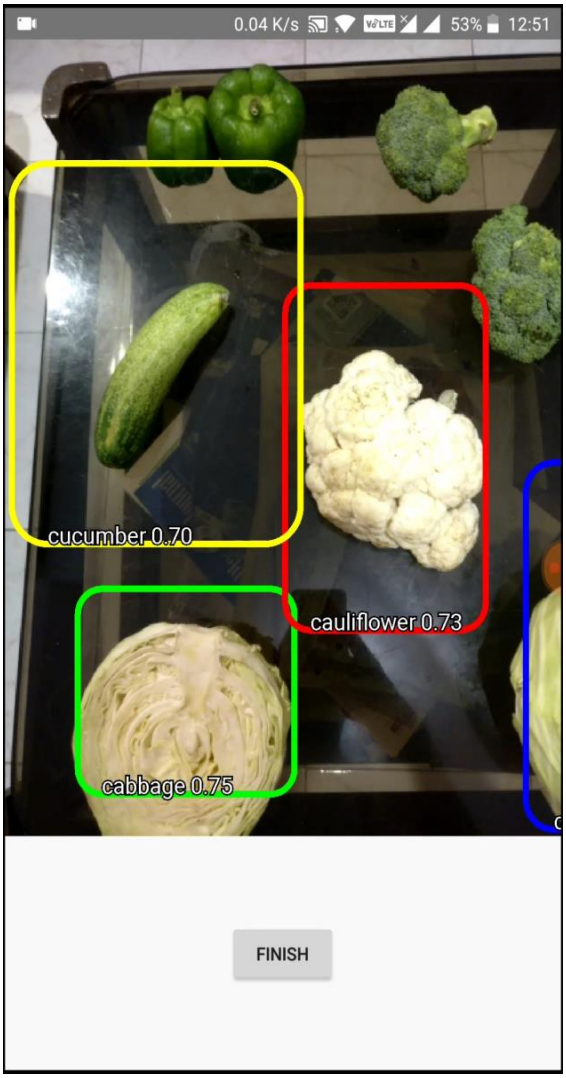


Fig.5 : Detection screen

On pressing the floating action button having eye shaped icon, user can start detecting the vegetables. The in-app camera starts, which pipelines the images coming from the camera preview, and does the preprocessing required and passes it to the detector model. The inference part is carried out by the trained model deployed within the application. This design decision is made so as to reduce the network overhead of uploading complete image to the servers and then performing object detection. It also reduces the overall execution time of vegetable/fruit detection.

User just has to point the smartphone camera to the vegetables until a bounding box appears around the vegetables. This means that the vegetable is detected. User has to tap on the 'Finish' button to end detection.

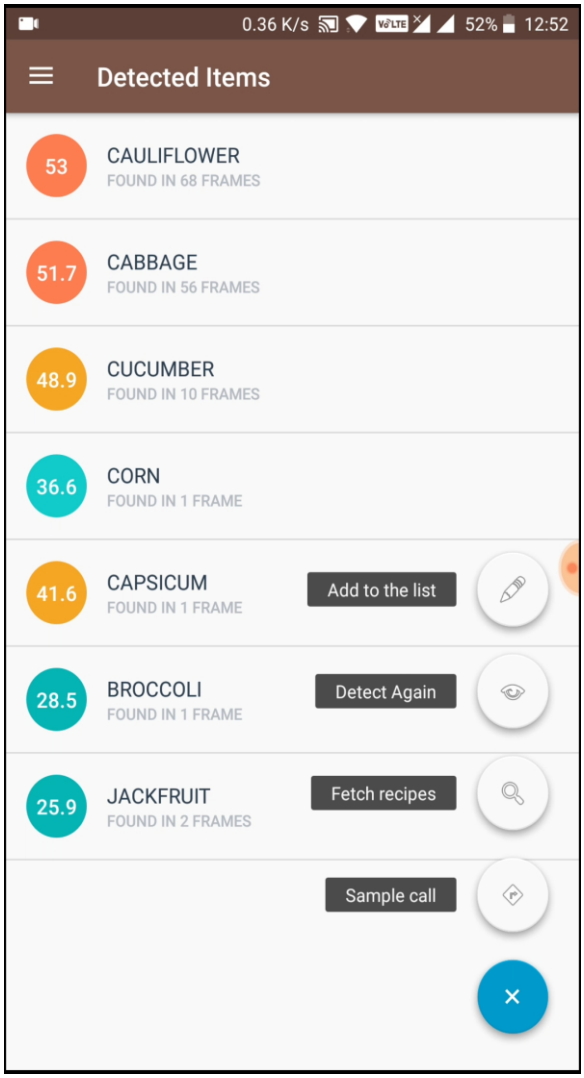


Fig.2 : Detected fruits and vegetables

After that, a list of detected vegetables will appear. User can verify the list as well as add/remove items from the list by pressing the '+' floating action button.

Fig.2 : Recipe overview from API

After pressing the 'Fetch Recipes' button, the above list and all the user preferences are sent to recipe API. The recipe API then returns the list of possible recipes satisfying user constraints. It is possible that the api returns no results. This happens when user specifies contradictory preferences.

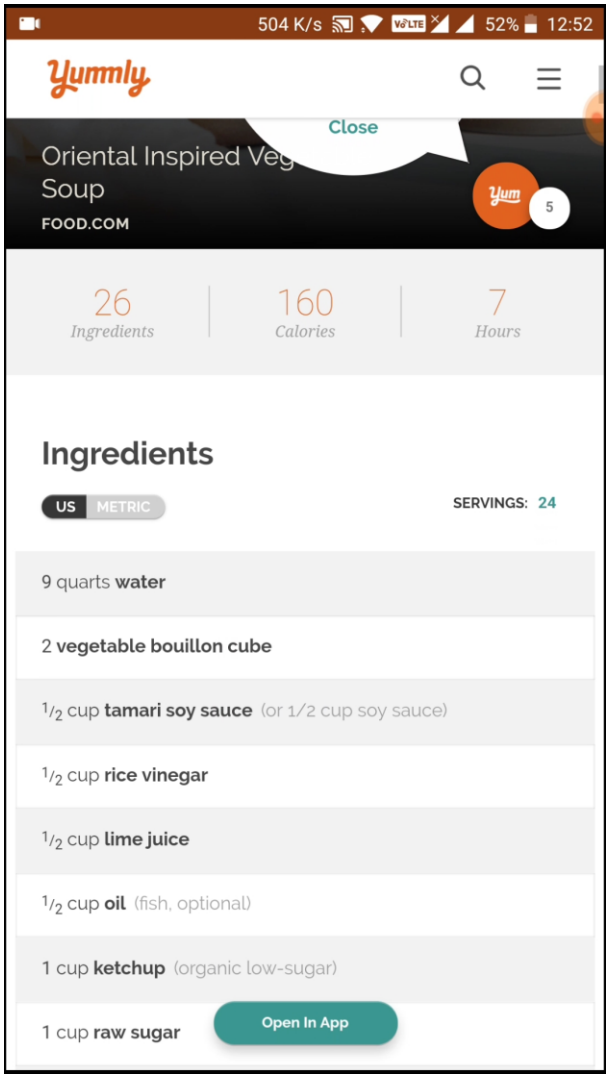


Fig.2 : Complete recipe

The user when seeing the list of recipes can add the desired recipes to favourites (by pressing the star shaped button to right of recipe title) so the user need not scan the vegetables again. The user can also share the recipe by pressing the share button which is located to right of star button.

B. Dataset collection

Available images and their annotations were downloaded from ImageNet and Googles Open Images Dataset. Googles Open Images dataset[12] has a relational database which mapped annotations to corresponding image urls and many other mappings such as test, training and validation sets. Python script was used to scrape the required images.

C. Tiny-YOLO Model Description

Layer	Filter size	Stride	Activation fn	Dimensions
Input				416,416,3
Convolution + bnorm	3×3	1	leaky	416,416,16
MaxPooling	2×2	2		208,208,16
Convolution + bnorm	3×3	1	leaky	208,208,32
MaxPooling	2×2	2		104,104,32
Convolution + bnorm	3×3	1	leaky	104,104,64
MaxPooling	2×2	2		52,52,64
Convolution + bnorm	3×3	1	leaky	52,52,128
MaxPooling	2×2	2		26,26,128
Convolution + bnorm	3×3	1	leaky	26,26,256
MaxPooling	2×2	2		13,13,256
Convolution + bnorm	3×3	1	leaky	13,13,512
MaxPooling	2×2	1		13,13,512
Convolution + bnorm	3×3	1	leaky	13,13,1024
Convolution + bnorm	3×3	1	leaky	13,13,512
Convolution	1×1	1	linear	13,13,90

We have performed transfer learning on Tiny-YOLO model, which is pre-trained on COCO dataset as the base model, on our own dataset of vegetables and fruits which has 13 different classes.

Originally, Tiny-YOLO is written using DarkNet which is written in C++ [3]. Here we have used DarkFlow which is a port of this C++ code to Tensorflow. Thus, instead of DarkNet we have used the DarkFlow library.

We have used Google Collaboratory to train the model. Google Collaboratory is an online Jupyter notebook like environment that provides free GPU (Tesla K80) processing for 12 hrs at a stretch.

For loading the images, annotations and script, a github repository is created to store all the required files. We got the images with their annotations from ImageNet and Google's Open Images dataset. The annotations were already

generated and system does not annotate any image.

The repository is cloned inside Google collab and trained the model for 150 epochs. After training, we generated the .pb file for weights and used it for offline prediction on the android application

D. Tiny-YOLO hyperparameters

YOLO model hyperparameters:

- H = 13
- W = 13
- Box = 5
- Classes = 13
- Scales = [1.0, 5.0, 1.0, 1.0]

Training hyperparameters:

- Learning rate = 1e-05
- Batch size = 32
- Epoch number = 150
- Optimiser = Adam
- Dataset size = 4823 images
- GPU = Tesla K80

D. Image preprocessing

A variety of android devices are available and each sports a different camera hardware, software and supported resolution. Our system uses custom in app camera in order to perform real time image preprocessing and drawing bounding boxes. It also eliminates dependency on system apps and services.

Our system displays camera preview in a fragment layout. We don't use front facing camera in our application, though it's possible but not practicable. Our application supports devices with Android version 6.0 and above. Our system also provides support for both Camera1 and Camera2 APIs provided by android. We check whether camera device is capable of supporting advanced imaging applications and whether the device provides full hardware level support. Following operations are supported in full hardware level support:

- Creating capture sessions
- Per frame control
- Manual sensor control
- Manual post-processing control
- Defining required exposure time range
- Defining required max frame duration
- Support for cropping preview regions

Our system requires a minimum preview size of 320x320 to be supported by the device and the ideal size is 640x480. This preview size is the input size in pixels desired by Tensorflow (width and height of a square bitmap). Storage bitmaps are initialised once the resolution is known.

Android Camera API streams image frames in the form of byte array which is in YUV color encoding system [11]. Our system performs YUV to RGB bitmap conversion using native C++ code in order to improve performance. The converted image is then cropped into 416x416 size which is the required input layer dimensions of Tiny YOLO model. The object detection is performed on this cropped RGB bitmap frame on a background thread.

E. Object tracking

Tiny YOLO model returns a set of bounding box coordinates along with its class labels for the detected objects. The result set is a hashset which stores all the recognitions while the camera is rolled on objects. This result set is passed to the DisplayResults activity. ResultSet stores objects which has many fields like id, name etc.

Before performing object tracking we initialise few parameters. The maximum percentage of bounding box that can be overlapped by another box is set to 20%. The bounding box is replaced by a higher score bounding box if overlap is more than 20% or if the correlation of new results with the old results falls below 75%. The minimum correlation value required to draw the bounding box is 30%.

For every frame on which inference is run, the bounding box coordinates is passed to the tracker which maps those coordinates to rectangle and redraws the frame using Android canvas library. When a frame is being inferenced the intermediate frames sent by the camera are dropped and after inference is complete the latest frame from the camera is used for object detection.

IV. RESULTS AND ANALYSIS

The running time of other object detection system: fast RCNN scanning takes 1-2 seconds depending on how objects are placed (on average about 1.5s), and Faster R-CNN with VGG 16 takes 220ms on 2000 proposals (or 223ms if using SVD on fully-connected layers [2]). This system with Tiny YOLO takes in total 25ms for both bounding box regression and detection. With the convolutional features shared, the bounding box alone only takes 10ms computing the additional layers.

Detection results of Tiny YOLO on MS COCO test set using different settings of anchors: Training data contains 4823 images with the default setting of using 3 scales and 3 aspect ratios results in 89.17% mAP. By default 3 scales and 3 aspect ratios are used. Using just 3 scales with 1 aspect ratio (89.17%) is as good as using 3 scales with 3 aspect ratios on this dataset, suggesting that scales and aspect ratios are not disentangled dimensions for the detection accuracy.

Algorithm	Test time per image(secs)	Speed Up	mAP (VOC)
R-CNN	50	1x	66%
Fast R-CNN	2	25x	66.9%

Faster R-CNN	0.2	250x	66.9%
YOLOv2	0.025	2000x	78.6%
Tiny-YOLO	0.025	2000x	89.1%

.

V. CONCLUSION AND FUTURE SCOPE

After training for 150 epochs, we achieved a Mean Average Precision (mAP) of 89.17%. This can be improved by training for more number of epochs and on more number of images.

Thus, the system implements Tiny-YOLO to detect as well as classify the vegetables and fruits from the user input images. The recipe suggestions are filtered depending upon the user preferences. Thus, this application negates the user's dependency on food franchises as he/she can prepare recipes with his/her own raw materials and if possible learn new recipes. Thus the proposed system uses state-of-art techniques to use machines in more intuitive manner.

VI. REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. Proc. of the IEEE, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in Neural Information Processing Systems (NIPS), 2012.

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. IJCV, 59:167–181, 2004.

[4] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," International Journal of Computer Vision (IJCV), 2013.

[5] Fast R-CNN, Ross Girshick Microsoft Research, 2015, arXiv:1504.08083.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015

[7] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, 2015.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation". IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[9] Datasets: Open Images Dataset https://github.com/openimages/dataset

[10] Recipe: Recipes with Rating and Nutrition:https://www.kaggle.com/hugodarwood/epi recipes, Nutrition facts from foods around the world: https://www.kaggle.com/openfoodfacts/world-food-facts

[11] https://en.wikipedia.org/wiki/YUV#Y%E2%80%B2UV420sp_(NV21)_to_RGB_conversion_(Android)

Project Evaluation Sheet 2017 - 18

GROUP NO.: 51

Title of Project: Seefood - Food that you can see

Group Members: Himanshu Rawlani, Jagesh Saita, Vignesh Zambre (D17A)

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg & Mgmt principles (3)	Life - long learning (3)	Professional Skills (5)	Innovative Approach (5)	**Total Marks** (50)
Review of Project Stage 1	5	5	4	3	5	2	2	2	2	3	2	3	4	3	45

Comments: Good knowledge about algorithm & tools used.
→ Recipe prediction is remaining.

Name & Signature Reviewer1

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg & Mgmt principles (3)	Life - long learning (3)	Professional Skills (5)	Innovative Approach (5)	**Total Marks** (50)
Review of Project Stage 1	5	5	4	3	5	2	2	2	2	3	2	2	4	4	45

Comments: Receipe prediction is not done, Training to less no of objects, Good knowledge about algo & tools used.

Date: 26/2/18

Rashmi Patel

Name & Signature Reviewer2

Inhouse/ Industry:

Class: D17 A/B/C

Group No.: 51

Project Evaluation Sheet 2017 - 18

Title of Project: ~~Himanshu~~ Seefood - Food that you can see!

Group Members: Himanshu Rawlani (D17A-64), Jayesh Saita (D17A-66), Vignesh Tambre (D17A-78)

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg &Mgmt principles (3)	Life - long learning (3)	Professional Skills (5)	Innovative Approach (5)	Total Marks (50)
Review of Project Stage 1	5	5	4	3	4	2	2	2	2	3	3	2	5	4	45
Comments:															

Name & Signature Reviewer1

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg &Mgmt principles (3)	Life - long learning (3)	Professional Skills (5)	Innovative Approach (5)	Total Marks (50)
Review of Project Stage 1	4	4	5	3	4	2	2	2	2	3	3	3	4	4	45
Comments:															

Date: 15th March,2018

Name & Signature Reviewer2

Plagiarism Report:

Plagiarism Scan Report

Summary	
Report Genrated Date	21 Apr, 2018
Plagiarism Status	**100% Unique**
Total Words	440
Total Characters	2853
Any Ignore Url Used	

Content Checked For Plagiarism:

This paper describes an android application that suggests recipes by scanning raw vegetables and fruits using smartphone camera. The user points the camera at the

Plagiarism Scan Report

Summary	
Report Genrated Date	21 Apr, 2018
Plagiarism Status	**100% Unique**
Total Words	488
Total Characters	3070
Any Ignore Url Used	

Content Checked For Plagiarism:

The proposed system is an android application that uses Tiny-YOLO, the smaller and fast version of YOLO. It was made specifically for embedded devices with low computational

Plagiarism Scan Report

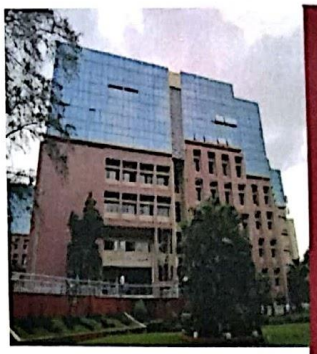
Summary	
Report Genrated Date	21 Apr, 2018
Plagiarism Status	**100% Unique**
Total Words	841
Total Characters	5219
Any Ignore Url Used	

Content Checked For Plagiarism:

Transfer learning is being performed on Tiny YOLO model, which is pre-trained on COCO dataset as the base model, on our own dataset of vegetables and fruits which has 13

RAIT project competition certificates:

D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI



National Level Project Competition
Srijan-2018

Certificate of Participation

This is to certify that Mr./Ms. HIMANSHU RAWLANI has participated and presented the project titled SEEFOOD - RECIPE SUGGESTION USING DEEP LEARNING in *"National Level Project Competition Srijan-2018"* conducted under Technovate-2018 .

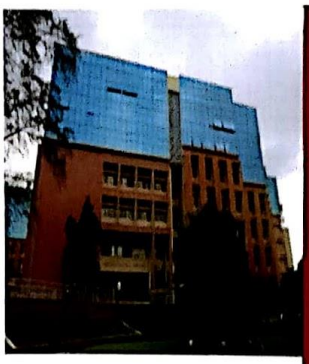
This competition was organized by Ramrao Adik Institute of Technology on April 4th, 2018.

Dr. Leena Ragha
Coordinator

Dr. Ramesh Vasappanavara, PhD.
Principal, RAIT

www.rait.ac.in

D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI



National Level Project Competition

Srijan-2018

Certificate of Participation

This is to certify that Mr./Ms. JAYESH SAITA has participated and presented the project titled SEEFOOD - RECIPE SUGGESTION USING DEEP LEARNING in *"National Level Project Competition Srijan-2018"* conducted under Technovate-2018 .

This competition was organized by Ramrao Adik Institute of Technology on April 4th, 2018.

Dr. Leena Ragha
Coordinator

Dr. Ramesh Vasappanavara, PhD.
Principal, RAIT

www.rait.ac.in

D Y PATIL
RAMRAO ADIK INSTITUTE OF TECHNOLOGY
NAVI MUMBAI

National Level Project Competition
Srijan-2018

Certificate of Participation

This is to certify that Mr./Ms. VIGNESH ZAMBARE has participated and presented the project titled SEEFOOD-RECIPE SUGGESTION USING DEEP LEARNING in *"National Level Project Competition Srijan-2018"* conducted under Technovate-2018 .

This competition was organized by Ramrao Adik Institute of Technology on April 4th, 2018.

Dr. Leena Ragha
Coordinator

Dr. Ramesh Vasappanavara, PhD.
Principal, RAIT

www.rait.ac.in