CSE 4020
MACHINE LEARNING
PROJECT REPORT

FOOD CLASSIFIER AND CALORIE INTAKE RECOMMENDATION APP

TEAM MEMBERS:

AISHWARYA S	**15BCB0055**
G LAKSHAYE VAIKUNTH	**15BCE0069**
VINEET NALAWADE	**15BCE0284**
NAMRATHA GONUGUNTLA	**15BCE0080**

Slot: E2

Google Drive Link:

https://drive.google.com/open?id=1Ne_-WO9pV17WcmMc_4j2JKLGnFlqrUu0

Submitted To
Prof. MANOOV R
ASSISTANT PROFESSOR(SENIOR)
School of Computer Science and Engineering

ABSTRACT

Automatic food image recognition systems are alleviating the process of food-intake estimation and dietary assessment. However, due to the nature of food images, their recognition is a particularly challenging task, which is why traditional approaches in the field have achieved a low classification accuracy.

Deep neural networks have outperformed such solutions, and we present a novel approach to the problem of food detection through which we can give the calories associated with each food and give recommendations through a Chabot based on food intake.

INTRODUCTION

Exact estimation of dietary caloric intake is essential for surveying the viability of weight reduction mediations. Current techniques for dietary evaluation depend on self-report and physically recorded instruments and food frequency surveys in spite of the fact that the 24-hour dietary review is the gold standard for announcing, this technique still encounters inclination as the member is required to estimate their dietary admission (short and long haul). Evaluation of dietary admission by the member can bring about underreporting and thinking little of food intake.

With a specific end goal to diminish member inclination and increment the precision of self-report, improvements are expected to supplement the present dietary reviews. One of the potential arrangements is a portable distributed computing framework, which is to utilize versatile processing gadgets (e.g., cell phone) to catch the dietary data in common living situations and to utilize the processing limit in the cloud to analyze the dietary data naturally for target dietary assessment.

While these applications have features to track food intake, exercise, and spare information in the cloud, the client needs to physically enter all their data. To defeat these obstructions, some innovative work endeavors have been made in the course of the most recent couple of years for visual-based dietary data examination While advances have been made, how to infer the nourishment data (e.g., sustenance type) from sustenance picture adequately and proficiently remains a testing and open look into issue.

In this project, we propose new profound learning-based sustenance picture acknowledgment calculation to address this test.

The proposed approach depends on Convolutional Neural Network (CNN) with a couple of significant enhancements. The exploratory consequences of applying the proposed way to deal with two true datasets have exhibited the viability of our answer.

LITERATURE SURVEY

[1] FooDD: Food Detection Dataset for Calorie Measurement Using Food Images (Parisa Pouladzadeh, Abdulsalam Yassine, Shervin Shirmohammadi): This paper presents the food image dataset. Furthermore, they provide examples of food detection using graph cut segmentation and deep learning algorithms and the dataset can aid further research on different types of food recognition and learning algorithms.

[2] DeepFood: Deep Learning-based Food Image Recognition for Computer-aided Dietary Assessment(Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane): They proposed a new Convolutional Neural Network (CNN)-based food image recognition algorithm to address this problem. We applied our proposed approach to two real-world food image data sets (UEC-256 and Food-101) and achieved impressive results. To the best of our knowledge, these results outperformed all other reported work using these two data sets.

[3] Caloric and Nutritional Information Using Image Classification of Restaurant Food(Arne Bech): This SVM based system shows that we get significant accuracy in image classification when supplemented with location information, and in this case it would do very well classifying fast food before sending it off to an operator.

[4] Im2Calories: Towards an automated mobile vision food diary (Austin Myers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban): The Im2Calories problem is also very interesting from the point of view of computer vision research. In particular, it requires solving various problems, such as: fine-grained recognition (to distinguish subtly different forms of food); hierarchical label spaces (to handle related labels); open-world recognition (to handle an unbounded number of class names); visual attribute recognition (to distinguish fried vs baked, or with or without mayo); instance segmentation; instance counting; amodal completion of occluded shapes.

[5] Insights from machine learned diet success prediction (Ingmar Weber, Palakorn Achananuparp): This paper presents a study that uses public food diaries of more than 4,000 long-term active MFP users. going above). Their results prove the feasibility of mining such data for health-related analysis. Especially with additional links to users' activity and general lifestyle patterns, automatically generated personalized and adaptive dieting seem a promising avenue to pursue. Health informatics is only starting to use the veritable gold mine that comes with public quantified-self data. The paper contributes to advances in this field by exploring how public food diaries can be mined to understand differences in unsuccessful and successful diets.

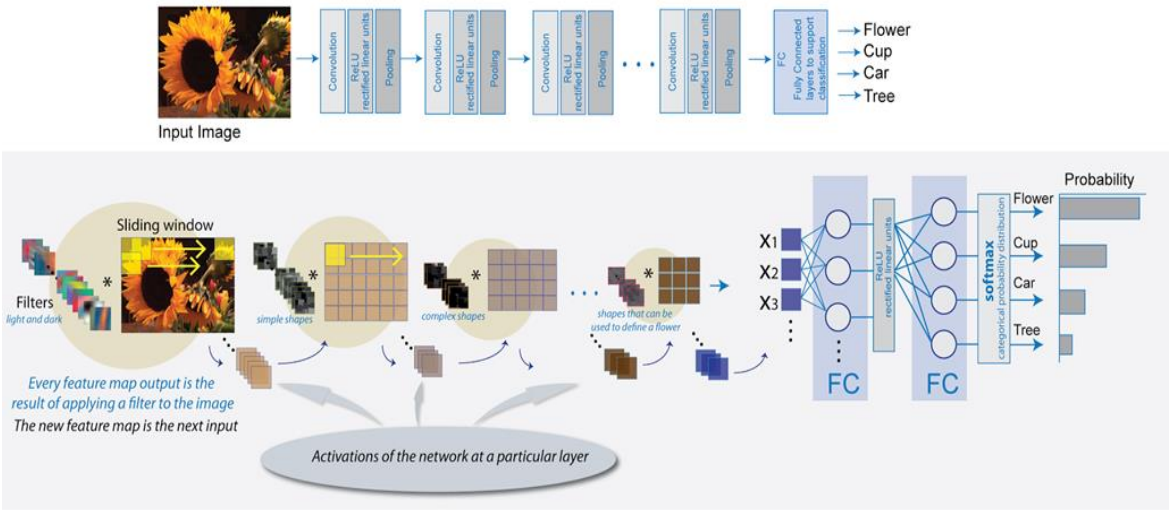
PROPOSED SYSTEM

- The main objective of our project was to create an application that would help facilitate a user with his or her diet.
- The application would do this by asking the user questions about his or her age, height, current weight, activity level and goal weight and use this information to set a specific amount of daily consumable calories for the user depending on if he/she wants to lose weight, gain weight or maintain the same weight.
- The user would keep track of these calories by using the application's food recognition software which allows a user to scan a food item that he or she is about to eat and then recognizes this food item through camera, based on Food detection dataset (Food 101 dataset).
- The app then Classifies the calories of a typical serving size of that food item associated with it based on Food nutrition and information dataset.
- The app then uses IBM Chabot which gives personalized recommendation based on the user's intake.

METHODOLOGY

1.Image Recognition using Machine Learning

- Used a framework called Tensor Flow to train a convolutional neural network (CNN).
- A CNN consists of multiple layers including a convolutional layer, pooling layer, ReLU layer, fully connected layer and loss layer.
- Each of these layers perform a function that alters the image being passed through the model in a certain way.
- The training is done using the Food 101 dataset.



2. We chose a convolutional neural network because the architecture of a CNN is the best for image recognition tasks. CNN architectures are inspired by biological processes and include variations of multilayer preceptors that result in minimal amounts of preprocessing. In a CNN, there are multiple layers that each have distinct functions to help us recognize an image.

3. The convolutional layer acts as the core of any CNN. The network of a CNN develops a 2-dimensional activation map that detects the special position of a feature at all the given spatial positions which are set by the parameters.

4.The pooling layer acts as a form of down sampling. Max Pooling is the most common implementation of pooling. Max Pooling is ideal when dealing with smaller data sets which is why we are choosing to use it

5.The ReLU layer or the Rectified Linear Units layer is a layer of neurons which applies an activation function to increase the nonlinear properties of the decision function

and of the overall network without affecting the receptive fields of the convolutional layer itself.

6.The Fully Connected Layer, which occurs after several convolutional and max pooling layers, does the high-level reasoning in the neural network. Neurons in this layer have connections to all the activations amongst the precious layers. After, the activations for the Fully Connected layer are computed by a matrix multiplication and a bias offset.

7.The Loss layer specifies how the network training penalizes the deviation between the predicted and true layers. We believe that Softmax Loss is the best for our project as this is ideal for detecting a single class in a set of mutually exclusive classes.

8. The Classification of the calories of a typical serving size of that food item associated with it is done based on Food nutrition and information dataset.

9.Firebase was used to store the user credentials and the datasets which have been used for training, testing and classification.

10.The IBM Watson Chatbot was used to give recommendations to the user based on the intake analysis and help the user achieve his diet plan.

DESCRIPTION OF THE DATASET

- **Food Images (Food-101) dataset:** Labeled food images in 101 categories from apple pies to waffles.

Source: https://www.kaggle.com/kmader/food41

The dataset contains a number of different subsets of the full food-101 data. The idea is to make a more exciting simple training set for image analysis than CIFAR10 or MNIST. For this reason, the data includes massively downscaled versions of the images to enable quick tests. The data has been reformatted as HDF5 and specifically Keras HDF5Matrix which allows them to be easily read in. The file names indicate the contents of the file.
For example,

- Food_c101_n1000_r384x384x3.h5 means there are 101 categories represented, with n=1000 images, that have a resolution of 384x384x3 (RGB, uint8)

- food_test_c101_n1000_r32x32x1.h5 means the data is part of the validation set, has 101 categories represented, with n=1000 images, that have a resolution of 32x32x1 (float32 from -1 to 1)

- **Food Nutrition Information Dataset: Nutritional information for raw fruits, vegetables, and seafood.**

Source: https://data.world/adamhelsinger/food-nutrition-information

Fruit:
Raw, edible weight portion. Percent Daily Values (%DV) are based on a 2,000 calorie diet.\

Vegetables:
Raw, edible weight portion. Percent Daily Values (%DV) are based on a 2,000 calorie diet.

Seafood:
Cooked (by moist or dry heat with no added ingredients), edible weight portion. Percent Daily Values (%DV) are based on a 2,000 calorie diet.

ALGORITHMS DEPLOYED

Algorithms deployed are decision tree, Naive Bayes and deep neural networks.

Decision tree

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute. This process is then repeated for the subtree rooted at the new node.

Naïve Bayes

It uses Bayes theorem to calculate the membership probability of an instance for a particular class. The Instance is then assigned to the class for which it is having the highest probability. Naïve Bayes assumes that all the features are independent of each other i.e. presence or absence of any feature will not affect the presence or absence of any other feature.

Deep neural networks

Neural networks are a set of algorithms, modelled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labelling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated. Neural networks also help us cluster and classify.

TECHNOLOGY STACK USED

Android studio (JAVA):

- Was used to create the user interface and the front-end of our application.
- Was used to connect all the components of our application together.

Tensor Flow:

-Tensor Flow is an open-source software library for dataflow programming across a range of tasks.

-It is a symbolic math library, and is also used for machine learning applications such as neural networks.

-Labelling, Classification and Training is done using tensor flow.

IBM Watson:

- Was used to provide recommendations to the user based on the user's intake and help him achieve his diet plan.

-All the analysis regarding the recommendation is done in IBM Watson cloud.

Firebase:

-The firebase acts as the backend and stores all the necessary datasets which is crucial for processing.

CODE

Image Classification and Calorie anlayis

```
public class Recognition {
 /**
  * A unique identifier for what has been recognized. Specific to the class, not the instance of
  * the object.
  */
 private final String id;

 /**
  * Display name for the recognition.
  */
 private final String title;

 /**
  * A sortable score for how good the recognition is relative to others. Higher should be better.
  */
 private final Float confidence;

 /** Optional location within the source image for the location of the recognized object. */
 private RectF location;
 private double calories;
 public Recognition(
  final String id, final String title, final Float confidence, final RectF location) {
  this.id = id;
  this.title = title;
  setCalories(this.title);
  this.confidence = confidence;
  this.location = location;
 }

 public String getId() {
  return id;
 }

 public String getTitle() {
  return title;
 }

 public Float getConfidence() {
  return confidence;
 }
```

```
public RectF getLocation() {
  return new RectF(location);
}

public void setLocation(RectF location) {
  this.location = location;
}

public void setCalories(String t){
    if(t.equals("apple")){
        this.calories=50;
    }else if(t.equals("banana")){
        this.calories=30;
    }else if(t.equals("broccoli")){
        this.calories=40;
    }else if(t.equals("chips")){
        this.calories=120;
    }else if(t.equals("cookies")){
        this.calories=150;
    }else if(t.equals("pizza")){
        this.calories=200;
    }else if(t.equals("soda can")){
        this.calories=80;
    }else{
        this.calories=0;
    }
}

public double getCalories(){return calories;}

@Override
public String toString() {
  String resultString = "";
  if (id != null) {
    resultString += "[" + id + "] ";
  }

  if (title != null) {
    resultString += title + " ";
  }

  if (confidence != null) {
    resultString += String.format("(%.1f%%) ", confidence * 100.0f);
  }
```

```
      if (location != null) {
        resultString += location + " ";
      }

      return resultString.trim();
    }
  }

  List<Recognition> recognizeImage(Bitmap bitmap);

  void enableStatLogging(final boolean debug);

  String getStatString();

  void close();
}
public void onPreviewSizeChosen(final Size size, final int rotation) {
    final float textSizePx =
        TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP, getResources().getDisplayMetrics());
    borderedText = new BorderedText(textSizePx);
    borderedText.setTypeface(Typeface.MONOSPACE);

    classifier =
        TensorFlowImageClassifier.create(
            getAssets(),
            MODEL_FILE,
            LABEL_FILE,
            INPUT_SIZE,
            IMAGE_MEAN,
            IMAGE_STD,
            INPUT_NAME,
            OUTPUT_NAME);

    resultsView = (ResultsView) findViewById(R.id.results);
    previewWidth = size.getWidth();
    previewHeight = size.getHeight();

    final Display display = getWindowManager().getDefaultDisplay();
    final int screenOrientation = display.getRotation();

    LOGGER.i("Sensor orientation: %d, Screen orientation: %d", rotation, screenOrientation);

    sensorOrientation = rotation + screenOrientation;
```

```
  LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
  rgbBytes = new int[previewWidth * previewHeight];
  rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight, Config.ARGB_8888);
  croppedBitmap = Bitmap.createBitmap(INPUT_SIZE, INPUT_SIZE, Config.ARGB_8888);

  frameToCropTransform =
      ImageUtils.getTransformationMatrix(
          previewWidth, previewHeight,
          INPUT_SIZE, INPUT_SIZE,
          sensorOrientation, MAINTAIN_ASPECT);

  cropToFrameTransform = new Matrix();
  frameToCropTransform.invert(cropToFrameTransform);

  yuvBytes = new byte[3][];

  addCallback(
      new DrawCallback() {
        @Override
        public void drawCallback(final Canvas canvas) {
          renderDebug(canvas);
        }
      });
}

@Override
public void onImageAvailable(final ImageReader reader) {
  Image image = null;

  try {
    image = reader.acquireLatestImage();

    if (image == null) {
      return;
    }

    if (computing) {
      image.close();
      return;
    }
    computing = true;

    Trace.beginSection("imageAvailable");

    final Plane[] planes = image.getPlanes();
```

```
  fillBytes(planes, yuvBytes);

  final int yRowStride = planes[0].getRowStride();
  final int uvRowStride = planes[1].getRowStride();
  final int uvPixelStride = planes[1].getPixelStride();
  ImageUtils.convertYUV420ToARGB8888(
      yuvBytes[0],
      yuvBytes[1],
      yuvBytes[2],
      previewWidth,
      previewHeight,
      yRowStride,
      uvRowStride,
      uvPixelStride,
      rgbBytes);

  image.close();
} catch (final Exception e) {
  if (image != null) {
    image.close();
  }
  LOGGER.e(e, "Exception!");
  Trace.endSection();
  return;
}

rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);
final Canvas canvas = new Canvas(croppedBitmap);
canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);

// For examining the actual TF input.
if (SAVE_PREVIEW_BITMAP) {
  ImageUtils.saveBitmap(croppedBitmap);
}

runInBackground(
    new Runnable() {
      @Override
      public void run() {
        final long startTime = SystemClock.uptimeMillis();
        final List<Classifier.Recognition> results = classifier.recognizeImage(croppedBitmap);
        lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

        cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
        resultsView.setResults(results);
```

```
          requestRender();
          computing = false;
        }
       });

    Trace.endSection();
  }

public class RecognitionScoreView extends View implements ResultsView {
    private static final float TEXT_SIZE_DIP = 25;
    private List<Recognition> results;
    private final float textSizePx;
    private final Paint fgPaint;
    private final Paint bgPaint;

    public RecognitionScoreView(final Context context, final AttributeSet set) {
      super(context, set);

      textSizePx =
          TypedValue.applyDimension(
              TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP, getResources().getDisplayMetrics());
      fgPaint = new Paint();
      fgPaint.setTextSize(textSizePx);

      bgPaint = new Paint();
      bgPaint.setColor(0xFF9999a3);
    }

    @Override
    public void setResults(final List<Recognition> results) {
      this.results = results;
      postInvalidate();
    }
    public List<Recognition> getResults(){
      return this.results;
    }

    @Override
    public void onDraw(final Canvas canvas) {
      final int x = 10;
      int y = (int) (fgPaint.getTextSize() * 1.5f);

      canvas.drawPaint(bgPaint);
```

```
      if (results != null) {
         for (final Recognition recog : results) {

            if (recog.getTitle().equals("apple")) {
               String calories = "95";

               canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
               y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("banana")) {
               String calories = "30";

               canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
               y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("water bottle")) {
               String calories = "0";

               canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
               y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("broccoli")) {
               String calories = "40";

               canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
               y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("chips")) {
               String calories = "120";

               canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
               y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("cookies")) {
               String calories = "150";
```

```
                canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
                y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("pizza")) {
                String calories = "200";

                canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
                y += fgPaint.getTextSize() * 1.5f;
            }

            if (recog.getTitle().equals("soda can")) {
                String calories = "80";

                canvas.drawText("This " + recog.getTitle() + " has " + calories + " calories",/*": " +
recog.getConfidence(),*/ x, y, fgPaint);
                y += fgPaint.getTextSize() * 1.5f;
            }
        }
    }
  }

public static Classifier create(
    AssetManager assetManager,
    String modelFilename,
    String labelFilename,
    int inputSize,
    int imageMean,
    float imageStd,
    String inputName,
    String outputName) {
  TensorFlowImageClassifier c = new TensorFlowImageClassifier();
  c.inputName = inputName;
  c.outputName = outputName;

  // Read the label names into memory.
  // TODO(andrewharp): make this handle non-assets.
  String actualFilename = labelFilename.split("file:///android_asset/")[1];
  Log.i(TAG, "Reading labels from: " + actualFilename);
  BufferedReader br = null;
  try {
    br = new BufferedReader(new InputStreamReader(assetManager.open(actualFilename)));
```

```
  String line;
  while ((line = br.readLine()) != null) {
   c.labels.add(line);
  }
  br.close();
 } catch (IOException e) {
  throw new RuntimeException("Problem reading label file!" , e);
 }

 c.inferenceInterface = new TensorFlowInferenceInterface(assetManager, modelFilename);

 // The shape of the output is [N, NUM_CLASSES], where N is the batch size.
 final Operation operation = c.inferenceInterface.graphOperation(outputName);
 final int numClasses = (int) operation.output(0).shape().size(1);
 Log.i(TAG, "Read " + c.labels.size() + " labels, output layer size is " + numClasses);

 // Ideally, inputSize could have been retrieved from the shape of the input operation.  Alas,
 // the placeholder node for input in the graphdef typically used does not specify a shape, so it
 // must be passed in as a parameter.
 c.inputSize = inputSize;
 c.imageMean = imageMean;
 c.imageStd = imageStd;

 // Pre-allocate buffers.
 c.outputNames = new String[] {outputName};
 c.intValues = new int[inputSize * inputSize];
 c.floatValues = new float[inputSize * inputSize * 3];
 c.outputs = new float[numClasses];

 return c;
}

@Override
public List<Recognition> recognizeImage(final Bitmap bitmap) {
 // Log this method so that it can be analyzed with systrace.
 Trace.beginSection("recognizeImage");

 Trace.beginSection("preprocessBitmap");
 // Preprocess the image data from 0-255 int to normalized float based
 // on the provided parameters.
 bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0, bitmap.getWidth(), bitmap.getHeight());
 for (int i = 0; i < intValues.length; ++i) {
  final int val = intValues[i];
  floatValues[i * 3 + 0] = (((val >> 16) & 0xFF) - imageMean) / imageStd;
  floatValues[i * 3 + 1] = (((val >> 8) & 0xFF) - imageMean) / imageStd;
```

```
  floatValues[i * 3 + 2] = ((val & 0xFF) - imageMean) / imageStd;
 }
 Trace.endSection();

 // Copy the input data into TensorFlow.
 Trace.beginSection("feed");
 inferenceInterface.feed(inputName, floatValues, 1, inputSize, inputSize, 3);
 Trace.endSection();

 // Run the inference call.
 Trace.beginSection("run");
 inferenceInterface.run(outputNames, logStats);
 Trace.endSection();

 // Copy the output Tensor back into the output array.
 Trace.beginSection("fetch");
 inferenceInterface.fetch(outputName, outputs);
 Trace.endSection();

 // Find the best classifications.
 PriorityQueue<Recognition> pq =
    new PriorityQueue<Recognition>(
       3,
       new Comparator<Recognition>() {
         @Override
         public int compare(Recognition lhs, Recognition rhs) {
           // Intentionally reversed to put high confidence at the head of the queue.
           return Float.compare(rhs.getConfidence(), lhs.getConfidence());
         }
       });
 for (int i = 0; i < outputs.length; ++i) {
  if (outputs[i] > THRESHOLD) {
    pq.add(
       new Recognition(
          "" + i, labels.size() > i ? labels.get(i) : "unknown", outputs[i], null));
  }
 }
 final ArrayList<Recognition> recognitions = new ArrayList<Recognition>();
 int recognitionsSize = Math.min(pq.size(), MAX_RESULTS);
 for (int i = 0; i < recognitionsSize; ++i) {
  recognitions.add(pq.poll());
 }
 Trace.endSection(); // "recognizeImage"
 return recognitions;
}
```

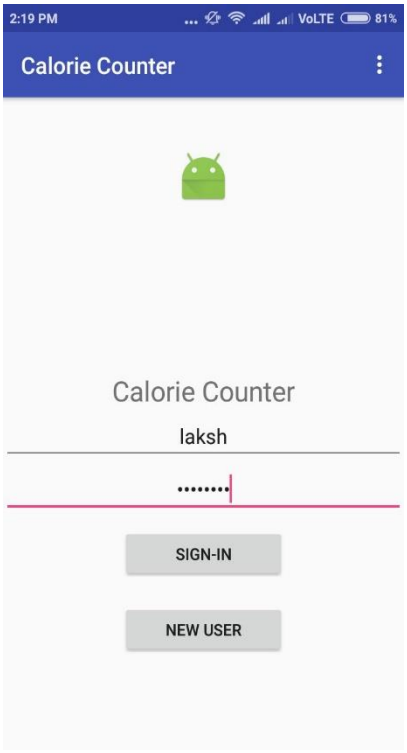
IBM Watson Recommender

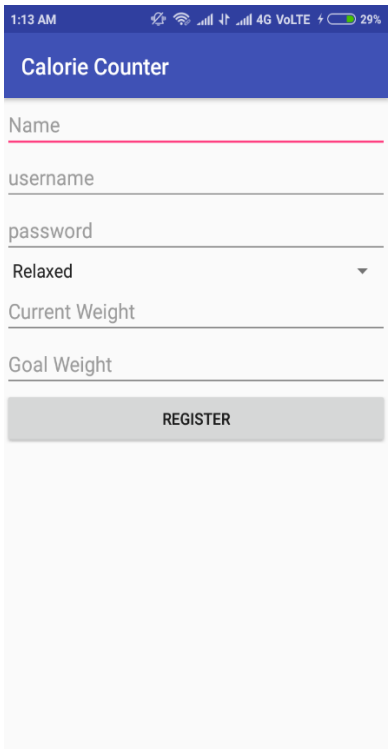
```
public class ConversationActivity extends AppCompatActivity {

    public ChatView mChatView;
    boolean firstTime;
    MessageResponse response;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_conversation);

        // init
        final Conversation service = new Conversation(Conversation.VERSION_DATE_2017_05_26);
        service.setUsernameAndPassword("80ec54cb-14e5-4a6f-8611-8f6521da6708",
"YcgYHANY4ioI");
        firstTime = true;

        //User id
        int myId = 0;
        //User icon
        Bitmap myIcon = BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
        //User name
        String myName = Tools.unFormatEmail(getIntent().getStringExtra("email"));

        int yourId = 1;
        Bitmap yourIcon = BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
        String yourName = "Calorie Bot";

        final User me = new User(myId, myName, myIcon);
        final User you = new User(yourId, yourName, yourIcon);

        mChatView = (ChatView) findViewById(R.id.chat_view);

        //Set UI parameters if you need
        mChatView.setRightBubbleColor(getColor(R.color.green500));
        mChatView.setLeftBubbleColor(Color.WHITE);
        mChatView.setBackgroundColor(getColor(R.color.blueGray500));
        mChatView.setSendButtonColor(getColor(R.color.cyan900));
        mChatView.setSendIcon(R.drawable.ic_action_send);
        mChatView.setRightMessageTextColor(Color.WHITE);
        mChatView.setLeftMessageTextColor(Color.BLACK);
        mChatView.setUsernameTextColor(Color.WHITE);
        mChatView.setSendTimeTextColor(Color.WHITE);
```

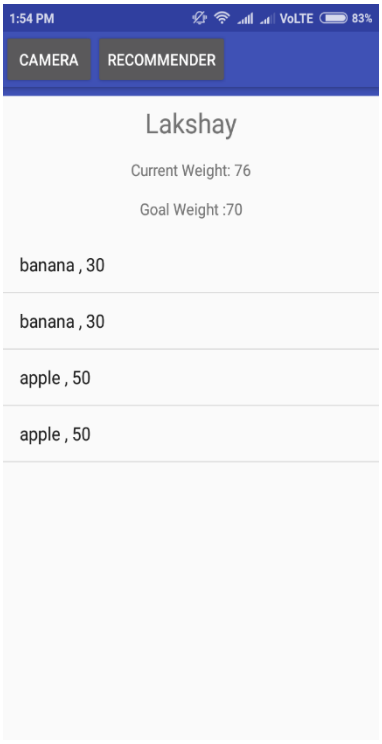
```
    mChatView.setDateSeparatorColor(Color.WHITE);
    mChatView.setInputTextHint("Hit me up!");
    mChatView.setMessageMarginTop(5);
    mChatView.setMessageMarginBottom(5);

    //Click Send Button
    mChatView.setOnClickSendButtonListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            //new message
            Message message = new Message.Builder()
                    .setUser(me)
                    .setRightMessage(true)
                    .setMessageText(mChatView.getInputText())
                    .hideIcon(true)
                    .build();
            //Set to chat view
            mChatView.send(message);
            //Reset edit text

            // watson bot
            InputData input = new InputData.Builder(mChatView.getInputText()).build();
            MessageOptions newMessageOptions;
            /*if (firstTime) {
                newMessageOptions = new MessageOptions.Builder()
                        .workspaceId("abd8be92-fe92-444d-9b04-06e8c5e1f312")
                        .input(new InputData.Builder(mChatView.getInputText()).build())
                        .context(null)
                        .build();
            } else {*/
                newMessageOptions = new MessageOptions.Builder()
                        .workspaceId("8da6bb18-3536-4d4d-864e-1be77f79c291")
                        .input(new InputData.Builder(mChatView.getInputText()).build())
                        .build();
            //}
            mChatView.setInputText("");
            NetworkStuff networkStuff = new NetworkStuff(service, newMessageOptions, you);
            networkStuff.execute();
        }

    });

}
```

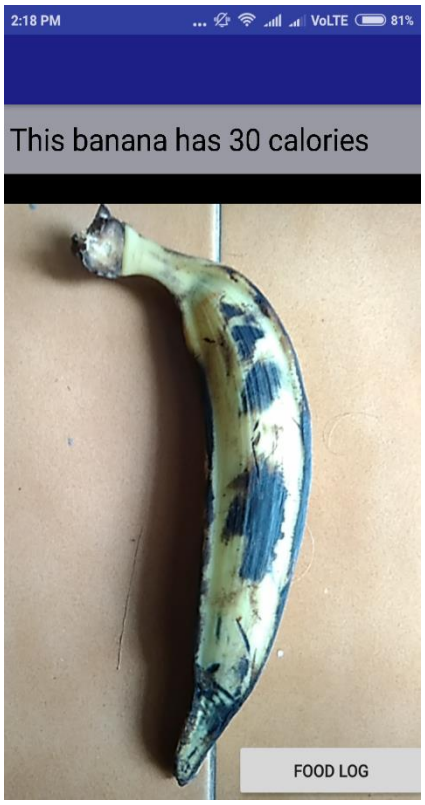
```
public class NetworkStuff extends AsyncTask<Void, Void, Void> {

    Conversation service;
    MessageOptions options;
    User you;

    public NetworkStuff(Conversation service, MessageOptions options, User you) {
      this.service = service;
      this.options = options;
      this.you = you;
    }

    @Override
    protected Void doInBackground(Void... voids) {
      response = service.message(options).execute();
      return null;
    }

    @Override
    protected void onPreExecute() {
      super.onPreExecute();
    }

    @Override
    protected void onPostExecute(Void aVoid) {
      super.onPostExecute(aVoid);

      try {
        JSONObject jsonObject = new JSONObject(response.toString());
        //Receive message
        final Message receivedMessage = new Message.Builder()
            .setUser(you)
            .setRightMessage(false)
            .setMessageText(new JSONArray(new
JSONObject(jsonObject.get("output").toString()).get("text").toString()).get(0).toString())
            .build();

        mChatView.receive(receivedMessage);
      } catch (JSONException e) {
        e.printStackTrace();
      }
    }
  }
}
```

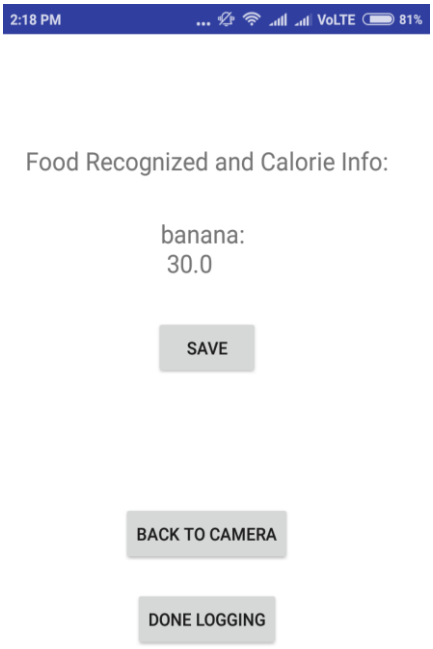
OUTPUT AND SCREENSHOTS

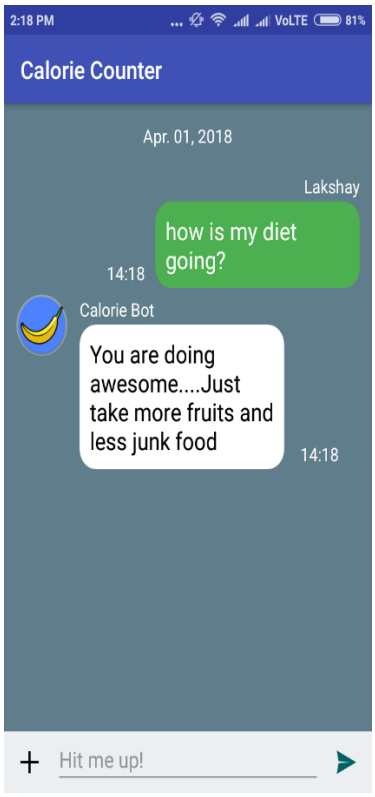


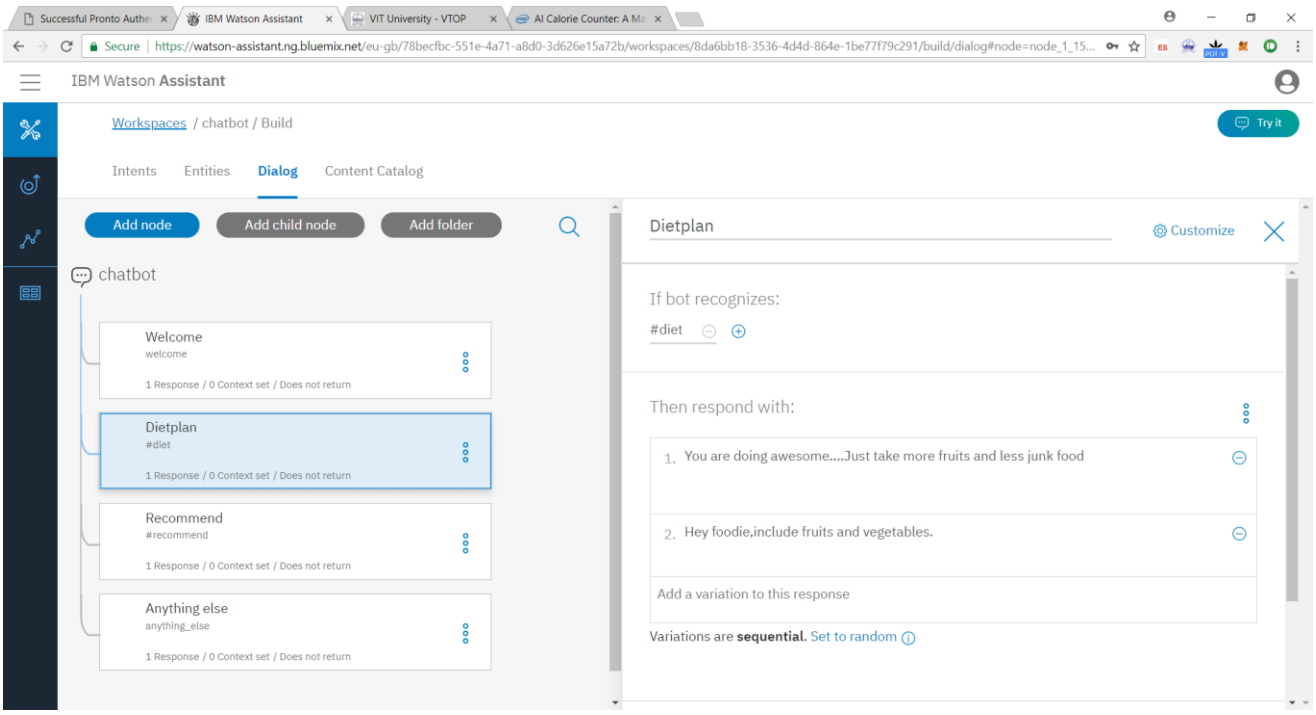





2:18 PM
VoLTE 81%
This banana has 30 calories
FOOD LOG


2:18 PM
VoLTE 81%
Food Recognized and Calorie Info:
banana:
30.0
SAVE
BACK TO CAMERA
DONE LOGGING


2:18 PM
VoLTE 81%
Calorie Counter
Apr. 01, 2018
Lakshay
how is my diet going?
14:18
Calorie Bot
You are doing awesome....Just take more fruits and less junk food
14:18
Hit me up!



CONCLUSION

Hence, we have successfully implemented the app that can be very helpful to people to solve their fitness issues the machine learning way.

FUTURE ENHANCEMENTS

- One feature we would like to improve upon is the calorie detector. Currently the number of calories being displayed is only the average amount of calories for the given food item. This method is not very accurate because a large apple would obviously contain more calories than a small apple. One way to improve this issue is by detecting the volume of the presented food item. There is a method called ‘space carving’ that detects the volume of 3D objects that we would like to look into and possibly implement in the future.
- Also it can be integrated with social networking applications like snapchat to attract the foodie users.

REFERENCES

- Glasgow, Russell E. et al. Twelve-month outcomes of an Internet- based diabetes self-management support program. Patient Education and Counseling, Sept, 2011
- Johnson, Thienne, Eleri Cardozo and Eliane Gomes Guimaraes. Pervasive Computing Applications, Technologies, and Challenges for E-Health. Designing Solutions-Based Ubiquitous and Pervasive Computing: New Issues and Trends. IGI Global, 2010. 232-247.
- Norman, Gregory J. et al. A Review of eHealth Interventions for Physical Activity and Dietary Behavior Change. American Journal of Preventive Medicine, 33(4), 336-345, 2007
- Franc, S. et al. Telemedicine and diabetes: Achievements and prospects. Diabetes and Metabolism 37, 2011, 463476
- Bouton, M.E. (2000). A learning-theory perspective on lapse, relapse, and the maintenance of behavior change. Health Psychology, 19, 5763.
- Epstein, L. H., Myers, M. D., Raynor, H. A., and Saelens, B. E. (1998). Treatment of pediatric obesity. Pediatrics, 101, 554570.
- https://data.world/adamhelsinger/food-nutrition-information
- http://www.site.uottawa.ca/~shervin/food/