FOOD RECOGNITION AND RECOMMENDATION APPLICATION USING MACHINE LEARNING

Prathmesh Godse, Chandan Kesarwani, Pravin Lohare, Rohan Kanwar, Prof. Deepali Patil

Student, Student, Student, Student, Project Guide
Information Technology,
Shree L.R. Tiwari College of Engineering, Mumbai, India

Abstract: **One of the emergent concerns of human life is about health and wellness. Undeniably, health and nutrition are one of the valuable aspects of life. Thus, technological innovations to help enhance and even promote health awareness is essential. With the advent of mobile computing, it is much easier to be aware of health information because of its mobility and availability. Much mobile application is being developed to serve as a tool for health monitoring and nutritional guide. Mobile applications have the ability to support health needs like detecting heart rate, classifying food, and many more. Taking advantage of technology, utilization of it hereby addresses certain issue and problems of human life, especially in health.**

In this project we attempt to design and develop an Android- based food recognition and recommendation application that could be used as a health advisor for every health group. The application lets the user take the photo of the food and recognizes it using deep learning and shows the nutritional value of the food and then applying classification algorithms on the gathered data, it will inform whether it is healthy for a particular group of people (like diabetic patients). We believe this will be an effective way to recognize the food category and efficiently analyze the nutritional value of it. Since it's a mobile- based application, it will be convenient for the users to access the application from wherever and whenever they wish to, and create awareness which will promote healthy lifestyle and better nutrition.

I. Introduction

The number of health-conscious people has grown rapidly in the past few years, thanks to the calorie counting applications which are widely available on smartphones. People try to record every meal they have and at the end of the day review how many calories did they consume and whether it is more than or less than the amount they should to consume. Although keeping track of the food items after a couple of days or so becomes tiring due to the fact that the user has to search for what s/he will consume, and it becomes even more so after finding out that the food item they are having does not exist in the applications database. FRRApp will help user ease the process by asking user to only click a picture of the food item. It will then manage the finding of what that food item is and provide with nutritional details as well as recommendation whether or not the user should avoid such food.

Keeping an accurate diet all the time is difficult and even more so knowing the exact amount of calorie intake. To overcome this problem applications were developed which helped the people to track their diet and maintain a record of their calorie intake. Even so, it becomes tedious to keep searching for the food item in the application every time the user has to eat. Not only that but if the user may know what the food item is called in his/her mother tongue but not in English or the food item does not exist in the applications data base, it discourages the user to keep maintaining the record in the future. In this project we aim to ease the process for users by recognizing the food for them using Convolutional Neural Network (CNN) and Machine Learning (ML). Our focus is to provide the user with a huge enough data base and fast algorithm so as to reduce the time and effort taken by them to know what nutritional values will the food item contain in it. The proposed system will help the user to quickly find nutritional information about the food items without the hassle of searching the food item. Not only that but also the system will provide the user with information other than the calorie count, like the protein, fat, calcium quantities. The FRRApp is intended to work as a standalone application where user can input picture of the food item, based on which system will process the request. System will provide user with the appropriate nutritional value.

Finally, the recommendations will allow user to decide whether the food item is a healthy choice.

II. Literature review

The paper [http://cs229.stanford.edu] suggests an approach to train the CNN model developed by Google named Inception VI with training data gathered from the UMPC- FOOD-101 and ETHZ-FOOD-101, which are twin data sets. They also suggest pre-processing the data because the environmental background varies quite a lot in different food pictures. Those environmental factors are the color temperatures, luminance and so on. To have similar background environment, they utilize two methods which are Grey World method and Histogram equalization. The white balance is processed by the Grey World Method which assumes the average of the RGB values are all similar to the one grey value. The Grey World algorithm is as followed, where α, β, γ are the scaling factors in the RGB color channels. Then, we applied Histogram Equalization algorithm to increase the contrast and luminance. The image pre- processing result is shown as below. The first one is the image of a baby rib. The middle one is the image after the white balance and the right one is the one after both white balance and histogram equalization. The paper [2] proposes the idea to recognize food as healthy or not for diabetic patients which is based on bag of features model. The proposed food recognition system consists of two stages: food image description and image classification. During food image description, a set of characteristics representing the visual content of the image is extracted and quantified. This set provides input to the second stage, where a classifier assigns to the image one class out of a predefined set of food classes. The design and development of both stages involves two phases: training and testing. During the training phase, the system learns from the acquired knowledge, while during the testing phase the system recognizes food types from new, unknown images.

III. DATASETS

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

1. food_c101_n1000_r384x384x3.h5 means there are 101 categories represented, with n=1000 images, that have a resolution of 384x384x3 (RGB, uint8)

2. 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)

The first goal is to be able to automatically classify an unknown image using the dataset, but beyond this there are a number of possibilities for looking at what regions / image components are important for making classifications, identify new types of food as combinations of existing tags, build object detectors which can find similar objects in a full scene..

IV. METHODOLOGY

We have designed the following algorithm which would be followed throughout the implementation phase of this system in a regular Neural Network there are three types of layers [3]:

1. Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels in case of an image).
2. Hidden Layer: The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.
3. Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or SoftMax which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss. Here's the basic python code for a neural network with random inputs and two hidden layers.

```
activation = lambda x: 1.0 / (1.0 + np.exp(-x)) # sigmoid function

    input = np.random.randn(3, 1)

    hidden_1 = activation (np.dot (W1, input) + b1) hidden_2 = activation (np.dot (W2, hidden_1) + b2) output = np.dot (W3,

hidden_2) + b3
```

Convolution Neural Networks or convnets are neural networks that share their parameters Convolution layers consist of a set of learnable filters (patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).

For example, if we have to run convolution on an image with dimension 34x34x3. Possible size of filters can be axax3, where 'a' can be 3, 5, 7, etc. but small as compared to image dimension. During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume. As we slide our filters, we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters. Layers used to build Convnets. A convnet is a sequence of layers, and every layer transform one volume to another through differentiable function.

Types of layers:

Let's take an example by running a covnets on ofimage of dimension 32 x 32 x 3.

Input Layer: This layer holds the raw input of image with width 32, height 32 and depth 3.

Convolution Layer: This layer computes the output volume by computing dot product between all filters and image patch. Suppose we use total 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.

Activation Function Layer: This layer will applyelement wise activation function to the output of convolution layer. Some common activation functions are RELU: max (0,x,) Sigmoid: 1/(1+e^-x), Tanh, LeakyRELU, etc. The volume remains unchanged hence output volume will have dimension 32 x 32 x 12.

Pool Layer: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents from overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

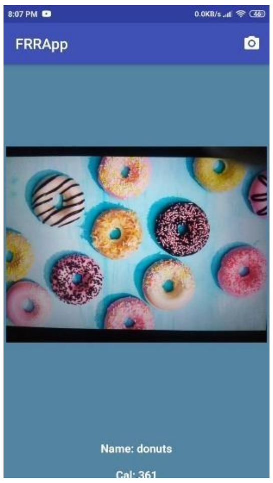
Fully-Connected Layer: This layer is regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

Libraries Required:

- TFLearn – Deep learning library featuring a higher-level API for TensorFlow used to create layers of our CNN
- tqdm – Instantly make your loops show a smart progress meter, just for simple designing sake
- NumPy – To process the image matrices
- open-cv – To process the image like converting them to grayscale and etc.
- os – To access the file system to read the image from the train and test directory from our machines
- random – To shuffle the data to overcome the biasing
- matplotlib – To display the result of our predictive outcome.
- TensorFlow – Just to use the tensorboard to compare the loss and adam curve our result data or obtained log.

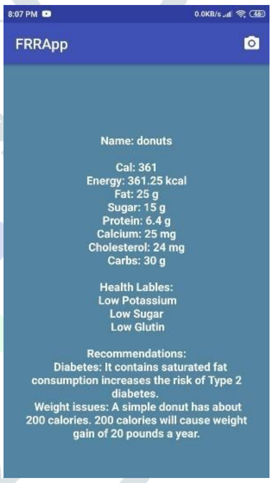
IV. RESULTS

The application provides the user with necessary nutritional information about the pictured food. Along with that, the application also provides labels which suggest what kind of food it is. The application works fast and requires internet connection to be able to return any nutritional, information about the food.



Name: donuts

Cal: 361



Name: donuts

Cal: 361
Energy: 361.25 kcal
Fat: 25 g
Sugar: 15 g
Protein: 6.4 g
Calcium: 25 mg
Cholesterol: 24 mg
Carbs: 30 g

Health Lables:
Low Potassium
Low Sugar
Low Glutin

Recommendations:
Diabetes: It contains saturated fat consumption increases the risk of Type 2 diabetes.
Weight issues: A simple donut has about 200 calories. 200 calories will cause weight gain of 20 pounds a year.

V. COLCLUSION

Health and Nutrition are crucial part of our lives and knowing what we consume is what helps us take care of ourselves even better. FRRApp allows us to be able to obtain that knowledge on the go. Even without understanding what exactly the values mean, one can understand what the food labels suggest and based on one's condition decide whether or not the consume the pictured food. FRRApp is a simple yet, effective means of spreading the awareness of healthy eating and should be utilized by as many people as possible.

REFERENCES

[1] http://cs229.stanford.edu/proj2016/report/YuMaoWang Deep%20Learning%20Based%20Food%20Recognition-report.pdf

[2] A Food Recognition System for Diabetic Patients Based on an Optimized Bag-of-Features Model Marios M. Anthimopoulos, Member, IEEE, Lauro Gianola, Luca Scarnato, Peter Diem, and Stavroula G. Mougiakakou, Member, IEEE.

[3] https://www.geeksforgeeks.org/introduction- convolution-neural-network