Food Image Recognition and Calorie Estimation

Abhitej Thonupunoori
attxv@umsystem.edu

Manoj Kumar Madireddy
mm68c@umsystem.edu

Aravind Kumar Gardas
agbnx@umsystem.edu

ABSTRACT

In this project, we have developed a food recognition model that predicts the food that are captured using the camera and also estimate its calorie, to help patients and dietitians in managing daily food intake.

INTRODUCTION

Food is crucial to the human body. Now a days, increasingly more people are concerned about their dietary intake because an unhealthy diet can lead to a variety of diseases. A diet plan should always consider the total number of calories to be consumed in order to live a fit and healthy life. Weight is an ailment that indicates an excess of muscle to fat ratio. If your BMI is greater than 30, you are obese. Weight gain can be caused by a variety of factors. One of these reasons is a high calorie expenditure. As a consequence of the vast application possibilities in human machine interface, systems based on vision-based interaction and control are becoming more common, and so a food recognition system which is vision-based interface would be more convenient, practical.

Finally, all that is necessary is an image capturing sensor that is independent of the user, such as a camera, infrared sensor, or depth sensor. This technique is the most practical since the user does not need to wear a heavy device in order to acquire enough identification accuracy and computing speed. It is vital for the infrastructure of any food recognition system to be practical. After all, we want to use it in real-world situations.

PROBLEM STATEMENT

The suggested framework trains different data set classes, which can be food object categories. The method is used to detect this food object. Keras is creating its own data set that will be trained using existing objects and used for evaluation. The proposed model's final phase involves calculating the individual food object class based on the input image.

RELATED WORK

CNNs have been more popular as a general-purpose computer vision tool due to their ability in object recognition and categorization. First, CNNs were used for video action and activity identification, and they have reached state-of-the-art performance in video analysis applications. CNNs have been used to extract spatiotemporal information from video footage in a variety of ways. Due to its performance in static pictures, 2D CNNs were first used in video analysis-based techniques. There are two-dimensional CNNs that use video frames as multi-channel input.

The Temporal Segment Network (TSN) uses 2D CNNs to separate video into segments, then employs spatio-temporal modeling to extract information from color and optical flow modalities for action identification. Video frames are first segmented into their individual characteristics using a 2D convolutional neural network (CNN), and then the LSTM algorithm is used to represent the overall temporal structure of the video. Because there are several viable 2D CNN designs that can be pretrained using the Kaggle dataset, the strength of all these techniques is derived.

Two-dimensional convolutional neural networks (2D CNNs) perform well on video analysis tasks, however they are restricted in their ability to understand motion patterns and temporal data. 3D CNNs that employ 3D convolutions and pooling to collect discriminative information along both spatial and temporal dimensions are the best choice for this task. 3D CNNs are different from 2D CNNs in that they use video frames as inputs instead of only one. Additionally, we use 3DCNN variations.

SOLUTION

To give a working solution, we created a food detection system that uses raw video data and deep convolutional neural networks (CNNs), CNNs now give cutting-edge results. In real-time food recognition applications, the system must meet specific requirements, such as (i) relatable accuracy for the classification, (ii) fast response time, (iii) Efficiency of resources. All these elements are required for an effective real-time vision-based food recognition system. The majority of past research, on the other hand, has focused primarily on enhancing food recognition classification accuracy while ignoring the remaining components.

DATASET

In this project, we have used food images data from Kaggle

https://www.kaggle.com/datasets/aelchimminut/fruits262

From the above dataset, we have obtained the food training and test images and the corresponding classifications.

Figure 1. Sample Food Images

IMPLEMENTATION

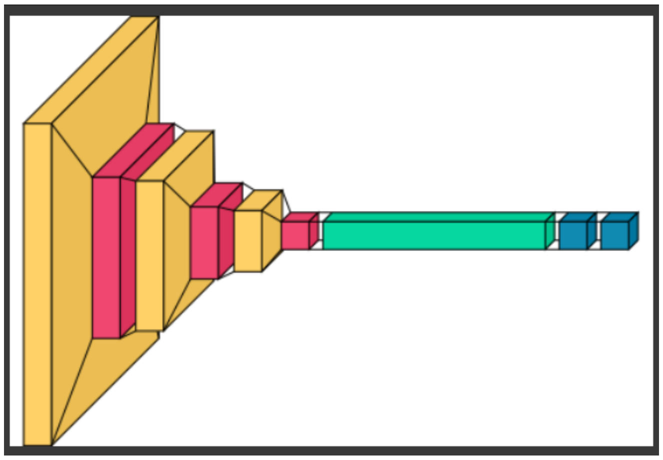
Pre-Processing

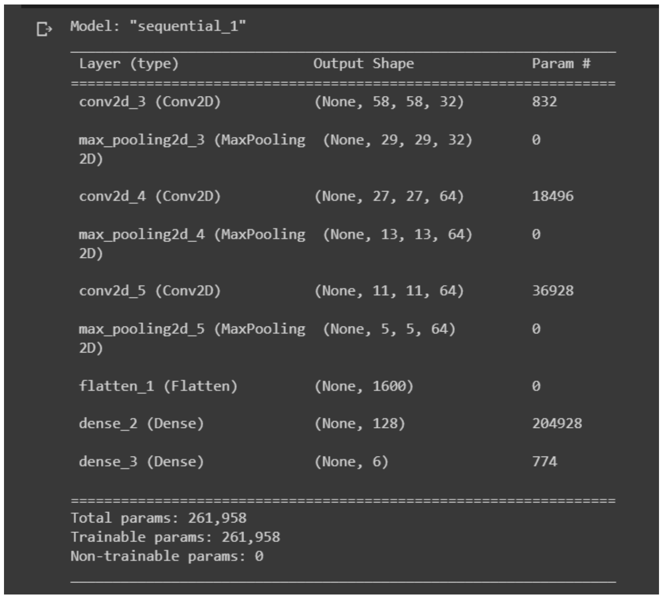
In the Pre-Processing stage, we have resized the images in the dataset to 120*120 size. Next, we have Defined the characters the key of the category dictionary is the folder name, and value is an integer. Then from each category we have taken 320 images for training and 80 images for testing. and then shuffled the training data using random.shuffle function. then we have Scaled the image pixel values between 0 and 1 for both training and testing data.

Model

After trying out different layers at different layers, we have converged at a model with the configuration as below and we have trained the model with above configuration using the training data obtained in the pre-processing stage. and saved the model to a file with.h5 extension, this file will be used in the prediction stage to load the model and predict the gesture of a new image.[1]

Once we have trained the model with the training data, we have evaluated our model using the test data and the model has an accuracy of 76.





```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 58, 58, 32)        832

 max_pooling2d_3 (MaxPooling  (None, 29, 29, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 27, 27, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 11, 11, 64)        36928

 max_pooling2d_5 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 1600)              0

 dense_2 (Dense)             (None, 128)               204928

 dense_3 (Dense)             (None, 6)                 774

=================================================================
Total params: 261,958
Trainable params: 261,958
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. Model Summary

Prediction/Application

We have built a python-flask application, which when called using an API, will call a method, where we read the image from the camera using open-CV library and followed the below steps in order to obtain the prediction of the food.

Identifying the Background

In order to identify the food in the image captured using the camera. we read the background image only for the first 30 frames and then take the average of those images, this average serves as reference background to distinguish the food in theimage.

Identifying the food

Once we have the reference background image data, for the subsequent frames we deduct the reference background image from the input image obtained from camera. Once the background is deducted, we obtain the food image in the input image.

Prediction

Once the food is identified in the above step, we use this food image captured and scale the image to that of the training, testing data sets that is 120*120, and use this image as input to model.predict function for predicting the food class. This process of prediction is done at every 100th frame which is the window size we have taken, as the input is a streaming data.

FUTURE WORK

Now that we are able to predict the food given by the end user using our application, we can extend the functionality so to predicting multiple types of food in a single image, calculate quantity, derive nutrition tables etc.

REFERENCES

https://machinelearningmastery.com/what-is-deep-learning/
https://brohrer.github.io/how_convolutional_neural_netwo rk s_work.html
https://keras.io/guides/sequential_model/
https://keras.io/api/models/
http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
https://www.jeremyjordan.me/nn-learning-rate/
https://keras.io/api/losses/probabilistic_losses/
https://keras.io/activations/
https://keras.io/optimizers
https://keras.io/losses/
https://keras.io/initializers/
https://keras.io/callbacks/
https://keras.io/metrics/
https://www.geeksforgeeks.org/underfitting-and-overfitting- in-machine-learning/
https://pylessons.com/CNN-tutorial-introduction/

REFERENCES

https://machinelearningmastery.com/what-is-deep-learning/
https://brohrer.github.io/how_convolutional_neural_netw ork s_work.html
https://keras.io/guides/sequential_model/
https://keras.io/api/models/
http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
https://www.jeremyjordan.me/nn-learning-rate/
https://keras.io/api/losses/probabilistic_losses/
https://keras.io/activations/
https://keras.io/optimizers/
https://keras.io/losses/
https://keras.io/initializers/
https://keras.io/callbacks/
https://keras.io/metrics/
https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
https://pylessons.com/CNN-tutorial-introduction/