Food Classification from Images Using Convolutional Neural Networks

Course Project Report

Submitted in partial fulfilment of the requirements for the

degree of

Master of Technology in

Computer Science and Engineering

Under the guidance of

Nevil Anto

172CS015 Kemanth PJ

Contents

Food Classification from Images Using Convolutional Neural Networks

1. Abstract

Recently, smart applications for mobile devices such as Android phones and iPhone, have increased tremendously. Due to the advances invarious technologies used in smartphones, their computational power has also increased. In the current age, people are more conscious about their food and diet. We can use this technology to help people classify differnt types of food and their health benefits. In this paper, an approach has been presented to classify images of food using convolutional neural networks. Unlike the traditional artificial neural networks, convolutional neural networks have the capability of estimating the score function directly from image pixels. There are multiple such layers, and the outputs are concatenated at parts to form the final tensor of outputs. We use MAX pooling to extract essential features from the images and use it to train the model. An accuracy of 86.97% for the classes of the FOOD-101 dataset is recognised using the proposed implementation.

2. Introduction

In the current age, people are more conscious about their food and diet to avoid either upcoming or existing diseases. Since people are dependent on smart technologies, provision of an application to automatically monitor the individuals diet, helps in many aspects. It increases the awareness of people in their food habits and diet. Recently, smart applications for mobile devices such as Android phones and iPhone, have increased tremendously. They are capable of balancing the food habits of users and also warn them about unhealthy food. Due to the advances in various technologies used in smartphones, their computational power has also increased. They are capable of processing real-time multi-media information with their computational power, whereas traditional mobiles are incapable and hence, used to send the images to high processing servers that increase the cost of communication and delay. Since the present smartphones can handle the high-quality images too, research on food classification is focused on developing real-time applications which capture images and train the machine learning models instantly. It helps to take prevention to avoids diseases such as diabetes, blood pressure and so on. Some of the methods currently in use for dietary assessment involve self-reporting and manually recorded instruments. The issue with such methods of assessment is that the evaluation of calorie consumption by a participant is prone to bias [6], i.e. underestimating and under reporting of food intake. In order to increase the accuracy and reduce the bias, enhancements to the current methods are required. One such potential solution is a mobile cloud computing system, which makes use of devices such as smartphones to capture dietary and

calorie information. The next step is to automatically analyse the dietary and calorie information employing the computing capacity of the cloud for an objective assessment. However, users still have to enter the information manually. Over the last few years, plenty of research and development efforts have been made in the field of visual-based dietary and calorie information analysis. However, the efficient extraction of information from food images remains a challenging issue. In this paper, an effort has been made to classify the images of food for further diet monitoring applications using convolutional neural networks (CNNs). Since the CNNs are capable of handling a large amount of data and can estimate the features automatically, they have been utilised for the task of food classification. The standard Food-101 dataset has been selected as the working database for this approach.

3. Related Work

The task of the food detection system is first initiated withfour fast-food classes namely fries, apple pies, hamburgersand chicken burgers. The images were segmented initiallyto form the feature vector with size, shape, texture, color(normalised RGB), and other context-based features. Withthis motivation, a minimised feature vector with the Gaborfilter responses (texture), pixel intensity, and color components is used to categorise the 19 classes of foods. However, theperformance is good for food replicas, and a less efficientperformance is observed with real images. The size ofimages and their variations in capturing could be the reasonfor the performance degradation. Based on this, scale invariantfeature transform (SIFT) features have been extracted andexperimented on homemade foods, fast-food, and fruits .With this, the better performance is found with less numberof classes, although the images of each class are more.

The term bag of features (BoF) which is derived from thebag of words (BoW) is the emerging trend in recent days.It is highly influenced to process the natural language. Itis designed to catch frequently appearing words by ignoringthe order in which they appear. Similarly, imagescontain some common visual patterns that are useful inrecognising the category of food. This process reducesthe complexity issues raised by the direct image matchingtechniques. Based on this, some works are found using theBoF approach.

Deep Convolutional Neural Networks have been used forfood recognition recently, which have used the UEC-100and UEC-256 datasets for testing, along with ImageNetand ILSVRC for training, which use a combination ofbaseline feature extraction and neural network fine-tuning.Another approach uses Convolutional Neural Networksalong with a Global Average Pooling layer, which generatesFood Activation Maps (heat maps of food probability). Finetuning is done for FAM generation, which includes adding aconvolutional layer with stride, and setting a softmax layer.Additionally, via thresholding, bounding boxes are generated.The present work aims to combine some of the abovemethodologies together, that creates a food classificationsystem, that predicts the class of food the image is in, and alsogives the calorie count based on the portion size visible. Thisconcept has a high scope in the health sector, as people want tokeep track of what and how much they eat and simplifying theprocess into the form of this implementation increases usageand awareness of health-related factors. Since CNNs are lessfocused in the literature, they have been utilized due to theirinherent capabilities in computing features automatically

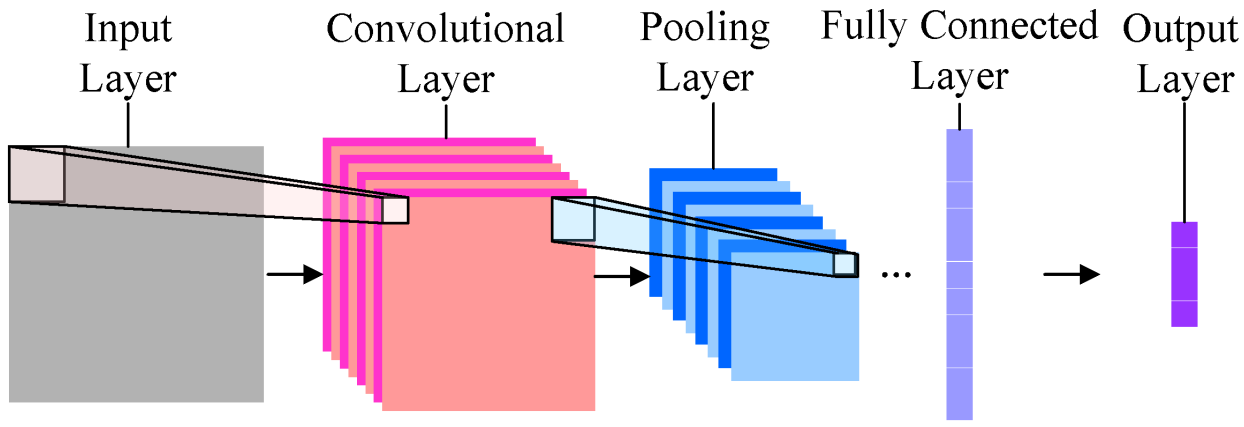
3. Food Classification using keras

Convolutional neural networks (CNN) have been widely used in automatic image classification systems. In most cases, features from the top layer of the CNN are utilized for classification; however, those features may not contain enough useful information to predict an image correctly. In some cases, features from the lower layer carry more discriminative power than those from the top. Therefore, applying features from a specific layer only to classification seems to be a process that does not utilize learned CNN's potential discriminant power to its full extent. This inherent property leads to the need for fusion of features from multiple layers. To address this problem, we propose a method of combining features from multiple layers in given CNN models. Moreover, already learned CNN models with training images are reused to extract features from multiple layers. The proposed fusion method is evaluated according to image classification benchmark data sets, CIFAR-10, NORB, and SVHN. In all cases, we show that the proposed method improves the reported performances of the existing models by 0.38%, 3.22% and 0.13%, respectively.

3.1 Convolutional Neural Networks

The Convolutional neural network(CNN) is a deep learning architecture that has numerous application in computer vision and natural language processing. The CNN classifies objects based on number of features matched. Steps involed in creating CNN

1. Convolution
2. Pooling
3. Flattening
4. Full Connection



- **Convolution**
  ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.
- **Pooling**

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. In case of Max Pooling, we define a spatial neighborhood (for example, a 2×2 window) and take the largest element from the rectified feature map within that window. Instead of taking the largest element we could also take the average (Average Pooling) or sum of all elements in that window. In practice, Max Pooling has been shown to work better.

- **Flattening**
  Convert the 2D matrix to a column vector so that it can passed through an artificial neural network
- **Full Connection**
  The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

4. Implementation of Food Classification using keras

First we need to import the libraries required by the application

In [1]:

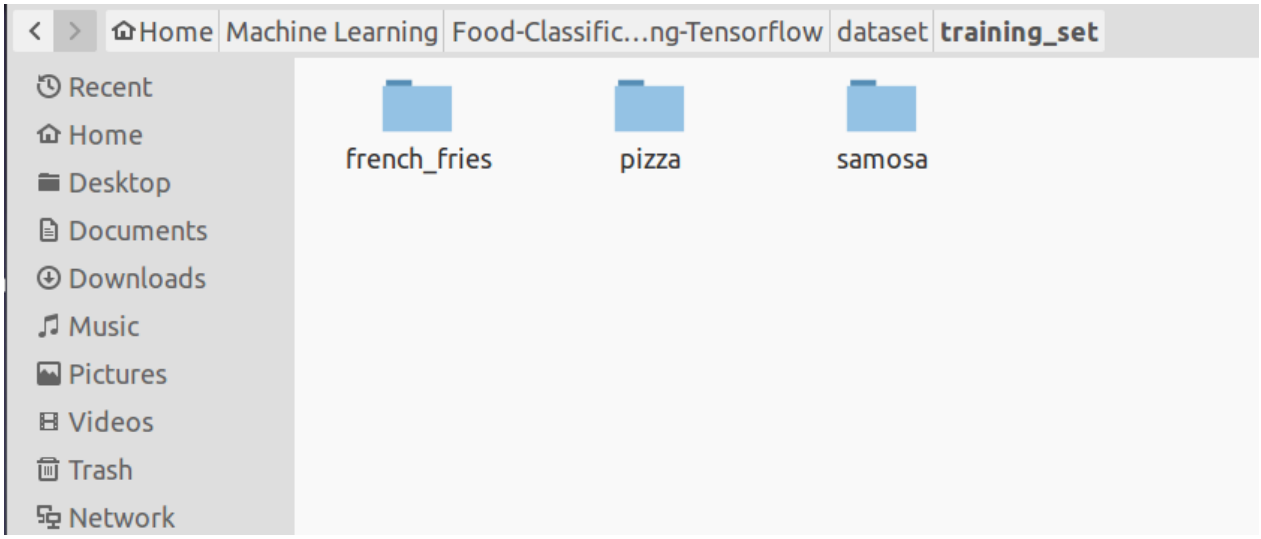
```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
```

```
/home/kemanth/anaconda3/envs/tensorflow/lib/python3.6/site-pac
kages/h5py/__init__.py:36: FutureWarning: Conversion of the se
cond argument of issubdtype from `float` to `np.floating` is d
eprecated. In future, it will be treated as `np.float64 == np.
dtype(float).type`.
  from ._conv import register_converters as _register_converte
rs
Using TensorFlow backend.
```

4.1. Dataset

The dataset is located in folder named dataset.

The data consists of three classes, french_fries, pizza and samosa.



Each folder contains approximately 1000 image files in each category. The name of the folder is actually the label of those files.

In [2]:

```
#Initialize the CNN
classifier = Sequential()
#Convolution and Max pooling
classifier.add(Conv2D(32, (3, 3), input_shape = (128, 128, 3), activation =
classifier.add(MaxPooling2D(pool_size = (2,2)))
classifier.add(Conv2D(64, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2,2)))
classifier.add(Conv2D(128, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

In [3]:

```
#Flatten
classifier.add(Flatten())
```

In [4]:

```
#Full connection
classifier.add(Dense(128, activation = 'relu'))
classifier.add(Dense(3, activation = 'softmax'))
```

In [5]:

```
#Compile classifier
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', m
```

In [6]:

```
#Fitting CNN to the images
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_ra
test_datagen = ImageDataGenerator(rescale=1./255)
training_set = train_datagen.flow_from_directory('./dataset/training_set',
test_set = test_datagen.flow_from_directory('./dataset/test_set', target_si
classifier.fit_generator(training_set, steps_per_epoch=800/32, epochs=50, v
```

```
ss: 0.4007 - acc: 0.8463 - val_loss: 0.4995 - val_acc: 0.830
4
Epoch 34/50
25/25 [==============================] - 14s 548ms/step - lo
ss: 0.3536 - acc: 0.8712 - val_loss: 0.4643 - val_acc: 0.817
0
Epoch 35/50
25/25 [==============================] - 12s 500ms/step - lo
ss: 0.3691 - acc: 0.8425 - val_loss: 0.6018 - val_acc: 0.781
2
Epoch 36/50
25/25 [==============================] - 12s 498ms/step - lo
ss: 0.3611 - acc: 0.8513 - val_loss: 0.5209 - val_acc: 0.825
9
Epoch 37/50
25/25 [==============================] - 12s 463ms/step - lo
ss: 0.3845 - acc: 0.8550 - val_loss: 0.5002 - val_acc: 0.781

2
Epoch 38/50
```

In [7]:

```
#save model
import os
target_dir = './models/'
if not os.path.exists(target_dir):
  os.mkdir(target_dir)
classifier.save('./models/model.h5')
classifier.save_weights('./models/weights.h5')
```

4.4 Creating a GUI and predicting a new image from url

Importing libraries for creating GUI

In [8]:

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_
from keras.models import Sequential, load_model
from PIL import Image, ImageTk
import requests
from io import BytesIO
from tkinter import Tk,Label,Canvas,NW,Entry,Button
from keras.preprocessing import image as image_utils
```

We now create a url and a button for browsing and loading image. The trained CNN is loaded and the saved parameters are restored. Prediction is made on the new image which is then output and the output is displayed on canvas.

In [9]:

```
#load model
img_width, img_height = 128, 128
model_path = './models/model.h5'
model_weights_path = './models/weights.h5'
model = load_model(model_path)
model.load_weights(model_weights_path)
```

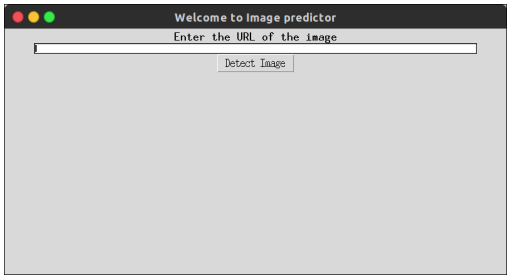
The main code which creates the gui with the help of clicked button

In [10]:

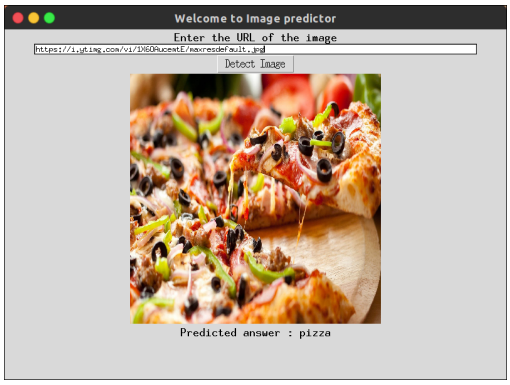
```
url = ''
window = Tk()
window.title("Welcome to Image predictor")
window.geometry('800x600')
lbl = Label(window, text="Enter the URL of the image", font=("Helvetica", 1
lbl.pack()
def clicked():
    global url
    lbl.configure()
    url  = (User_input.get())
    print(url)
    response = requests.get(url)
    test_image = Image.open(BytesIO(response.content))
    put_image = test_image.resize((400,400))
    test_image = test_image.resize((128,128))
    img = ImageTk.PhotoImage(put_image)
    pic = Label(image=img)
    pic.pack()
    pic.image = img
    test_image = image_utils.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)


    result = model.predict_on_batch(test_image)

    if result[0][0] == 1:
        ans = 'french fries'
    elif result[0][1] == 1:
        ans = 'pizza'
    elif result[0][2] == 1:
        ans = 'samosa'
    out = Label(window, text  = 'Predicted answer : ' +  ans, font=("Helvet
    out.pack()

User_input = Entry(width = 100)
User_input.pack()
btn = Button(window, text="Detect Image", font=("Helvetica", 12), command=c
btn.pack()
window.mainloop()
```

4.4 A sample run

The input window



Predicting Output



4.5 Conclusion

The performance of the system is high, and is considered acceptable from a usage point of view. However, the CNNs need high-performance computing machines in order to experiment on the huge multi-media datasets. The CNN is capable of train highly non-linear data, and for that in contrast, it takes more computational time to train the network. However, the performance matters a lot, and once the system is properly trained, the system can produce the results in less time. The images are properly preprocessed and all kinds of images are tested with CNN. From this, it is concluded that CNNs are more suitable for classifying the images when the number of classes are more. The task of image classification can be extended using prominent features that can categorize food images. Since the CNNs are consuming high computational time, the feature-based approach is highly appreciable. A multi-level classification approach (hierarchical approach) is suitable to avoid mis-classifications when the number of classes is more. Moreover, a dataset containing all food categories is also not available in the literature yet.

4.6 References

[1] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8228338 (http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8228338)
[2] https://keras.io/ (https://keras.io/)
[3] https://www.tensorflow.org/ (https://www.tensorflow.org/)