**COSC 189.02 – Lab 1 – Team 11**

Sanket S. Joshi ([f0042hj@dartmouth.edu](mailto:f0042hj@dartmouth.edu)),Anmol Chachra ([f0041wf@dartmouth.edu](mailto:f0041wf@dartmouth.edu))

**Abstract**

This project aims to visually classify 5 fruits, namely apple, guava, peach, mango and pomegranate. In order to be able to recognize the most general image to a good extent, the assortment of images includes 800+ images of fruits that vary in background, multiplicity, cuts, levels of ripening, exposure, etc. These are then split into training and test sets. The python code is a web application using Flask API on top of the Watson visual recognition API that extends the classifier functionality to a webpage. Users can upload custom images from local directory to the webpage and view matches along with their respective match scores as a list.

**Video :** <https://dartmouth.techsmithrelay.com/joao>

**Tools**

* **Server-side:** Python’s Flask API, Python OS library, IBM Watson visual recognition API
* **Client-side:** JavaScript, jQuery, AJAX, HTML5, CSS3, Bootstrap

**Process and techniques**

Figure : Process

1. The webpage loads with Python Flask API serving an HTML file over REST API
2. An HTML form takes in the file path of the local image that the user chooses
3. A validation layer confirms appropriate file extension
4. A synchronous AJAX call with image file payload is served to flask upon clicking on “Classify”
5. The server endpoint receives this file and stores it locally temporarily
6. The IBM Watson visual recognition API reads the image file and calls the classify function
7. The temporary image file is deleted
8. The response consisting of key value pairs of classes and scores is returned
9. The webpage displays this response in a tabular form

**Data Sources**

* The images are sourced from google image search
* Irrelevant images like graphical designs, sketches, etc. are filtered out
* Around 200 images are selected for each class with varying background, multiplicity, cuts, levels of ripening, exposure, etc
* Around 100 images are selected of fruits other than the mentioned ones for negative class
* Around 10 images from each class are taken out as test sets

**Special processing requirements**

* Python 3 needs to have flask installed
* All the other web dependencies like CSS and JS libraries have been included already as minified versions
* You just need to run the python file *webapp.py* and open the browser at *localhost:5000*
* If localhost:5000 does not work, check the python logs for the correct port

**The Cognitive Approach**

* Image classification is next to impossible without a cognitive approach, barring a few trivial cases. An algorithmic approach would mean relying on increasingly rule based mechanisms to detect geometric features, and collate them to interpret shapes, colours, etc.
* The cognitive approach does exactly this, but more implicitly. For example, if the image classification problem uses a neural network, then the individual layers are meant to capture basic to advanced features in abstracted layers, with accuracy improving with the increase in training size.

**Problems, Solutions and Retries**

* We started our exercise with bird and animal classifiers. The problems we mainly faced were the unavailability of images where the object was clearly separated from the background. This meant either supplying a large number of training images, or by hand picking examples with unambiguous background. The former would exceed the training capacity and the later would exceed time limitations. We then made a decision to classify fruits instead.
* We initially trained a fruit classifier for apple, banana and oranges. This seemed to be very trivial since the fruits vary by shape as well as colour. It was difficult to get low score matches and impossible to get negative matches. Hence, we decided to select fruits that are visually overlapping in their different stages. For instance, apples, peaches and pomegranates are very similar when ripe. The only things that differentiates them is their texture, as well as the stalk.
* We faced similar problems with completely ripe mangoes that are more red than yellow. individual images did not seem to be affected in terms of accuracy, but when the number of fruits were high, errors started to seep in. We notice that not enough of our training samples had fruits at different ripening stages and multiplicities. We decided to even out the proportions different factors in the training set to further improve accuracies of such outlier test examples.
* It seems like the model does not have the capability to differentiate background from the object. Hence, it was necessary to provide ample training examples without backgrounds as well as with varied backgrounds.
* Unlike training examples, the background did not usually matter for the test sets. For instance, when supplied with an apple juice carton as a test example, it still recognizes the apple. But what is slightly surprising is that when you supply a test image with multiple fruits across the classes, it does not match any class, whereas what is expected is that it should give a good match for all the identified classes.

**Summary of results**

The following table shows the number of images in the training set, test set, and their classification results. With a threshold of 0.6, the accuracy is 83.7%. The number of false positives and negatives seems to be very low despite of the similarity in test examples across class. This implies that the model has been trained well.

TODO

Table 1 Classification accuracy [TP : True positives, FP : False Positives, TN : True Negatives, FN : False Negatives]]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class | Training | Test | Right | Wrong | Accuracy |
| Apple | 190 | 9 | 8 | 1 | 88.89% |
| Guava | 133 | 7 | 5 | 2 | 71.43% |
| Peach | 174 | 11 | 9 | 2 | 81.82% |
| Mango | 153 | 5 | 5 | 0 | 100.00% |
| Pomegranate | 168 | 11 | 9 | 2 | 81.82% |
| Total | 818 | 43 | 36 | 7 | **83.72%** |
| Negative | 67 | 6 | 4 | 2 | 66.67% |

**Assessment of the resulting system’s effectiveness**

TODO

**Improvements**

TODO

**References**

TODO