DIP PROJECT 2K18

2 Fast 2 Fourier

PROJECT ID - 20

-Lovish Narang (201612158)
-Himansh Sheoran (20161211)
-Ayush Anand (20161085)

IDENTIFYING AND LABELLING OF INDIAN DISHES

MAIN GOALS:

Need to successfully segment out regions in an image consisting of Indian Foods using some sophisticated image segmentation techniques. Then train our model to so that it can recognise the label of any given image of an Indian meal spread.

PROBLEMS FACED:

The four main problem that will arise in this project is to :

1) Find an algorithm which can successfully segment the required part which we require, out of the image so that our training can be made more efficient.
2) Implementing efficient feature descriptors which can extract out features from the segmented out part.
3) Training the model on the basis of the features extracted using different Multi-Class classifiers and finding out respective accuracies.
4) Prevent overfitting.

GLIMPSE OF THE DATASET USED:

The model trained is capable of determining out of the following Indian dishes:

- Aloo Gobhi
- Biryani
- Butter Paneer
- Chapati
- Dhokla
- Dosa
- Gulab Jamun
- Ladoo
- Lassi
- Palak Paneer

Courtesy of: Google Images

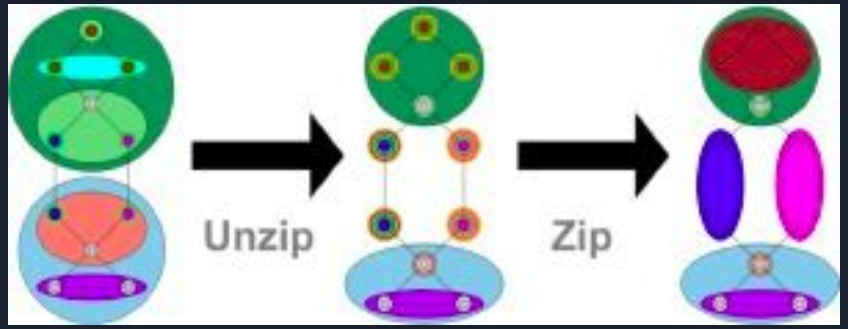
MAIN STEPS FOLLOWED:

- Create Training And Testing Dataset of all 10 classes with 300 Images in Training Set And 100 Images in Testing Images.
- Performing Segmentation using a graph based technique.
- Extracting the following features out of the segmented part:
    - Shape Descriptor -> HU Moments
    - Texture Descriptor -> Haralick
    - Color Descriptor -> Histograms
    - Gradient Descriptor -> HOG
- Scaling and appending all features of an image together.
- Training the model using different Multi-Class Classifiers and then testing it on new samples to report respective accuracies
- Tuning the model to predict better.

Brief explanation of the steps followed

MINIMUM SPANNING TREE SEGMENTATION:

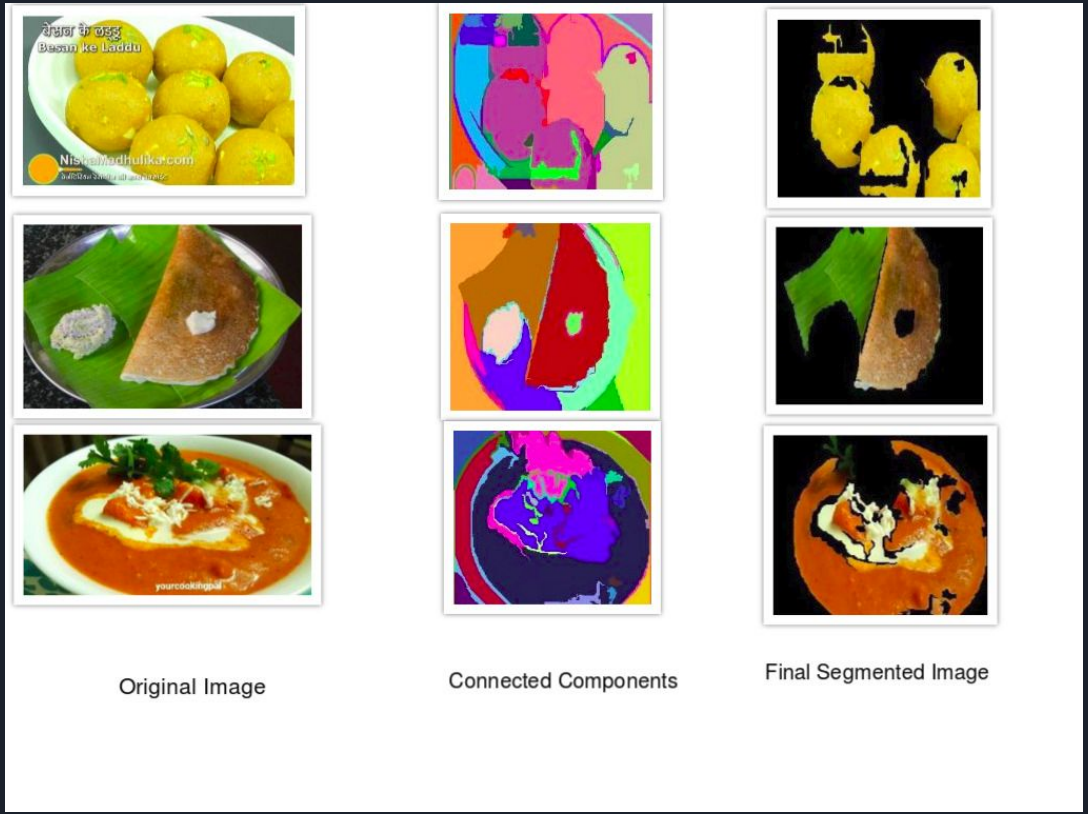
- The purpose of segmentation is to extract out regions out of image with maximum useful information and discard unwanted regions.
- The technique used in the model is graph based segmentation.
- Graph partitioning methods are an effective tools for image segmentation since they model the impact of pixel neighborhoods on a given cluster of pixels or pixel, under the assumption of homogeneity in images.
- In these methods, the image is modeled as a weighted, undirected graph. Usually a pixel or a group of pixels are associated with nodes and edge weights define the (dis)similarity between the neighborhood pixels. The graph (image) is then partitioned according to a criterion designed to model "good" clusters. Each partition of the nodes (pixels) output from these algorithms are considered an object segment in the image.



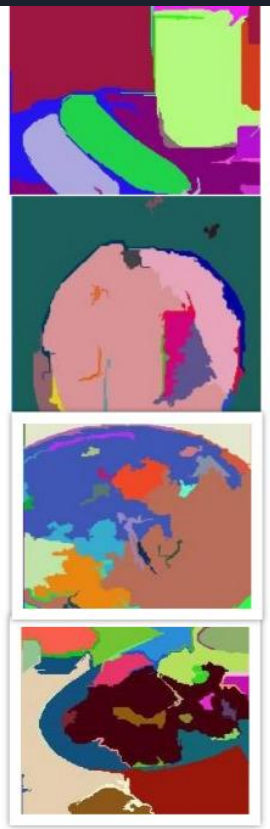
The basic logic of Graph based segmentation is to first convert the image into a collection of forests and then keep on merging the similar trees together. The merging and segmenting procedures are both parameterized. Thus it is imperative to find the correct hyperparameters so that valuable information can be extracted out of each and every class of objects.

RESULT OF GRAPH BASED SEGMENTATION IS A COLLECTION OF CONNECTED COMPONENTS, OUT OF WHICH THE LARGEST COMPONENT IF PICKED UP.

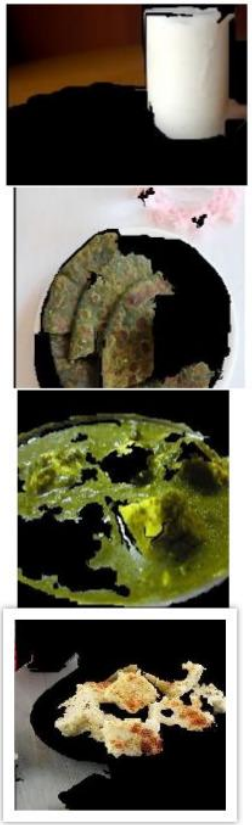
EXAMPLES OF SEGMENTATION:



Original Image



Connected Components



Segmented Image

FEATURE EXTRACTION:

- **HU MOMENTS (SHAPE DESCRIPTOR) :**

  In image processing, computer vision and related fields, an **image moment** is a certain particular weighted average (moment) of the image pixels' intensities, or a function of such moments, usually chosen to have some attractive property or interpretation.Image moments are useful to describe objects after segmentation. Simple properties of the image which are found *via* image moments include area (or total intensity), its centroid, and information about its orientation.

  Hu Moments are normally extracted from the silhouette or outline of an object in an image. By describing the silhouette or outline of an object, we are able to extract a shape feature vector (i.e. a list of numbers) to represent the shape of the object.

  We can then compare two feature vectors using a similarity metric or distance function to determine how “similar” the shapes are.

FEATURE EXTRACTION:

- **HARALICK (TEXTURE DESCRIPTOR):**

  Haralick's texture features were calculated to describe the texture of the image. The basis for these features is the gray-level co-occurrence matrix.

  This matrix is square with dimension N_g, where N_g is the number of gray levels in the image. Element [*i*,*j*] of the matrix is generated by counting the number of times a pixel with value *i* is adjacent to a pixel with value *j* and then dividing the entire matrix by the total number of such comparisons made.

  Each entry is therefore considered to be the probability that a pixel with value *i* will be found adjacent to a pixel of value *j*.

FEATURE EXTRACTION:

- **HISTOGRAMS (COLOR DESCRIPTOR):**

  Color can be considered a very crucial descriptor for our model. Each object has a distinct color which is specific to all the instances belonging to one class.

  Therefore it is very important to store the colors of the segmented out part of the image as it can help the classifier to recognize an image by analyzing the color of the largest component of the image.

FEATURE EXTRACTION:

- **HOG (GRADIENT DESCRIPTOR):**

  The **histogram of oriented gradients (HOG)** is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

  The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms.

CLASSIFIERS USED:

The model trains using 5 different classifiers:

- **Random Forest Classifier**
- **Linear SVM**
- **RBF Kernel SVM**
- **POLY Kernel SVm**
- **Logistic Regression**
- **Ensemble Methods(Logistic Regression,Decision Tree,Linear SVM)**

RANDOM FOREST CLASSIFIER:

In [9]:
```
import matplotlib.pyplot as plt
clf  = RandomForestClassifier(n_estimators=206, random_state=16)
count = 0
clf.fit(trainDataGlobal, trainLabelsGlobal)
y_pred = clf.predict(testDataGlobal)
for i in range(len(testLabelsGlobal)):
    if testLabelsGlobal[i] == y_pred[i]:
        count+=1
print(100*count/len(y_pred))
```

```
60.13513513513514
```

RBF KERNEL SVM

In [15]:
```
svclassifier = SVC(kernel='rbf')
svclassifier.fit(trainDataGlobal, trainLabelsGlobal)
count = 0
y_pred = svclassifier.predict(testDataGlobal)
for i in range(len(testLabelsGlobal)):
    if testLabelsGlobal[i] == y_pred[i]:
        count+=1
print(100*count/len(y_pred))
```

```
/home/yudhik/.local/lib/python3.5/site-packages/sklearn/svm/base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

```
38.54054054054054
```

LOGISTIC REGRESSION:

In [11]:
```
from sklearn.linear_model import LogisticRegression
count = 0
logisticRegr = LogisticRegression()
logisticRegr.fit(trainDataGlobal, trainLabelsGlobal)
y_pred = logisticRegr.predict(testDataGlobal)
for i in range(len(testLabelsGlobal)):
    if testLabelsGlobal[i] == y_pred[i]:
        count+=1
print(100*count/len(y_pred))
```

```
/home/yudhik/.local/lib/python3.5/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/home/yudhik/.local/lib/python3.5/site-packages/sklearn/linear_model/logistic.py:459: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

```
35.810810810810814
```

LINEAR SVM:

In [12]:
```
from sklearn.svm import SVC
count = 0
svclassifier = SVC(kernel='linear')
svclassifier.fit(trainDataGlobal, trainLabelsGlobal)
```

Out[12]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [13]:
```
y_pred = svclassifier.predict(testDataGlobal)
for i in range(len(testLabelsGlobal)):
    if testLabelsGlobal[i] == y_pred[i]:
        count+=1
print(100*count/len(y_pred))
```

```
33.108108108108105
```

POLY KERNEL SVM:

In [14]:
```
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(trainDataGlobal, trainLabelsGlobal)
count = 0
y_pred = svclassifier.predict(testDataGlobal)
for i in range(len(testLabelsGlobal)):
    if testLabelsGlobal[i] == y_pred[i]:
        count+=1
print(100*count/len(y_pred))
```

```
12.162162162162161
```

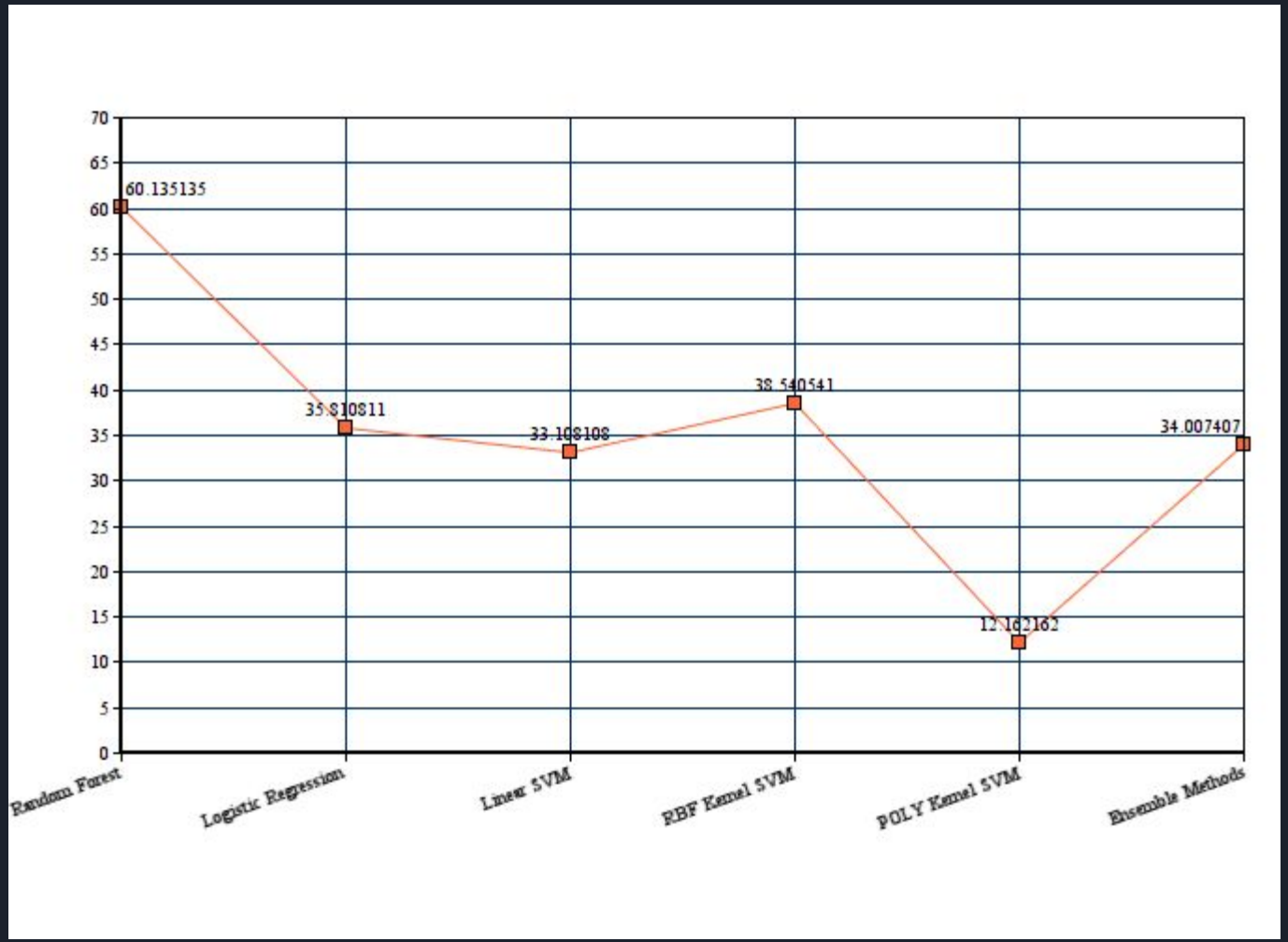
ENSEMBLE METHODS:

In [18]:
```
kfold = model_selection.KFold(n_splits=10, random_state=seed)
estimators = []
model1 = LogisticRegression()
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
# create the ensemble model
ensemble = VotingClassifier(estimators)
results = model_selection.cross_val_score(ensemble, trainDataGlobal,trainLabelsGlobal, cv=kfold)
```

...

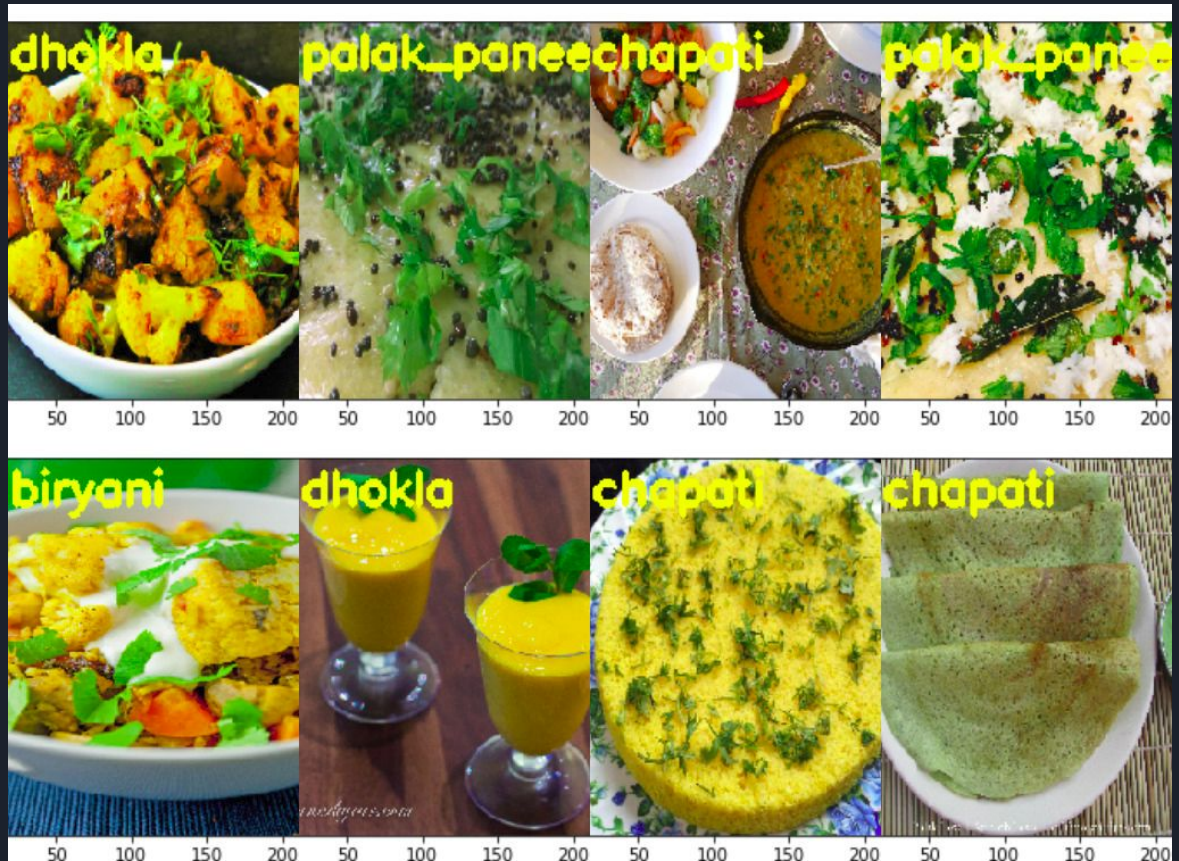
In [19]:
```
print(results.mean())
```

```
0.3400740657614184
```


70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
60.135135
35.810811
33.108108
38.540541
12.162162
34.007407
Random Forest
Logistic Regression
Linear SVM
RBF Kernel SVM
POLY Kernel SVM
Ensemble Methods

EXAMPLES OF CORRECTLY CLASSIFIED:

EXAMPLES OF WRONGLY CLASSIFIED



FUTURE GOALS

1) Creating a more robust model capable of deciding among larger dataset with greater number of classes
2) Implementing neural network and test the model on it to compare with the classical machine learning algorithms
3) Implementing an even better type of feature descriptor more specific to a particular class so as to increase the accuracy of that particular class.
4) Implementing a bounding box object detector which correctly segments out the required pixels out of an image.

THANK YOU!