Introduction to Image Processing and Computer Vision

Laboratory Project 2: Image Classification

Jakub Bodak[1]

[1]Faculty of Mathematics and and Information Science, Warsaw University of Technology

January 22, 2023

1 Introduction

Image classification is a fundamental task in computer vision, with applications in a wide range of industries such as autonomous vehicles, medical imaging, and security systems. This report aims to provide example of how to classify simple images.

In this lab, we are using a particular directory from $KTH - texture - Data$ dataset with pictures of *Aluminium foil*, *Brown bread*, *Corduroy*, *Cork*, *Cotton*, *Cracker*, *Linen*, *Orange peel*, *Sponge*, $Styrofoam$ and *Wool*. Our goal is to train a chosen classifier on features extracted from images from folder *train* and test the alogrithm for classification on images obtained from *valid* folder.

Figure 1: Random image from each folder in a sequence according to the description

2 Solution

The main algorithm can be separated into 2 stages: Training and Testing. Firstly I declared variables for directory paths as well as storing and calculating features, classifiers:

```
# directories
dir_train = 'Splited/train/'
dir_valid = 'Splited/valid/'

# variables for features
labels = []
global_features_hu = []
global_features_hara = []
global_features_hist = []
global_features_tog = []
method = 'uniform'
radius = 3
n_points = 3

# variables for the classifier
num_trees = 100
seed = 9
```

2.1 Features and labels preparation

The first task is to prepare data structures to store features and labels of images from *train* folder:

```
# labels for classification
train_labels = os.listdir(dir_train)
train_labels.sort()
test_labels = os.listdir(dir_valid)
test_labels.sort()
```

Now we have to iterate through all files in the *train* directory:

```
# iterating through folders in train directory
for folder_name in train_labels:

    dir_train_folder = os.path.join(dir_train, folder_name)

    # iterating through files in a folder
    for file_name in os.listdir(dir_train_folder):
```

The features chosen in the algorithm are HuMoments, Haralick Textures and Color Histogram features. They are combined into one set of features for the last features named *Together*. These should be evaluated for every file:

```
img_start = cv2.imread(dir_train_folder + "/" + file_name)
img_gray = cv2.cvtColor(img_start, cv2.COLOR_BGR2GRAY)

hu = cv2.HuMoments(cv2.moments(img_gray)).flatten()
hara = mahotas.features.haralick(img_gray).mean(axis=0)
hist = cv2.calcHist([img_start], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256]).flatten()
cv2.normalize(hist, hist)
tog = np.hstack([hu, hara, hist])

labels.append(folder_name)
global_features_hu.append(hu)
global_features_hara.append(hara)
global_features_hist.append(hist)
global_features_tog.append(tog)
```

2.2 Classifier training

When it comes to choosing a classifier, *Random Forest* was chosen due to its best benchmark results. We have to declare a classifier for each of the features and train it with the *train* folder data of features and labels:

```
clf_hu = RandomForestClassifier(n_estimators=num_trees, random_state=seed)
clf_hu.fit(global_features_hu, labels)
clf_hara = RandomForestClassifier(n_estimators=num_trees, random_state=seed)
clf_hara.fit(global_features_hara, labels)
clf_hist = RandomForestClassifier(n_estimators=num_trees, random_state=seed)
clf_hist.fit(global_features_hist, labels)
clf_tog = RandomForestClassifier(n_estimators=num_trees, random_state=seed)
clf_tog.fit(global_features_tog, labels)
```

2.3 Testing

With the classifiers trained we can move on to testing whether they can recognise to what category the picture would suit. Firstly, we need to declare variables for accuracy allocation:

```
# variables for result correctness checking
good_hu = [0] * 11
good_hara = [0] * 11
good_hist = [0] * 11
good_tog = [0] * 11
count = [0] * 11
j = 0
```

Now iteration through files in *valid*/*test* folder is needed:

```
# iterating through folders in train directory
for folder_name in test_labels:

    dir_test_folder = os.path.join(dir_valid, folder_name)

    # iterating through files in a folder
    for file_name in os.listdir(dir_test_folder):
```

For each image, same features as for images from *train* folder need to be evaluated:

```
img_start = cv2.imread(dir_test_folder + "/" + file_name)
img_gray = cv2.cvtColor(img_start, cv2.COLOR_BGR2GRAY)

hu = cv2.HuMoments(cv2.moments(img_gray)).flatten()
hara = mahotas.features.haralick(img_gray).mean(axis=0)
hist = cv2.calcHist([img_start], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256]).flatten()
cv2.normalize(hist, hist)
tog = np.hstack([hu, hara, hist])
```

Now we let the classifier predict which label should be chosen for each image based on training results:

```
# prediction of what is on the image by the classifier
prediction_hu = clf_hu.predict(hu.reshape(1, -1))[0]
prediction_hara = clf_hara.predict(hara.reshape(1, -1))[0]
prediction_hist = clf_hist.predict(hist.reshape(1, -1))[0]
prediction_tog = clf_tog.predict(tog.reshape(1, -1))[0]
```

3 Results

The algorithm also compares the resulting label with folder name to determine whether the classifier predicted correctly and adds values to counters:

```
count[j] += 1
if prediction_hu == folder_name:
    good_hu[j] += 1
if prediction_hara == folder_name:
    good_hara[j] += 1
if prediction_hist == folder_name:
    good_hist[j] += 1
if prediction_tog == folder_name:
    good_tog[j] += 1
```

Accuracies for each folder where computed by:

$$A = \frac{GOOD}{COUNT}$$

3.1 Mean results

The results were saved for each feature to see what is the best one to classify images from this particular dataset. The accuracy obtained in this project across all images was 33,7499% for HuMoments features, 79,6874% for Texture Haralick features, 96,3750% for Color Histogram features and 96,0624% for these grouped (Together) features. As we can see the images were best classified by *RandomTreeForest*, when a Color Histogram features have been used.

3.2 Best and worst classifications

My algorithm checks classification accuracies for all combinations, that is each label has 4 accuracies (computed with results from predictions based on different features). Five best results in these field were:

- styrofoam with Color Histogram - 100%
- styrofoam with Together - 100%
- orange peel with Color Histogram - 100%
- orange peel with Together - 100%
- aluminium foil with Together - 100%

These were also only results with 100% accuracy. This suggests that *Together* features work the best for this given dataset.

Five worst results were:

- brown bread with HuMoments - 0%
- cracker with HuMoments - 8,8235%
- orange peel with HuMoments - 8%
- sponge with HuMoments - 2,439%
- styrofoam with HuMoments - 2,439%

3.3 Result analysis

From previous two subsections we can clearly conclude which features were better for this dataset classication. *HuMoments*, which are features of shape does not work with these images. These results reign superior in the worst cases competition. On the other hand *Color Histogram* and *Together* features are very efficient, being very close in mean values for whole dataset and takin the best cases places for themselves.