EE58J - Homework 1
Product Image Recognition Challenge 2019

Efehan Atıcı - 2016700186

22 March 2019

1 Size Normalization

In this step I've resized every image in the dataset to 128x128 pixels. This function is called 'resizeBatch' on util.py in my code. We run this resizing algorithm for all folders in the dataset, 'confectionery', 'icecream', 'laundry', 'softdrinks-1' and 'softdrinks-2'.

After we resize all the images to 128x128, we can continue with other steps.

2 Image Description

2.1 Color Histogram

In this step I've created color histograms for every image in the dataset. This function is called 'colorHist' on util.py in my code. It checks for all files ending with '.jpg' extension, loads images and converts them to HSV color space. According to the given window size (1x1 or 2x2 windows for each image), it creates an array that consists of Hue, Saturation and Value histograms. And then we normalize the HSV histograms and append them to a final array which describes the image. Finally, we save this image description as 'crop###_color.npy'. We then use this .npy to save time for calculating color histograms on the next steps.

For creating the histograms, I used bin size as 10 bins for every histogram. Using more than 10 bins makes a big performance hit (finding all descriptors and making classifications for 15 bins with 4x4 windows took more than 30 minutes). Using less than 10 bins would mean there is less information about the image. Choosing 10 bins for histograms seemed the ideal amount, so I used 10 bins when creating every histogram.

Code snippet for this step is as follows:

```
# Only look for files that end with .jpg
if name.endswith(".jpg"):
```

```
# Load image file
filePath = os.path.join(root, name)
im = cv2.imread(filePath, 1)

# Convert to HSV color space
HSV_im = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)

# Grid stride value for chosen window size
s = int(128 / windowNr)

# Final array for holding all the histograms
full_hsv_hist = []
for i in range(windowNr):
    for j in range(windowNr):

        # Take a window from image
        #  given window size
        window=im[i*s:(i+1)*s,j*s:(j+1)*s]

        # Create Hue, Sat, Value Histograms
        hHist=np.histogram(window[:,:,0])
        sHist=np.histogram(window[:,:,1])
        vHist=np.histogram(window[:,:,2])

        # Normalize HSV histograms
        hNorm=np.divide(hHist[0],np.sum(hHist[0]))
        sNorm=np.divide(sHist[0],np.sum(sHist[0]))
        vNorm=np.divide(vHist[0],np.sum(vHist[0]))

        # Concatenate histograms into 1 big array
        hsvHist=np.concatenate((hNorm, sNorm, vNorm))

        # Append them to a final array
        full_hsv_hist.append(hsvHist)
saveName = name.replace(".jpg", "_color.npy")
np.save(os.path.join(root,saveName),full_hsv_hist)
```

After we create color histogram for every image, we move onto the next step.

2.2 Gradient Orientation Histogram

In this step I've created gradient orientation histograms for every image as I did for color histograms. This function is called 'HOGHist' on util.py in my code. It checks for all files ending with '.jpg' extension, loads images as grayscale images. According to the given window size, it creates the image gradients for X and Y coordinates. We do this by using Sobel operator inside the opencv (cv2.Sobel). Then, after creating gradients for X and Y, we find the gradient orientation simply by taking the arctan of Y and X gradients (np.arctan2). Finally, we save this image description as 'crop###_orient.npy'. Code snippet for this step is as follows:

```
if name.endswith(".jpg"):

    # Load image as grayscale
    filePath = os.path.join(root, name)
    im = cv2.imread(filePath, 0)

    # Find X and Y Gradients
    sobelx = cv2.Sobel(im, cv2.CV_64F, 1, 0, ksize=5)
    sobely = cv2.Sobel(im, cv2.CV_64F, 0, 1, ksize=5)

    # Find orientations in angles, from X and Y grad
    orient = np.arctan2(sobely, sobelx) * 180 / np.pi

    for i in range(windowNr):
        for j in range(windowNr):
            ...

            # Find the histogram for the current window
            orient_hist=np.histogram(window,bins=bin_num)

            # L1 Normalize the histogram
            ...
    # Save the descriptor for future
    saveName = name.replace(".jpg", "_orient.npy")
    np.save(os.path.join(root, saveName), full_orient)
```

After we create gradient orientation histogram for every image, we move onto the next step.

3 Nearest Neighbor (NN) Classification

In this step, after creating color and gradient orientation histograms for every image, I've separated images into training and test sets by 80% and 20% respectively.

In total there are 10362 images in the dataset. Among these images, I've randomly selected 2037 images as test set, and the rest 8325 images as training set. I've selected roughly 20% of images from all classes as test set.

3.1 NN Classification using Color Histograms

This function is called ‘nnColor’ in util.py in my code. It takes the all color histograms inside the training dataset, and then appends them to create big array consisting of all the training color histograms. After that, we check for every image in the test set and subtract the color histogram from every element in the training array. We then take the absolute value of this subtracted array of histograms, and find the element with the minimum mean value which tells us the closest neighbour to our guessed image.

Using this technique, we get the following accuracy:

Window Size	Accuracy	Correct	Total
1x1	49.39%	1006	2037
2x2	55.72%	1135	2037
4x4	57.63%	1174	2037

As the window size increases, we see an increase in the accuracy. This happens because there is more information contained in the descriptor. There will be 16 different histograms for 4x4 window size, but there will be only 1 histogram that explains the image in the 1x1 window.

Here is the table of accuracy for every product category when window size is 1x1:

Category	Accuracy	Correct	Total
Confectionery	39.56%	161	407
Icecream	45.24%	190	420
Laundry	60.69%	247	407
Softdrinks-I	34.33%	137	399
Softdrinks-II	67.08%	271	404

On 1x1 window size, using the color histograms descriptors, the most confused category is Softdrinks-I (34.33% accuracy). However the most accurately predicted product category is Softdrinks-II (67.08%), with over 50% accuracy.

The class that had the lowest accuracy was named ‘8693’. It is the class of Coca-Cola can. This class had an accuracy of 1 correct guesses out of 20 total images in the test set. The class is shown in Figure 1. Here is the table of accuracy for every product category when window size is 2x2:

Category	Accuracy	Correct	Total
Confectionery	54.54%	222	407
Icecream	40.95%	172	420
Laundry	67.32%	274	407
Softdrinks-I	40.35%	161	399
Softdrinks-II	75.74%	306	404

Figure 1: Most confused class for color histogram method when used with 1x1 windows: ‘8693’. Coca-cola can

On 2x2 window size, using the color histograms descriptors, the most confused category is Softdrinks-I (29.82%). The most accurately predicted product category is Softdrinks-II (60.89%).

Here is the table of accuracy for every product category when window size is 4x4:

Category	Accuracy	Correct	Total
Confectionery	54.30%	221	407
Icecream	47.62%	200	420
Laundry	65.35%	266	407
Softdrinks-I	44.11%	176	399
Softdrinks-II	76.98%	311	404

On 4x4 window size, we see accuracy increase in every category. Still, the worst guessed category is Softdrinks-I (31.58%) and the best guessed category is Softdrinks-II (67.33%). The most confused class in this case is named ‘8706’, which is a big Coca-Cola bottle which can be seen in Figure 2.

Figure 2: Class 8706: Big Coca-Cola bottle

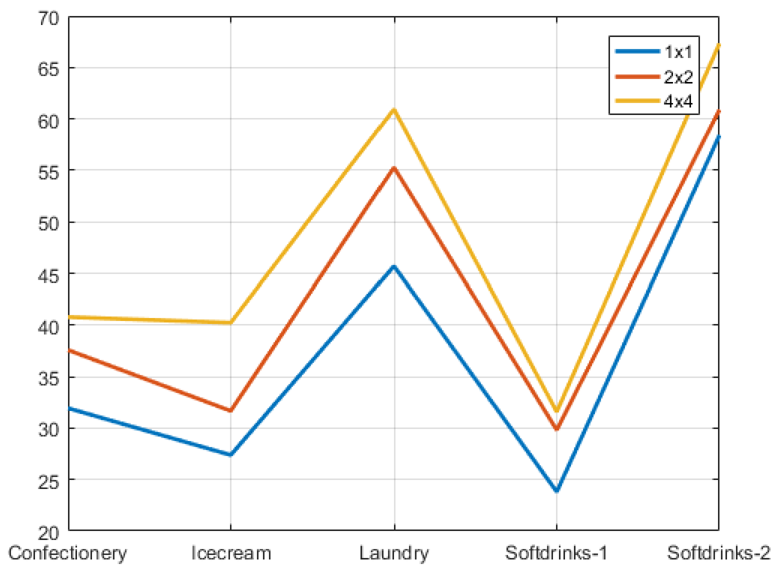

Figure 3: Plot for accuracy of every window size using color histograms

3.2 NN Classification - Gradient Orientation Histograms

This function is called ‘nnOrient’ in util.py in my code. It works the same way as calculating the classification for Color Histograms.

Using this technique, we get the following accuracy:

Window Size	Accuracy	Correct	Total
1x1	9.52%	194	2037
2x2	25.08%	511	2037
4x4	39.02%	795	2037

As we can see, we get lower accuracy than using the color histograms method. However, as the window size increases, the accuracy gets closer to the color histograms method.

Here is the table of accuracy for every product category when window size is 1x1:

Window Size	Accuracy	Correct	Total
Confectionery	8.10%	33	407
Icecream	10.00%	42	420
Laundry	13.51%	55	407
Softdrinks-I	4.76%	19	399
Softdrinks-II	11.14%	45	404

On 1x1 window size, the most confused category Softdrinks-I (4.76%). Whereas the most accurately predicted product category is Laundry (13.51%). Since we get such a low accuracy with this method, I couldn’t identify the lowest accuracy class. More than many classes had zero percent accuracy.

Here is the table of accuracy for every product category when window size is 2x2:

Category	Accuracy	Correct	Total
Confectionery	20.88%	85	407
Icecream	21.90%	92	420
Laundry	33.66%	137	407
Softdrinks-I	16.54%	66	399
Softdrinks-II	32.42%	131	404

On 2x2 window size, bad accuracy shifts to the Icecream product category. However, Icecream, Confectionery and Softdrinks-I classes have almost equal accuracy. All these categories perform worse than Laundry and Softdrinks-II. Worst class accuracy was 0% on many classes.

Here is the table of accuracy for every product category when window size is 4x4:

Category	Accuracy	Correct	Total
Confectionery	30.96%	126	407
Icecream	33.33%	140	420
Laundry	48.16%	196	407
Softdrinks-I	28.82%	115	399
Softdrinks-II	53.96%	218	404

On 4x4 window size, we see accuracy increase in every category. Still, the worst guessed category is Softdrinks-I (28.82%) and the best guessed category is Softdrinks-II (53.96%). Most confused class here was named '6800' which can be seen in Figure **??**. It had 1 correct prediction out of 21 total images.

Figure 4: Class 6800: Ülker Hanımeller

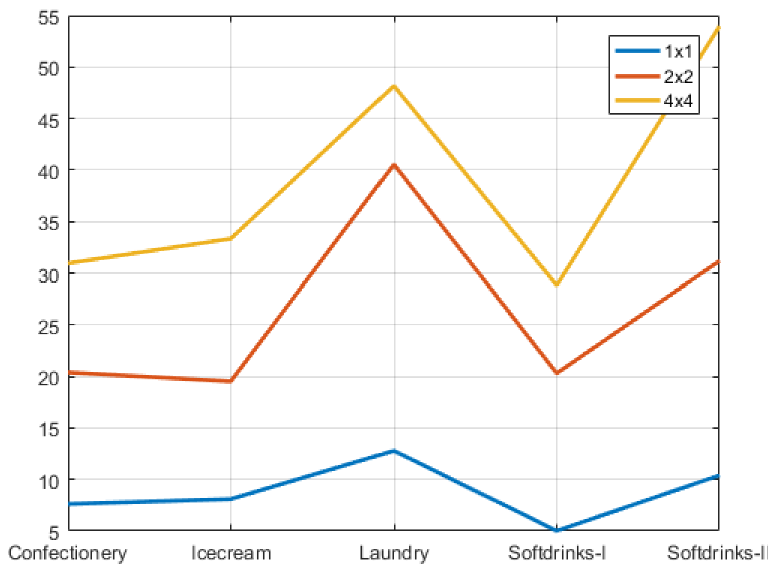

Figure 5: Plot for accuracy of every window size using gradient orientations

3.3 NN Classification - Combination

This function is called 'nnCombine' in util.py in my code. In this function, we calculate the difference between color histograms and orientation histograms separately, and then we sum them up to find the closest neighbour. Using this way, we get a higher accuracy than both of the other techniques:

Window Size	Accuracy	Correct	Total
1x1	52.77%	1075	2037
2x2	58.62%	1194	2037
4x4	62.15%	1266	2037

Since we combined both techniques, we get a higher accuracy for detection overall. Our best classification accuracy is 56.90% with 1159 correct guesses out of 2037 total images when we have 4x4 windows on the image.

Here is the table of accuracy for every product category when window size is 1x1:

Category	Accuracy	Correct	Total
Confectionery	48.4%	197	407
Icecream	45.7%	192	420
Laundry	62.16%	253	407
Softdrinks-I	35.84%	143	399
Softdrinks-II	71.78%	290	404

On 1x1 window size, using the combination of descriptors, the most confused category is Softdrinks-I with 35.84% accuracy. Whereas the most accurately predicted product category is Softdrinks-II with 71.78% accuracy.

The class that had the worst accuracy in this category is '8736', Figure 6.

Figure 6: Class 8736: Fanta can

Figure 7: Badly cropped image

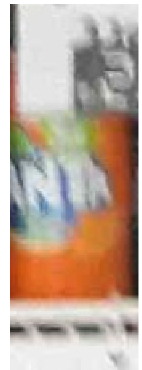

Figure 8: Low quality image

This class has the lowest accuracy of 0 correct predictions out of 20 total images in the test set. This is due to this class having photos that are cropped inaccurately, or low quality photos such as in Figures 7 and 8.

Here is the table of accuracy for every product category when window size is 2x2:

Category	Accuracy	Correct	Total
Confectionery	58.48%	238	407
Icecream	44.05%	185	420
Laundry	67.08%	273	407
Softdrinks-I	42.86%	171	399
Softdrinks-II	80.94%	327	404

On 2x2 window size, both Icecream and Softdrinks-I categories share bad accuracy. However, Softdrinks-II category is the best performing category, with over 70% accuracy! The worst accurate class in this case was the '8693': Coca-Cola can, from the previous color histogram method. It has 2 correct predictions out of 20.

Here is the table of accuracy for every product category when window size is 4x4:

Category	Accuracy	Correct	Total
Confectionery	62.65%	255	407
Icecream	52.62%	221	420
Laundry	66.09%	269	407
Softdrinks-I	47.12%	188	399
Softdrinks-II	82.42%	333	404

On 4x4 window size, we see a little accuracy drop in confectionery, and we don't see any big improvements on other categories. Still, the worst guessed category is Softdrinks-I and the best guessed category is Softdrinks-II. Worst class this time was ‘8706’: big Coca-Cola bottle, with 1 out of 20 correct guesses, Figure 2.

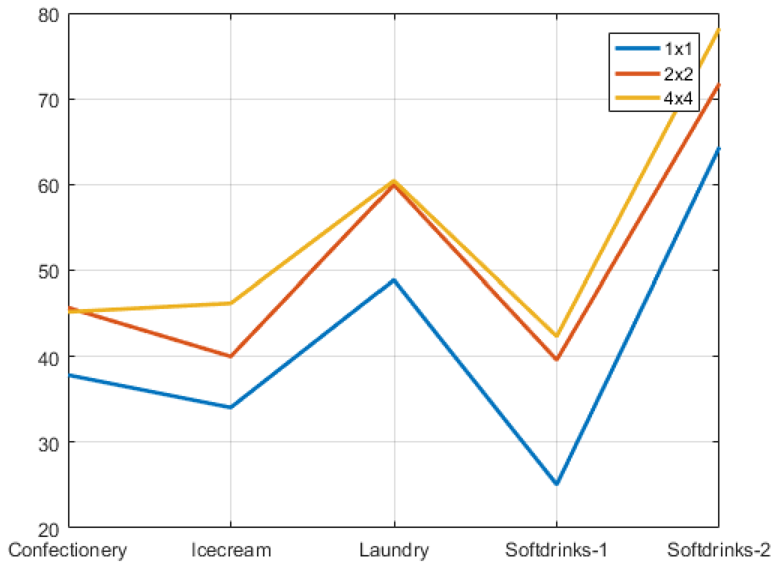

Figure 9: Plot for accuracy of every window size using both descriptors