Assignment #1: (Uniform) LBP recognition using Python

Robert Jutreša
IBB 22/23, FRI, UL
rj7149@student.uni-lj.si

I. Introduction

The Local Binary Pattern (LBP) descriptor is a classification descriptor used in computer vision. It calculates a binary representation for each pixel in regards to it's surrounding texture. For this assignment, it will be used for recognition for images of ears. This will be done by comparing various parameter combinations and implementations to see improvements/deteriorations to rank-1 classification accuracy compared to methods, implementations or versions of this descriptor.
Firstly the background and logic of the LBP descriptor will be described, followed by it's implementation for the purposes of this assignment. Lastly the results of the experimentation will be presented and discussed.

II. Methodology

The aim of this assignment was to do implement a basic recognition system using the Local Binary Pattern (LBP) descriptor.
In order to have comparative results, firstly a pixel by pixel descriptor was developed. This descriptor produced a feature vector, that is the result of vectorizing the original image

$$vec(A) = [a_{1,1}, ..., a_{m,1}, a_{1,2}, ..., a_{m,2}, ..., a_{1,n}, ..., a_{m,n}]^T \tag{1}$$

where the image is a square matrix $A \in \mathbb{R}^{n,n}$ and thus the resulting feature vector is defined as $\vec{v} \in \mathbb{R}^{nn}$.
The end product of the LBP descriptor was also a vectorized image 1, where each pixel is transformed in accordance to its surrounding texture. Each pixel takes on a binary value on the interval $[0, 2^P]$, where P is defined as the word length of the descriptor. This is done by computing

$$\sum_{p=0}^{P-1} s(g_p - g_c) * 2^p \tag{2}$$

for each pixel g_c, where the function s is defined as

$$s(x) = \begin{cases} 1 & x >= 0 \\ 0 & x < 0 \end{cases} \tag{3}$$

and the points g_p are obtained by computing their coordinates $(R * cos(\frac{2\pi p}{P}), R * sin(\frac{2\pi p}{P})$ where R is the radius around the center point g_c.

To expand on this two additional improvements on the descriptor were implemented. Namely the Uniform Local Binary Pattern (ULBP), which introduces rotational invariance and easier implementation using a look-up table, and representing the texture image using a histogram. ULBP is implemented as the number of spacial transitions (bitwise changes) in the LBP pattern. For every pixel in the image, a local pattern is calculated using

$$s(x) = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{if } U(LBP) <= 2 \\ P + 1 & \text{otherwise} \end{cases} \tag{4}$$

The function U calculates the bitwise changes

$$U(LBP) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{p=0}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)| \tag{5}$$

[1], [2]. Histogram was implemented such that it kept the local information of the image [3]. This was done by calculating

$$H(k) = \sum_{n=0}^{N/h} \sum_{m=0}^{M/h} f(LBP(i, j), k), k \in [0, K] \tag{6}$$

where h is the number of evenly sized areas into which the image is divided and K is the maximum number in the pattern.

III. Experiments

Basic recognition for this assignment was done on the Annotated Web Ears (AWE) dataset. Which is a dataset of tightly cropped images of ears of varied sizes. They span classes (subjects) of different ethnicities and genders, with varied accessories and in different head positions. Images of a single class contain both left and right ear images [4].
The program, and the corresponding functions, were written using Python in a Anaconda environment. The images were read and stored using the OpenCV (`cv2`) and `os` libraries. The images were read in greyscale and resized to a common image size. Feature vectors were obtained by using the `numpy` implementation of equation 1, for which a distance matrix was computed for all pairs or feature vectors using the `scipy` library. From there, the classification was computed based on the index of the minimum distance value of each image, and compared with the original classification of that image. Based on this rank-1 classification accuracy was calculated.

LBP and ULBP were done by directly implementing equations 2, 4, 5 (as well as 3) with the `generic_filter` function from the `scipy` library. To build on top of those implementations, local histograms were computed and concatenated using `numpy`. Rank-1 accuracy scores were stored using `pandas` and some baseline results were obtained using the function `skimage.feature.local_binary_pattern`.
The above roughly described implementations were ran using various different parameter values seen in Table I. The default parameter values were 1) Image Size: 128px x 128px 2) $R = 1$ 3) $P = 8$ 4) Step: 1px 5) Histogram Area: 16px. The histogram area parameter was used to explicitly state the size of the are for local histograms, rather than computing the it as stated in equation 6. The parameter step stated the level of local region overlap, where higher values meant bigger distances between subsequent pixels (or number of skipped pixels).

TABLE I: Used parameter values.

Image Size [px]	R	P	Histogram Area [px]	Step [px]
(64, 64)	1	8	16	1
(128, 128)	2	16	32	2
(256, 256)	3	32	64	4

IV. Results and Discussion

Rank-1 accuracy scores were computed for every combination of the above stated parameters. The complete results can be found on this GitHub repository.

A. Results

The baseline pixel by pixel comparison results were mostly the same over all image sizes as seen in figure 1. The value for the euclidean metric, around 0.127, was then chosen as the baseline comparison as to see how different parameters improve/reduce this rank-1 accuracy. The below/above accuracy scores where counted and are presented in figure 2. These counts include only the LBP descriptor, with all parameter combinations, using the euclidean metric. In addition the mean and standard deviation for each value of four parameters was calculated and presented in table III.
Rank-1 accuracy results using default parameters are seen in table II

TABLE II: Rank-1 Accuracy with default parameters (metric euclidean).

pixel by pixel	LBP	ULBP	LBP + Hist	ULBP + Hist
0.127	0.235	0.028	0.242	0.011

Out of the 216 configurations of parameters where both our implementation of LBP/ULBP and the one from the `skimage` library where computed. The former proved to provide same or better results in 161 cases. Or in other words it proved superior in 74% of the cases.

TABLE III: Means and standard deviation of different parameters.

Histogram	mean	std
false	0.20	0.088
true	0.10	0.044

R(P)	mean	std
1(8)	0.14	0.064
2(16)	0.13	0.076
3(32)	0.12	0.081

Histogram Tile Size	mean	std
16	0.12	0.045
32	0.11	0.04
64	0.08	0.039

Image Size [px]	mean	std
64	0.13	0.071
128	0.13	0.071
256	0.12	0.057

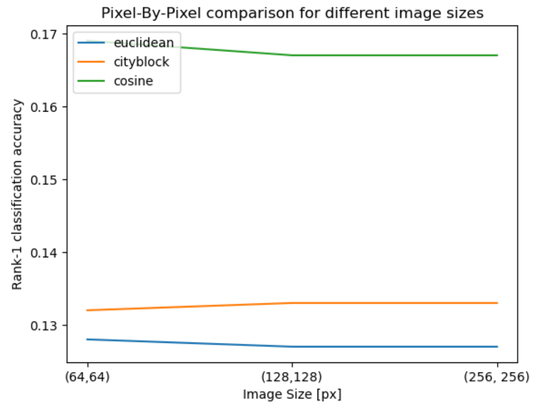

Fig. 1: Image size effects on the the pixel by pixel rank-1 accuracy for different metrics.

B. Discussion

Looking at all the results, generally the cityblock metric presented the best results, even if it was omitted in the results above. This is somewhat expected as the metric is best when dealing with binary patterns, which are results of both LBP and ULBP descriptors. It was interesting to note that most changes in the base parameters caused worse results than the baseline pixel by pixel recognition, as seen in figure 2. This is likely due to potentially incorrectly implemented descriptors, or loss of locality due to some parameters (like raising the step or histogram tile size parameters). Additionally the ULBP descriptor returned marginally worse results to the LBP descriptor and thus again isn't represented in the results.
Another made observation is how each parameter effected the rank-1 accuracy. This was made most apparent when introducing histograms and additionally changing the tile size. As stated previously the ladder made the accuracy worse as it was increased. The former produced better results using the default parameter values, but as seen in table III in general lowered the rank-1 accuracy. This is likely due to the connection to it's specific parameter, as well as general decreases in accuracy with all other parameters, as the larger set of tests were done with histograms than without and such has more data to influence the mean. Standard deviation was calculated in order to see if there is a parameter that didn't

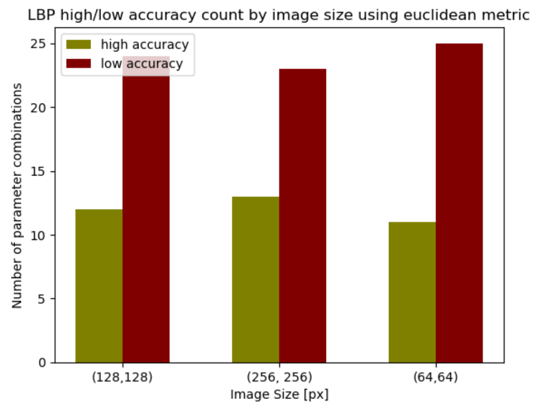

Fig. 2: The number of above/below parameter combinations using the euclidean metric as compared to pixel by pixle comparison.

function properly, as seeing a large standard deviation would mean that, that parameter had a large influence on the final outcome of the rank-1 accuracy.

V. Conclusion

Implementing the LBP and ULBP descriptors, returned varied results, mostly worse in comparison to the pixel by pixel classification method.

This shows that while the LBP descriptor can be good, it is much better used for detection than recognition. As seen in the images during testing, it provides good outlines to images, which can then be used for more detailed feature extraction and teaching a deep learning model, but it in general produced lower results due to loss of data.

During this assignment a basic understanding of the recognition process was developed, as well as some concepts for detection. The most important note is that localization is important, and any loss in data can effect the accuracy in large ways as seen in some of the results presented in this report.

To improve on this assignment, and possibly get better results, a different approach to recognition could be used, where we could split the data into a train and test set. Use the train set to learn and create model to predict classes of the test set.

References

[1] Z. Guo, L. Zhang, and D. Zhang, “A completed modeling of local binary pattern operator for texture classification,” *IEEE transactions on image processing*, vol. 19, no. 6, pp. 1657–1663, 2010.

[2] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[3] K. Salton de Prado, “Face recognition: Understanding lbph algorithm,” Nov 2017. [Online]. Available: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

[4] Ž. Emeršič, V. Štruc, and P. Peer, “Ear recognition: More than a survey,” *Neurocomputing*, vol. 255, pp. 26–39, 2017.