Universal distance measure for images

Uzi A. Chester
Electrical and Electronics Engineering Department, Ariel University Center of Samaria, Ariel 40700

Joel Ratsaby
Electrical and Electronics Engineering Department, Ariel University Center of Samaria, Ariel 40700

Abstract

We introduce an algorithm for measuring the distance between two images based on computing the complexity of two strings of characters that encode the images.

Given a pair of images, our algorithm transforms each one into a text-based sequence (strings) of characters. For each string, it computes the LZ-complexity and then uses the string-distance measure of [1] to obtain a distance value between the images.

The main advantages of our algorithm are that it is universal, that is, it neither needs nor assumes any spatial or spectral information about the images, it can measure the distance between two images of different sizes, it works for black and white, grayscale and color images, and it can be implemented efficiently on an embedded computer system.

We present successful experimental results on clustering images of different sizes into categories based on their similarities as measured by our algorithm.

I. Introduction

Image classification research aims at finding representations of images that can be automatically used to categorize images into a finite set of classes. Typically, algorithms that classify images require some form of pre-processing of an image prior to classification. This process may involve extracting relevant features and segmenting images into sub-components based on some prior knowledge about their context ([1],[2]). Typically in order for the classification to be accurate one needs to choose one of the many methods and algorithms based on whatever side-information that one may have about the images that one expects to classify. There is no universal best algorithm that is guaranteed to yield high rates of classification for all kinds of problems.

However, in the area of text recognition there have recently been new string-distance measures [3],[4] that can compare any two strings of characters without any assumption on their context. These distances have been shown to be successful on a variety of pattern recognition tasks of data clustering and pattern classification.

In this paper we introduce a new image distance measure which is based on such string distances. We are able thus to measure distances between any pair of images and do classification and clustering of images without any feature extraction.

II. Theory

A. LZ – complexity

The distance that we introduce is based on the LZ-complexity of a string. The definition of this complexity follows [3]: let S, Q and R be strings of characters that are defined over the alphabet A, $l(S)$ is the length of S, $S(i)$ is the i th element of S and $S(i,j)$ defines the substring of S composed of characters of S between position i and j. An extension $R = SQ$ of S is *reproducible* from S (denote $S \to R$) if there exists an integer $p \le l(S)$ such that $Q(k) = R(p+k-1)$ for $k = 1,...,l(S)$. For example $'aacgt' \to 'aacgtcgtcg'$ with $p = 3$ and $'aacgt' \to 'aacgtac'$ with $p = 2$. R is obtained from S (the seed) by copying elements from the p th location in S to the end of S.

A string S is *producible* from its prefix $S(1,j)$ (denoted $S(1,j) \Rightarrow R$), if $S(1,j) \to S(1, l(S)-1)$. For example $'aacgt' \to 'aacgtac'$ and $'aacgt' \to 'aacgtacc'$ both with pointers $p = 2$. The production adds an extra 'different' char at the end of the coping process which is not permitted in reproduction.

Any string S can be built using a *production process* where at its i th step we have the production $S(1,h_{i-1}) \Rightarrow S(1,h_i)$ where h_i is the location of a character at the i th step. (note that $S(1,0) \Rightarrow S(1,1)$).

An m - step production process of S results in parsing of S in which $H(S) = S(1,h_1) \cdot S(h_1+1,h_2) \cdots S(h_{m-1}+1,h_m)$ is called the *history* of S and $H_i(S) = S(h_{i-1}+1,h_i)$ is called the i th component of $H(S)$. For example for $S = 'aacgtacc'$ $H(S) = 'a \cdot ac \cdot g \cdot t \cdot acc'$ is the history of S.

If $S(1,h_i)$ is not reproducible from $S(1,h_{i-1})$ then $H_i(S)$ is called *exhaustive* meaning that the copying process cannot be continued and the component should be halted with a single char *innovation*. Moreover, every string S has a unique exhaustive history [5][1].

978-1-4244-8682-3/12/$26.00 ©2012 IEEE

We let $c_H(S)$ be the number of components in a history of S. Then the LZ complexity of S is $c(S) = \min\{c_H(S)\}$ over all histories of S [1]. It can be shown that $c(S) = c_E(S)$ where $c_E(S)$ is the number of components in the exhaustive history of S.

III. The Distance Function

A. Sayood distance

According to [3], given two strings X and Y, we consider the concatenation XY and define the function $d(X,Y)$ as

$$d(X,Y) = \max\left(c(XY) - c(X), c(YX) - c(Y)\right).$$

The normalized version of $d(X,Y)$ is defined as

$$d^*(X,Y) = \frac{max(c(XY) - c(X), c(YX) - c(Y))}{max(c(X), c(Y))}$$

In this paper we use another version of this distance which is defined as

$$d^{**}(X,Y) = \frac{c(XY) - \min\left(c(X), c(Y)\right)}{\max\left(c(X), c(Y)\right)}$$

It resembles the Normalized Compression Distance of [4] except that here we do not use a compressor but rather the LZ-complexity *c* of a string. Note that d^{**} is not a metric since it does not satisfy the triangle inequality and a distance of 0 implies that the two strings are close (but not necessarily identical). This distance is universal in the sense that it is not based on some representation or heuristic features as many of the other string distances. Instead it makes reference only to their LZ-complexity which is purely an information quantity independent of their context or representation.

IV. Universal Image Distance (UID)

In this section we introduce an image distance measure which we call Universal Image Distance (UID). It is based on the universal distance d^{**} where the idea is to transform the two images into strings of characters. For grayscale images we use the regular ASCII codes thus the alphabet consists of 256 characters. For color images we use the UTF-8 character table.

We first describe how the UID is computed for a pair of grayscale images (in section V.B we explain how it is defined for color images).

We start with a pair of grayscale images: $\mathbf{x} = [x_1, x_2, ..., x_{Nx}]$ and $\mathbf{y} = [y_1, y_2, ..., y_{Ny}]$ where N_x and N_y are the number of pixels in images $\mathbf{x}$ and $\mathbf{y}$ respectively. We first convert each pixel into a binary representation (B/W) using a threshold of 128 for decision which is the half point of the range of the grayscale range [0,255]. Then we take two groups of eight B/W pixels that are aligned one on top of each other and encode them into two bytes. As displayed in Figure 1 a sliding window consisting of these two groups moves to the next two groups by shifting one pixel at a time. We convert into a single character the two-byte code obtained from the two eight pixel groups

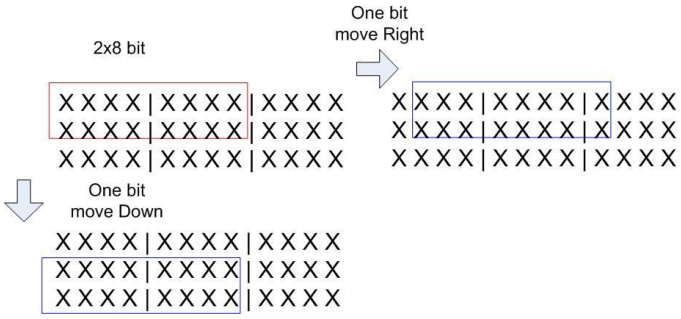

Figure 1

The 16 bit codeword obtained in this way is encoded into an 8-bit codeword by a simple 2D uniform quantization procedure. There are 256 prototypes each with a two-dimensional vector with both components in the range [0,255]. Each prototype is associated with a unique ANSI symbol.

Given a 16-bit codeword based on the sliding window we map it into one of the 256 prototypes by minimizing the two-dimensional Euclidian distance. The symbol corresponding to the closest prototype is then used to represent the sliding-window's current value.

Thus a picture is transformed into a string of symbols. Figure 2 shows an example of this transformation process: the original gray-scale image is on the left, middle is the B/W image after thresholding, the right image is a gray-scale representation after quantization, where we take the character sequence (string) and display the ASCII codes in terms of gray-scale values. As can be seen, the information content in this character sequence resembles to a high extent that of the original image.

Figure 2

With the two images **x** and **y** represented as strings we then apply the formula for d^{**} to get the UID distance between **x** and **y**.

V. Experiments and Results

A. Black and white image similarity measurements

In order to check the UID we performed clustering on a group of black and white fruit images, shown in Figure 3, and vehicles shown in Figure 4 and thereby discovered the inter-image similarities. The results show that the UID distance reflects well the different level of

dissimilarity amongst the images. As shown in Figure 5 and 6 clustering yields well the expected clusters.

Note that clustering by UID works well even while the images are of different size and resolution.

Figure 3

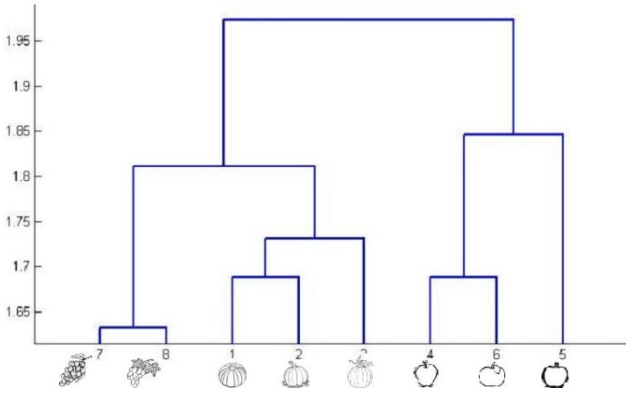

Figure 4

Figure 5

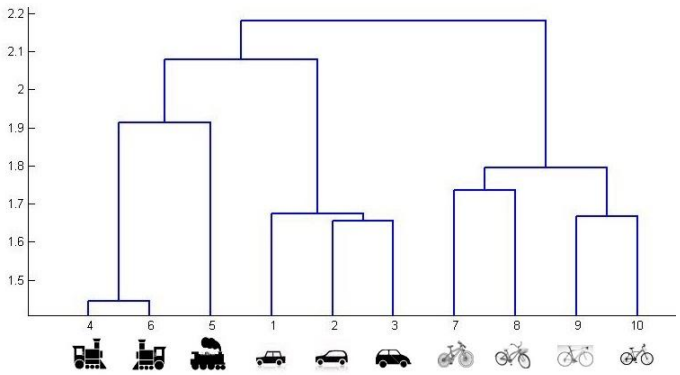

Figure 6

B. Colored-image similarity measurements

We tested the UID distance on color images where instead of the 2 by 8 sliding window described in section IV we now have a single-pixel window and use uniform quantization on a pixel by pixel basis.

Each of the three R,G,B colors forms a dimension in this representation and is quantized separately into 10 quantization levels. Hence we have 10x10x10=1000 prototypes uniformly spread over a three-dimensional domain. Each prototype is now associated with a UTF-8 character (we cannot use ASCII since it is limited to 256 characters). Each pixel is transformed into a single character in an alphabet of size 1,000. Thus an image is transformed into a string of UTF-8 characters.

To test how well the UID distance works for color images we created a database of 16x16 color-image templates that are representative of three image categories: **sea**, forests (**tree**) and urban (**city**) regions (Figure 7)

We then obtained a set of real satellite images (each of size 272x192) to be classified into these three categories. We classify them by a *soft-label*, that is, we express the *percentage of similarity* between the satellite image to each of the three categories.

We describe the algorithm next (*N* stands for the number of categories):

Algorithm - Universal Image Classifier (UIC)

Input: k_i template images of size r x s from category i, $1 \le i \le N$, and a color image P to be classified.

1. Let $M = \sum_{i=1}^{N} k_i$ be the number of templates. Denote by H the collection of templates, $H = \{p_1, p_2, \ldots, p_M\}$.
2. Initialize the count variables $A_i = 0$, $1 \le i \le N$.
3. Scan P from top left to bottom right, and segment the image into a sequence of sub-images of P each of size r x s.

   Denote this collection of sub-images by $C = \{q_1, q_2, \ldots, q_L\}$.

4. For $1 \leq i \leq L$, get sub-image $q_i \in C$ and find the template $p^* \in H$ which minimizes the distance to q_i, that is,

$$UID(q_i, p^*) = min_{p \in H} UID(q_i, p).$$

If p^* is from category j then increment the count A_j, where $1 \leq j \leq N$.

Output: soft-classification of image P as: $\frac{A_j}{L} \times 100$ percent from category j, $1 \leq j \leq N$.

As can be seen in Figure 8,9,10 the UIC classifier succeeds to classify these real satellite images. We can it to automatically extract the context of any given image and estimate the percentage of sea, tree and city in the image. This can easily be expanded to a larger set of categories. We note that UIC makes no assumption on the given image and no feature extraction of any kind.

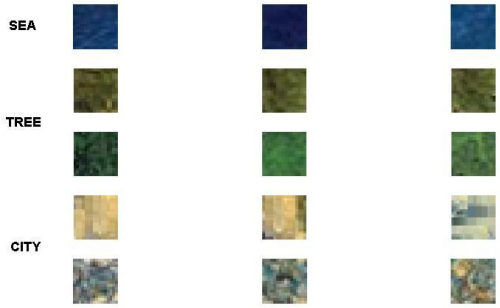

Figure 7

Figure 8: Sea: 61.4%, Tree: 22.8%, City: 15.7%

Figure 9: Sea: 34.3%, Tree: 48.6%, City 17.1%

Figure 10: Sea: 0%, Tree: 23.4%, City 76.6%

VI. Conclusion

We introduced a new distance function UID for a pair of images. The UID measure is universal in that it does not necessitate any type of image extraction, pre-processing nor any domain knowledge about the type of images.

The only processing done is a simple conversion of the image into a string of characters based on a simple uniform quantization. Once in a string format, we measure the LZ-complexity of each of the images in the pair and their concatenation and plug the values into a formula of the Sayood distance.

We also introduced a new image classification algorithm (UIC) which uses the UID to soft-classify a given image.

Based on clustering and multi-category classification experiments that we have performed on several gray-scale and color images, we established that the distance function UID captures correctly the dissimilarity between images.

References

[1] T. Lillesand, R. W. Kiefer, J. Chipman, *Remote Sensing and Image Interpretation*, 2008.

[2] M. J. Canty, *Image Analysis, Classification and Change Detection in Remote Sensing, with Algorithms for ENVI/IDL*, 2010.

[3] H. H. Otu and K. Sayood, A new sequence distance measure for phylogenetic tree construction. *Bioinformatics*, 19 (16):2122-2130, 2003.

[4] R. Cilibrasi and P.M.B. Vit´anyi, Clustering by Compression. *IEEE TRANSACTIONS ON INFORMATION THEORY*, VOL. 51, NO 4, pp. 1523–1545, 2005.

[5] Lempel, A. and Ziv,J., On the complexity of finite sequences, *IEEE T. Inform. Theory*, 22, 75–81, 1976.