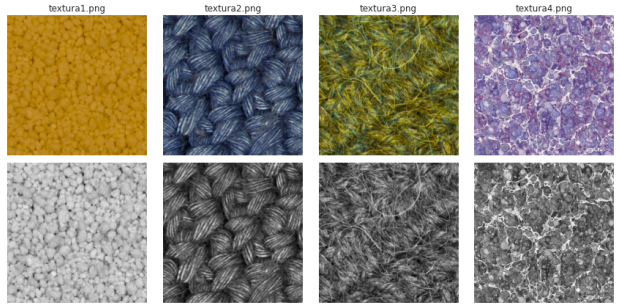
Lucas David, 188972 | Texture Descriptors | lucas.david@ic.unicamp.br

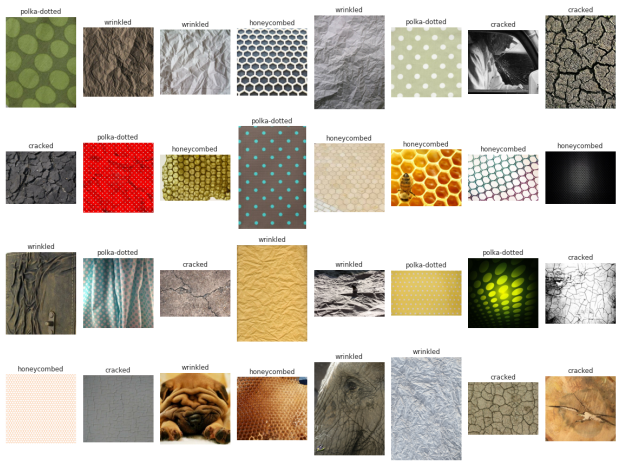
# 1 Introduction

This assignment was submitted to the class of 2021/1 of course Introduction to Image Processing (MO443) at Universidade Estadual de Campinas. Its goal is to apply texture-describing methods to extract descriptors from image samples.

## 1.1 Dataset and Setup




(a)



(b)

Figure 1: Examples of images (a) containing simple textures made available in class and (b) in the DTD dataset.

I employed Scikit-Image [6] and Google Colaboratory [1] when developing this assignment. The notebook produced is available for direct access[1].
Two sets of images were utilized in this assignment, and are illustrated in Fig. 1. First, four considerably distinct texture images provided in class are used to illustrate variations in the parameters of the texture descriptor extraction algorithms. The second set used is the Describable Textures Dataset (DTD)[2], which can be downloaded in the tf-records format using the *tensorflow-datasets* library. This dataset represents a multi-class (mono-label) texture classification problem, and comprises 5,640 images associated with one of the 47 available textures. In order to simplify the problem (for demonstration purposes of this assignment), I opted to filter images of one of the following labels: *honeycombed, cracked, polka-dotted* and *wrinkled*.

[1] Iterative report available at colab/mo-443-assignment-4
[2] DTD dataset is available at tensorflow.org/datasets/catalog/dtd

# 2 Local Binary Patterns

In this section, we describe the application details of Local Binary Patterns in texture images.

Local Binary Patterns (LBP) is an effective texture extraction method that demonstrated competitive results in texture analysis and facial recognition applications [5, 2]. Given a gray-scale image, this method consists of defining a pixel neighborhood based on free parameters *radius* and *number of pixels*, and thresholding each pixel within a single neighborhood according to its central pixel. Pixels with higher intensity than the central one are assigned 1, while pixels with lower intensity are assigned 0. Finally, the binary mask is multiplied by its weight matrix. While LBP considers in its original formulation the weight matrix defined by a predetermined ordering scheme over the sequence $w_n = [2^n]_{N^2-1}$, where $N \times N$ is the neighborhood defined, strategies to learn said matrix have been proposed [2].
As remarked by Song et al. [5], the correlation among pixels decreases as the radius increases, which in turn deteriorates the local texture information. Thus, considering the scope of this assignment, low radii values were used in my experiments.

## 2.1 Impact of parameters over LBP

Proposed activity: present the Local Binary Patterns for a few images used in the experiments.

```
from skimage import io, color, feature

gray = color.rgb2gray(image)
lbp = feature.local_binary_pattern(
    gray,
    P=8,
    R=3,
    method='default'
)
```

Listing 1: Applying the LBP method over an image.

Scikit-Image provides an implementation for the LBP method. The code in Lst. 1 illustrates the expected configuration in order to reproduce LBP's original formulation.

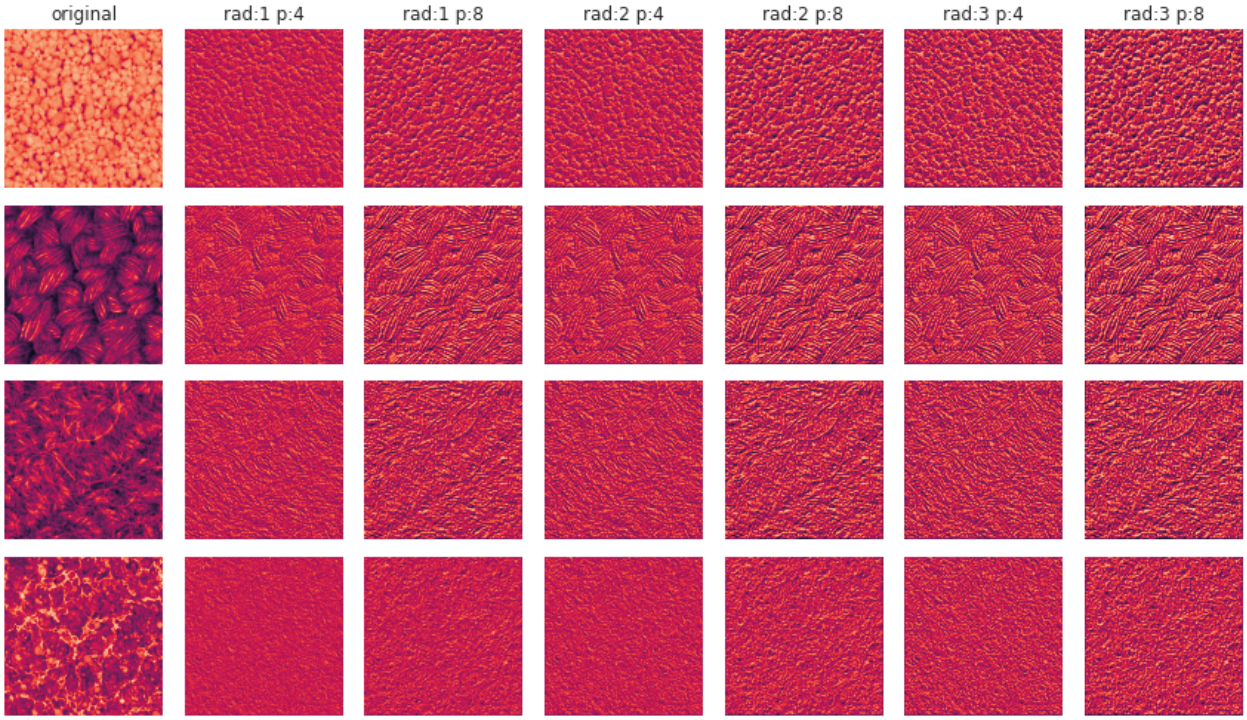



Figure 2: LBP of simple texture images.

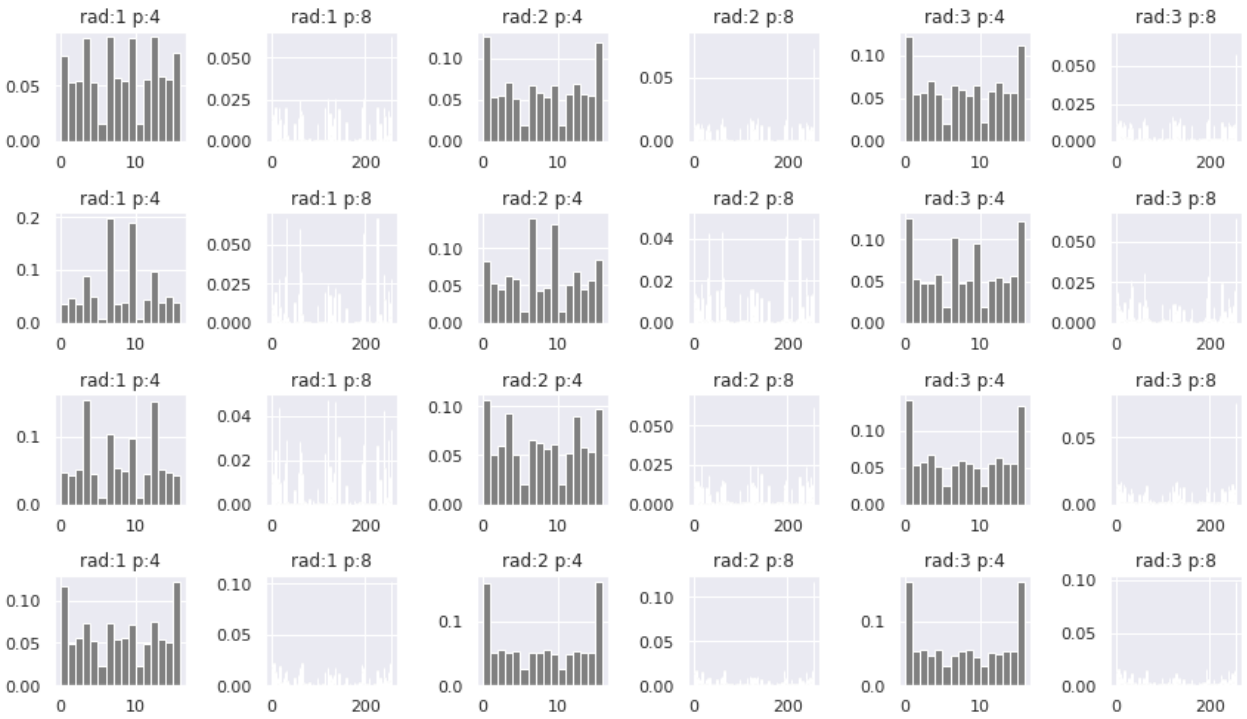



Figure 3: Histogram of LBP of simple texture images.

Fig. 2 illustrates the results of the LBP method over four simple texture images with a varying set of parameters *radius* and *number of points*. A black-red color map is used to make the patterns easier to distinguish. A higher radius and number of points produces stronger, more varying strokes in the output pattern images. Conversely, a low number of points produces more homogeneous patterns.

The histograms presented in Fig. 3 were extracted from the gray-scale intensity patterns shown in Fig. 2. As expected, configurations that generated more homogeneous patterns resulted in histograms with fewer bins (16), while more varying patterns were summarized in histograms with 256 bins.

## 2.2 Matching Textures based on LBP Histograms

Proposed activity: use a few texture images (other than the ones provided in class) and compare them with respect to their Local Binary Patterns histograms.

The remaining results for the LBP method presented in this work were obtained considering `R=3`, `P=8` and `method='default'`. Fig. 4 illustrates the application of the LBP method over images in the DTD dataset considering these same parameters.
In order to match images according to their texture, one must first compute the pairwise dissimilarity among the samples in the set. This can be achieved by leveraging the `cdist` function contained in the python package `scipy`. Multiple distance functions are available in this function, such as Euclidean and Correlation. I used the Kullback–Leibler divergence function in this step, which is commonly employed when comparing two probability distributions. Lst. 2 describes the steps necessary for the construction of the dissimilarity matrix between samples in the DTD dataset, which is illustrated in Fig. 5a. We observe that the subset of images associated with the label *cracked* seems to form a dark triangle (a closely related cluster) on the top-left corner of the matrix. The remaining labels present a more varying set of distances, indicating they would be more easily confused with each other.
With the dissimilarity matrix in hand, one can find the most similar texture matches by argument-sorting each row in the matrix and retaining the $k$ most similar indices.

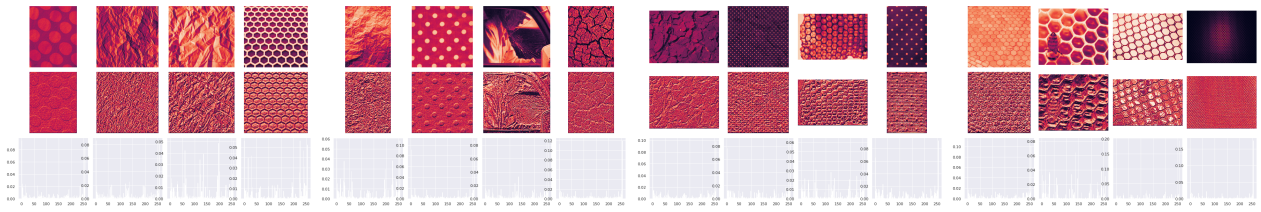


Figure 4: Examples of application of the LBP method over images in the DTD dataset.

```
from scipy.spatial.distance import cdist

lbps = [
  feature.local_binary_pattern(
    gray,
    P=8,
    R=3,
    method='default')
  for gray in images]
bins = [int(l.max()+1) for l in lbps]

hists = [
  np.histogram(
    l,
    density=True,
    bins=b,
    range=(0, b))[0]
  for l, b in zip(lbps, bins)]

dst = cdist(
    hists,
    hists,
    metric=kullback_leibler_divergence
)
```

Listing 2: The construction of the dissimilarity matrix between samples in the DTD dataset.

# 3 Gray Level Co-occurrence Matrices

Similar to LBP, Gray level co-occurrence matrix (GLCM) is a popular method used to extract texture statistics from images. It consists of creating a $G = [p_{ij}]_{l \times l}$ matrix, where $l$ is the number of distinct gray intensities in the original image and $p_{ij}$ is the relative frequency in which two pixels occur with the intensities $i$ and $j$, respectively, given the predefined free parameters *distance* and *angle* [3]. GLCM is frequently followed by the extraction of conventional statistical descriptors, such as angular second moment, inverse difference moment, entropy, and correlation [3].

```
from skimage.feature import greycomatrix,
    greycoprops

distances = [1, 2, 3, 4]
angles = [0, np.pi/4, np.pi/2, 3*np.pi/4]
props = ('dissimilarity', 'correlation')

def to_uint8(x):
  return (x*255).astype('uint8')

def glcm_features(x):
  co = greycomatrix(
    to_uint8(x),
    distances=distances,
    angles=angles,
    levels=256,
    symmetric=True,
    normed=True
  )

  return np.asarray([greycoprops(co, prop=p)
                     for p in props]).ravel()
```

Listing 3: The application of the GLCM method over samples in the DTD dataset.

Lst. 3 describes the application of the GLCM over grayscale image samples using the Scikit-Image library. In the scope of this work, the co-occurrence matrix was built considering low distances (1, 2, 3 and 4) and multiple angles (0°, 45°, 90° and 135°). Additionally, a pairwise dissimilarity matrix can be built on top of the features extracted by GLCM, similarly to what was performed in Section 2.1. However, as features extracted here do not form a probability distribution, the *cosine* distance function was selected to evaluate the dissimilarity between pairs of texture samples. Fig. 5b

illustrates the cosine dissimilarity matrix between samples in the DTD dataset. The average absolute distance measurement is considerably lower than the values found in 5a, making it harder to distinguish. However, it is clear that many values in the diagonal line immediately below the matrix's main diagonal are extremely close to 0, indicating these samples have close associations with at least one other label-sharing sample.

# 4 Evaluation

As the DTD dataset associates each image to a texture label, metrics such as *accuracy score* and *top-k accuracy score* — commonly employed when validating machine learning systems [4] — can be used to evaluate the efficacy of the matching given by the two sets of features described in Section 2 and Section 3.
In order to conduct a more statistically stable experiment, we used all of the 160 samples in the DTD dataset associated with the subset of labels considered in Section 1.1.

| Metric | LBP | GLCM |
|---|---|---|
| top-1 | 46.88% | 44.38% |
| top-3 | 74.38% | 76.25% |
| top-5 | 84.38% | 85.00% |

Table 1: Evaluation results for the two matchings produced from LBP and GLCM features.

Table 1 shows the evaluation results for both methods. We observe that LBP performs slightly better when retrieving the most similar match (top-1 score). On the other hand, GLCM showed better accuracy when considering the three and five most similar matches (top-3 and top-5 scores).

# References


[1] Tiago Carneiro et al. "Performance analysis of google colaboratory as a tool for accelerating deep learning applications". In: *IEEE Access* 6 (2018), pp. 61677–61685.

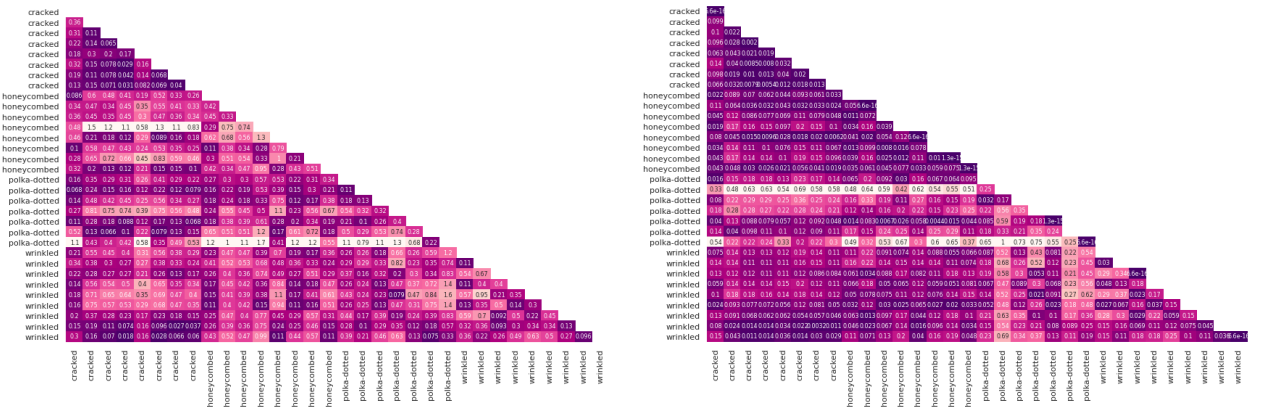
[2] Felix Juefei-Xu and Marios Savvides. "Weight-optimal local binary patterns". In: *European Conference on Computer Vision*. Springer. 2014, pp. 148–159.

[3] P Mohanaiah, P Sathyanarayana, and L GuruKumar. "Image texture feature extraction using GLCM approach". In: *International journal of scientific and research publications* 3.5 (2013), pp. 1–5.

[4] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation". In: *AI 2006: Advances in Artificial Intelligence*. Ed. by Abdul Sattar and Byeong-ho Kang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1015–1021.

[5] Ke-Chen Song et al. "Research and Perspective on Local Binary Pattern". In: *Acta Automatica Sinica* 39 (June 2013), pp. 730–744.

[6] Stefan Van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (2014), e453.

(a) (b)

Figure 5: Pairwise dissimilarity matrix between (a) LBP histograms of samples in DTD dataset; (b) GLCM features extracted from samples in the DTD dataset.

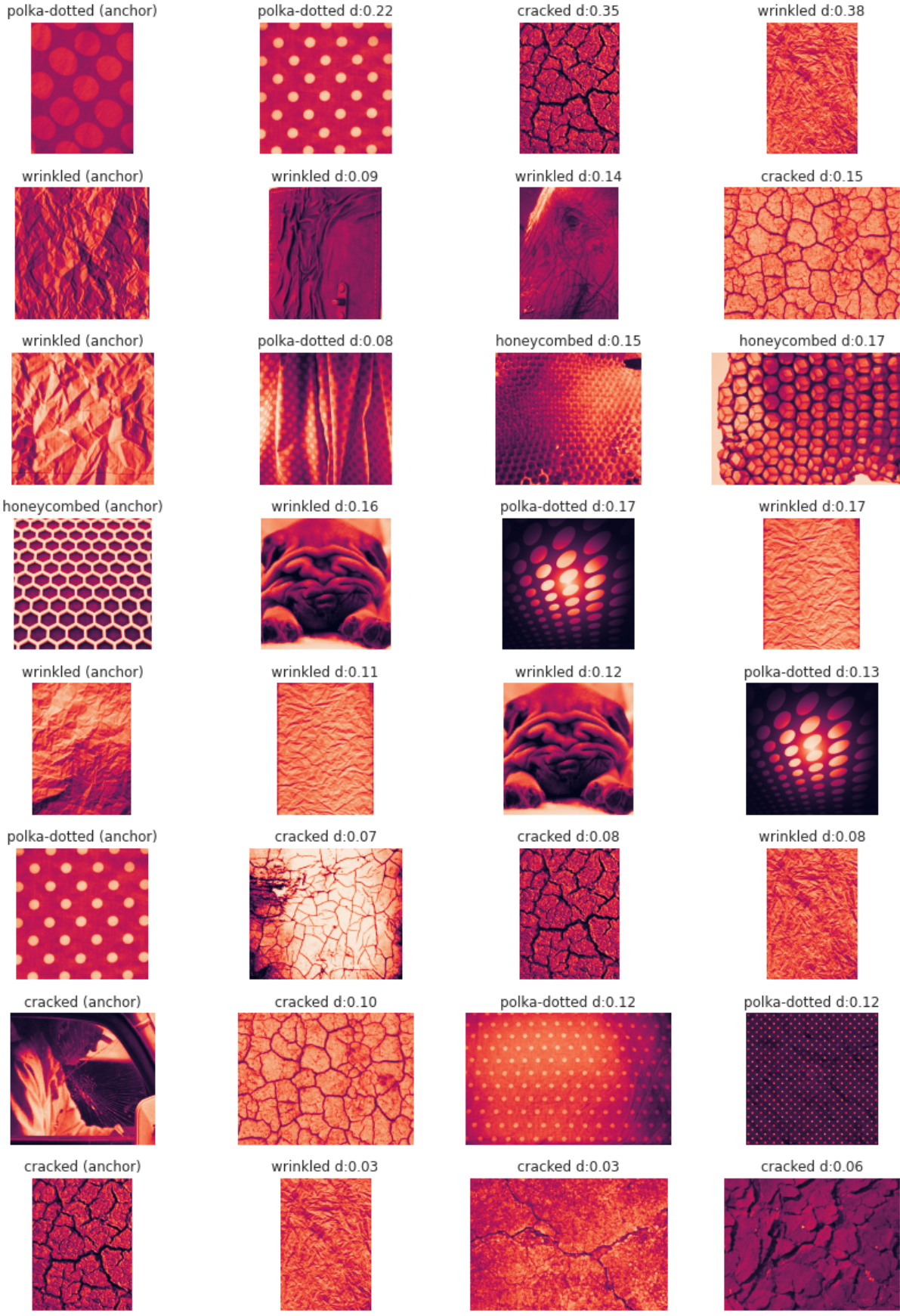



Figure 6: Most similar matches for 8 samples in the DTD dataset.