Pattern Recognition Letters
journal homepage: www.elsevier.com

Noise-Resistant LBP Descriptor for Texture Recognition - A Supervised Approach

Priya Sen Purkait[a,**], Rishideep Chatterjee[a], Sulogna Roy[a], Sanando Chanda[a], Debasmita De[a], Hiranmoy Roy[b]

[a]*Dept. of Computer Science & Engineering, RCC Institute of Information Technology, India*
[b]*Dept. of Information Technology, RCC Institute of Information Technology, India*

ABSTRACT

The Local Binary Pattern (LBP) descriptor is widely used for texture recognition due to its simplicity, computational efficiency, and effectiveness. However, LBP is sensitive to noise and variations in lighting conditions, which can hinder its performance in real-world applications. This paper aims to develop a noise-resistant LBP descriptor to enhance texture recognition accuracy while minimizing computational cost and maintaining efficiency. The proposed noise-resistant LBP descriptor includes four steps: preprocessing the image, extracting LBP features, applying a noise-resistant thresholding technique, and classifying the texture pattern using a machine learning algorithm. Preprocessing involves converting images to grayscale and applying filters to remove aberrations. LBP features are extracted, followed by a novel thresholding technique to enhance noise resistance. Finally, machine learning algorithms classify the texture patterns. Experimental results demonstrate that the proposed noise-resistant LBP descriptor outperforms traditional LBP methods and other variants in texture recognition tasks under noisy conditions. The descriptor shows significant improvements in accuracy and robustness across various datasets. Thus, the noise-resistant LBP descriptor developed in this study provides a new and efficient solution for texture recognition.

1. Introduction

An image texture is a set of metrics calculated during image processing, developed to compute an image's perceived texture. It gives us information about the spatial arrangement of color or intensities in an image or a region of an image. This is primarily done to substantiate our perception with quantifiable measures. By definition, Texture recognition is the task of identifying such an image texture. This task is critical in many domains, such as computer vision, medical imaging, material sciences, and often in space sciences. Many texture recognition methods have been proposed in the literature, including statistical methods, filter-based methods, and transform-based methods. In recent years, local feature-based methods have become popular due to their effectiveness, computational efficiency, and increased level of accuracy.
The local binary pattern (LBP) descriptor is a widely used 'local feature-based' method for texture recognition. LBP was first introduced by Ojala et al. in 1996

In this paper, a noise-resistant LBP descriptor for robust texture recognition is proposed. The goal is to minimize the computational cost and enhance efficiency. The method combines the LBP descriptor with a noise-resistant thresholding technique to improve recognition accuracy while maintaining said efficiency. The rest of this essay is structured as follows: A brief survey of the literature is given in Section II in the context of existing local-binary pattern descriptors. In Section III, the suggested approach is thoroughly explained, while Section IV presents the experimental results and analysis. The paper is finally concluded in Section V.

2. Literature Review

Texture recognition is a fundamental task in computer vision, with applications ranging from image classification to object detection and segmentation. The simplicity, usefulness, and computational efficiency of the Local Binary Pattern (LBP) descriptor have made it a popular technique for texture analysis. . Local binary patterns are a powerful local descriptor

**Corresponding author: -
e-mail: priyasen.purkait@rcciit.org.in (Priya Sen Purkait)

for microstructures of images. It has been widely used in face recognition systems due to its robustness and computational efficiency (1). The new and effective facial image representation presented in this study is based on textural properties known as local binary patterns (LBP). The face image was separated into multiple areas from which the distributions of the LBP features were taken. It was also combined to create an enhanced feature vector to be utilized as a facial descriptor. The performance of the proposed method is assessed in the face recognition problem under different challenges. The effectiveness of the suggested approach is evaluated in the face recognition task under various conditions(1). A wide range of facial recognition algorithms have already been presented by numerous researchers, depending on the applications. A new and effective method was explained by Timo Ahomen et al. in 2006 for representing faces using texture features based on LBP.

The paper by Guoying Zhao et al. (2) addresses dynamic texture (DT) recognition, an extension of texture analysis into the temporal domain. The authors propose a novel approach using Volume Local Binary Patterns (VLBP) to model DTs, which combine motion and appearance. Their technique focuses on local processing, robustness to monotonic grey-scale changes, and straightforward computing.

The two major difficulties in Heterogeneous Face Recognition (HFR) stem from the notable intra-class variation between the query and gallery images and the challenging task of training a deep network for many cases. To tackle these two major issues, two models were designed by Hiranmoy Roy et al. in 2016 and 2022 respectively (3) (4). The former is Local Gravity-Face (LG-face), a unique approach for recognizing heterogeneous and illumination-invariant faces (HFR). LG-face uses a concept called Local Gravitational Force Angle (LGFA), which simply takes the reflectance component of the local texture impact of the surrounding pixels and it is an illumination-invariant characteristic. The latter is an interpretable deep learning model named, Local Frequency Binary Pattern (LFrBP) which considerably enhances the performance of the network. The Local Discrete Cosine Transform Binary Pattern (LDCTBP) is another new face image descriptor that can recognize faces regardless of illumination or modality by Subhadeep Koley et al. in 2023 (5). The suggested LDCTBP was coupled with a lightweight Convolutional Neural Network (CNN) to demonstrate the significance of handcrafted features in CNN training. Another technique has been proposed by Jiwen Lu et al. in 2017 for both homogeneous and heterogeneous face recognition, named Simultaneous Local Binary Feature Learning and Encoding for Homogeneous and Heterogeneous Face recognition (SLBFLE) (6). SLBFLE is an unsupervised feature learning method that automatically learns facial representation from raw pixels. It uses a one-stage feature learning and encoding approach to gather discriminative information from raw pixels in facial images of distinct identities by learning binary codes and local face patch codebooks.

An object descriptor named Local Derivative Pattern (LDP) was explained by Baochang Zhang et al. in 2010 to represent the high-order variations in local derivatives (7). They carried out the experiments by using diverse facial situations, such as altered lighting, expressions, age, accessories, and posture. The results indicate that high-order local patterns (LDPs) outperform first-order local patterns (LBPs). A range of studies have evaluated the performance of local descriptors in computer vision applications. The performances of different descriptors have been evaluated and also computed for areas of local interest by Mikolajczyk et al. in 2005 (8). For various interest region types, they have compared shape context, steerable filters, differential invariants, spin pictures, Scale Invariant Feature Transform (SIFT), complex filters, moment invariants, cross-correlation etc. Another image descriptor named Pattern of Local Gravitaional Force (PLGF) has been proposed by Bhattacharjee et al. in 2021 which outperforms existing descriptors in various image recognition tasks (9). Here, the histograms of the two components are concatenated to construct the PLGF descriptor A new feature descriptor for the 3D palmprint recognition approach using structured light matching has been introduced by Zhang et al. in 2008 (10). These studies collectively demonstrate the potential of 3D feature descriptors in palmprint recognition. Another local robust image descriptor, Weber Local Descriptor (WLD), based on human perception and consisting of differential excitation and orientation components was described by Chen et al. in 2010 (11).

A series of studies have explored the use of Local Tetra Patterns (LTrPs) as a feature descriptor for content-based image retrieval. Murala et al. (2012) proposed a generic strategy for computing nth-order LTrPs, which was found to be more effective than other methods i.e. it has demonstrated the potential of LTrPs as a powerful tool for content-based image retrieval (12).

A series of studies have explored the development of discriminant face descriptors for face recognition. Roy et al. (2018) introduced a novel method for heterogeneous face recognition, the Quaternary Pattern of Local Maximum Quotient (QPLMQ), which is particularly effective for sketch-photo and near infrared-visible image matching. This method outperforms other state-of-the-art methods, including deep learning-based ones (13). Again, for face recognition under difficult lighting situations, a number of research have suggested enhanced local texture feature sets(14). In this paper, the local distance transform-based similarity metric improves LBP/LTP-based face recognition performance. Chakraborty et al. (2022) have described the Local Gradient Hexa Pattern (LGHP) as a robust descriptor for face recognition and retrieval, outperforming other state-of-the-art descriptors(15). Comparing the suggested descriptor to state-of-the-art descriptors, the former has higher recognition and retrieval rates. Lei et al. (2014) proposed a learning-based mechanism to enhance the discriminative ability of face descriptors, leading to improved face recognition performance(16). Simultaneously, the discriminant image filters and the best weight assignments of nearby pixels are learned to strengthen the descriptor's capacity for discrimination.

There are several difficulties with face recognition, like pose variations, occlusions, illumination and noise variations, expression variations etc. Challenges due to illumination and noise variations are addressed by Koley et al. (2021) using shearlet-based Gammodian Binary Pattern (GBP) (17). Again, a range of studies have explored the development of compact binary face descriptors for face recognition. Several face recognition systems have made extensive use of local binary patterns and their variants because of their exceptional robustness and potent discriminative power. Lu et al. have explained methods for learning compact binary face descriptors (18). An effective method for illumination-invariant and heterogeneous face recognition method has been proposed by Roy et al. in 2016 for achieving high recognition rates under varying illumination and different modalities (3). Bhattacharya et al. (2019) developed the Local Force Pattern (LFP) as a feature descriptor for heterogeneous face recognition, encoding directional texture information (19). Roy et al. in 2021 proposed the Local-Friis-Radiation-Pattern (LFRP) for both homogeneous and heterogeneous face recognition, incorporating the Friis equation to capture illumination and modality-invariant features (20). The proposed LFRP is robust against face recognition under non-permanent facial cosmetics. These studies collectively highlight the potential of compact binary face descriptors in face recognition.

A survey of instance retrieval techniques, including SIFT and Convolution Neural Network (CNN) based algorithms, was carried out by Zheng et al. in 2015 for a decade (21). This survey highlights the evolution of these methods and their performance on various datasets, particularly in the context of deep learning and CNN-based methods. In 2017, He et al. presented the Wasserstein CNN, an approach that outperforms the traditional techniques for learning invariant characteristics between near-infrared (NIR) and visual (VIS) face images (22). Another work in this field done by Liu et al. (2016) presented the deep Transfer NIR-VIS heterogeneous face recognition network (TRIVET), which achieves state-of-the-art performance by transferring discriminative models from unpaired VIS images to the NIR-VIS domain using deep CNNs and ordinal metrics (23). Lightweight CNN models for face recognition in noisy situations are proposed by Guo (2019) and Wu (2015) (24) and (25). To increase accuracy, Guo's LD-MobileFaceNet makes advantage of swish nonlinearity and a denoising block while Wu's Illumination To deal with noisy labels, CNN uses semantic bootstrapping and a max-feature-map activation. These works demonstrate how deep learning methods can be used to overcome the difficulties associated with NIR-VIS face identification.

Texture analysis of medical images is a complex and evolving field, with a range of applications including segmentation, lesion detection, and tissue differentiation (26). A novel and robust state-of-the-art objection detection and classification model, YOLOv8 has been proposed for skin cancer classification (2023). 1000 pre-defined classes can be instantly classified the disease in real-time images, ensuring outstanding precision in accuracy by this model (27). The use of automated and computer-aided detection systems with artificial intelligence is particularly promising, but they face challenges in image acquisition, pre-processing, segmentation, and data management (28). Despite these challenges, these techniques are crucial for meeting the needs of a growing patient population and improving the healthcare system.

The use of gray scaling is also quite common. Ojala et al. (2001) and Ojaha et al. (2002), in their papers (29) and (30) both talk about a multiresolution gray-scale and rotation invariant texture classification methods based on LBP. They identify "uniform" LBP patterns as crucial texture features and derive a generalized operator for rotation invariance. Using local frequency components for texture classification, Manni et al. conducted more work in 2013 using the rotation invariant technique (31). The suggested approach can significantly increase classification accuracy and is highly resistant to noise, especially when there is a lot of noise present. These methods are computationally simple and robust, with the latter introducing a generalized LBP operator for any quantization of the angular space and spatial resolution.

The work of Zhenhua Guo, Lei Zhang, and David Zhang (32) employs a local region's centre pixel and a Local Difference Sign-Magnitude Transform (LDSMT) to represent texture through global thresholding. Again, In a nonparametric approach, M. Topi et al. (33) show that a carefully chosen subset of patterns derived from LBP offers an efficient texture description, outperforming the entire LBP histogram in classification accuracy. The work by Topi et al. (34) proposes a multi-predicate version of the LBP, making it useful for textures with multiple scales. The usage of the Enhanced Fisher Model (EFM) is highlighted in a paper by Sugata Banerji, Abhishek Verma, and Chengjun Liu (35). Four novel colour LBP descriptors are proposed in the work and the classification relies heavily on the nearest neighbour classification rule (EFM-NN). A study on a fusion strategy built around a consistent local quinary pattern (LQP) and a rotation invariant local quinary pattern was conducted by L Nanni, A Lumini and S Brahnam (36). In another paper, N Kazak, M Koc (37) introduce the use of symmetric two-spiral LBP to measure grayscale differences between the centre pixel and its neighbours.

A range of studies have explored the use of extended local binary patterns for texture classification. Liao et al. in 2010 (38) introduced the extended local ternary pattern (ELTP), which demonstrated improved noise resistivity and effectiveness in texture classification compared to the original LBP. Mäenpää et al. in 2003 (39) proposed multi-scale binary patterns, using exponentially growing circular neighbourhoods and cellular automata, for detecting large-scale texture patterns. Sima et al. in 2018 (40) developed the extended contrast local binary pattern, which included sign, energy, and centre pixel features, and outperformed other methods in texture classification. Li et al. in 2016 introduced the median robust extended local binary pattern, which improved the robustness of LBP to image noise (41). These studies collectively highlight the potential of extended local binary patterns in texture classification.

However, it is well-known that LBP is sensitive to noise and variations in lighting conditions, which can hinder its performance in real-world applications. In this paper, the use of a noise-resistant LBP descriptor aims to fill this research gap.

3. Methodology

The proposed noise-resistant LBP descriptor consists of four steps: (1) preprocessing the image, (2) extracting LBP features,

(3) applying a noise-resistant thresholding technique, and (4) classifying the texture pattern using a machine learning algorithm.

The initial stage involves a critical step known as preprocessing, aimed at improving the overall quality of the image while removing any unwanted aberrations that may be present. These aberrations often lead to undesirable variations or irregularities in pixel values that can obscure the underlying information in the image. To tackle this issue, various filtering techniques are employed, with two common choices being the Gaussian filter and the Median filter. In the specific context of this scenario, a systematic approach is adopted in order to enhance the image.

Initially, the image is converted into grayscale. This conversion simplifies the image by representing it solely in terms of grayscale values, eliminating color information. Grayscale images are often easier to process, and they retain vital information regarding brightness and contrast, which is essential for many image analysis tasks, in this case, extracting the lbp features. It is, therefore, not uncommon to find a lot of the textured datasets used today already converted to grayscale.

After the grayscale conversion, the next step is to employ a median filter as a filtering tool. The median filter is a powerful non-linear filter that plays a pivotal role in noise reduction, particularly for mitigating a type of noise known as "salt-and-pepper noise." This type of noise appears as sporadic, isolated pixels with extreme brightness values, similar to grains of salt and pepper scattered across the image. The median filter operates by swapping out each pixel in the image for its neighbor's median value, within a given kernel size or window.

In place to the median filter, another widely used tool for image preprocessing and noise reduction is the Gaussian filter. Like the median filter, the Gaussian filter is a fundamental component of image processing that contributes to enhancing image quality and making subsequent feature extraction more reliable.

The Gaussian filter, as we know, derives its name from the Gaussian distribution. Thus, this filter works by applying a weighted average to each pixel in the image, taking into account the values of its neighboring pixels within a specified kernel or window. However, unlike the median filter, which uses the median value, the Gaussian filter uses a weighted average that places more emphasis on nearby pixels and less on distant ones.

The effectiveness of median filter over the Gaussian filter lies in its ability to effectively suppress salt-and-pepper noise while preserving the essential structural details of the image. By selecting the median value within the local neighborhood, it ensures that extreme outliers, that is, the noisy pixels are replaced with values that are more representative of the underlying image content. This filtering process helps to smoothen the image, making it more visually pleasing and suitable for subsequent image analysis or recognition tasks. In texture recognition, preserving texture details is essential for achieving high discriminative power. The median filter's noise reduction capabilities (without significant texture loss) contribute to better feature vectors for distinguishing between different texture patterns.

Although the choice between Median and Gaussian filtering for LBP feature extraction isn't universal, when the primary concern is noise reduction while preserving texture information, the Median filter is often the preferred choice for LBP-based texture analysis.

Algorithm 1: Converting digital image to grayscale

```
Input  : resized image-list
Output: grayscale images
1 for $img \in images$ do
2     $gray \leftarrow cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)$;
      $gray\_images.append(gray)$;
3 end
```

Algorithm 2: Using denoising filter - Median Blur

```
Input  : grayscale image-list
Output: median blurred images
1 for $gray \in gray\_images$ do
2     $filtered \leftarrow cv2.medianBlur(gray, 5)$ ;
3     $filtered\_images.append(filtered)$ ;
4 end
```

Algorithm 1 takes a list of color images as input and aims to convert them into grayscale images, which contain only intensity or brightness information, devoid of color data. The algorithm employs the OpenCV library's `cv2.cvtColor` function to perform this conversion for each image in the input list. It iterates through the images, converts them to grayscale using the specified color conversion code (`cv2.COLOR_BGR2GRAY`), and accumulates the resulting grayscale images in a separate list. One can also use the `cv2.IMREAD_GRAYSCALE` method to convert the color images to corresponding grayscale.

Algorithm 3: calculate lbp features

```
Input  : filtered image-list
Output: lbp feature map
1 for $filtered \in filtered\_images$ do
2     $lbp \leftarrow local\_binary\_pattern(filtered, n\_points, radius)$;
3     $lbp\_images.append(lbp)$;
4 end
5
```

Algorithm 2 takes a list of grayscale images and applies a median blur filter to reduce aberrations. It iterates through each image, applies the filter, and stores the smooth versions in a new list. Inside the loop, Line 2 applies the median blur filter to the grayscale image denoted as `gray`. The `cv2.medianBlur` function is called, taking two arguments: the input grayscale image (`gray`) and the kernel size (`5`). The kernel size determines the size of the neighborhood used for median filtering, affecting the extent of noise reduction.

The second step is to extract the LBP features from the preprocessed image. This is typically achieved by

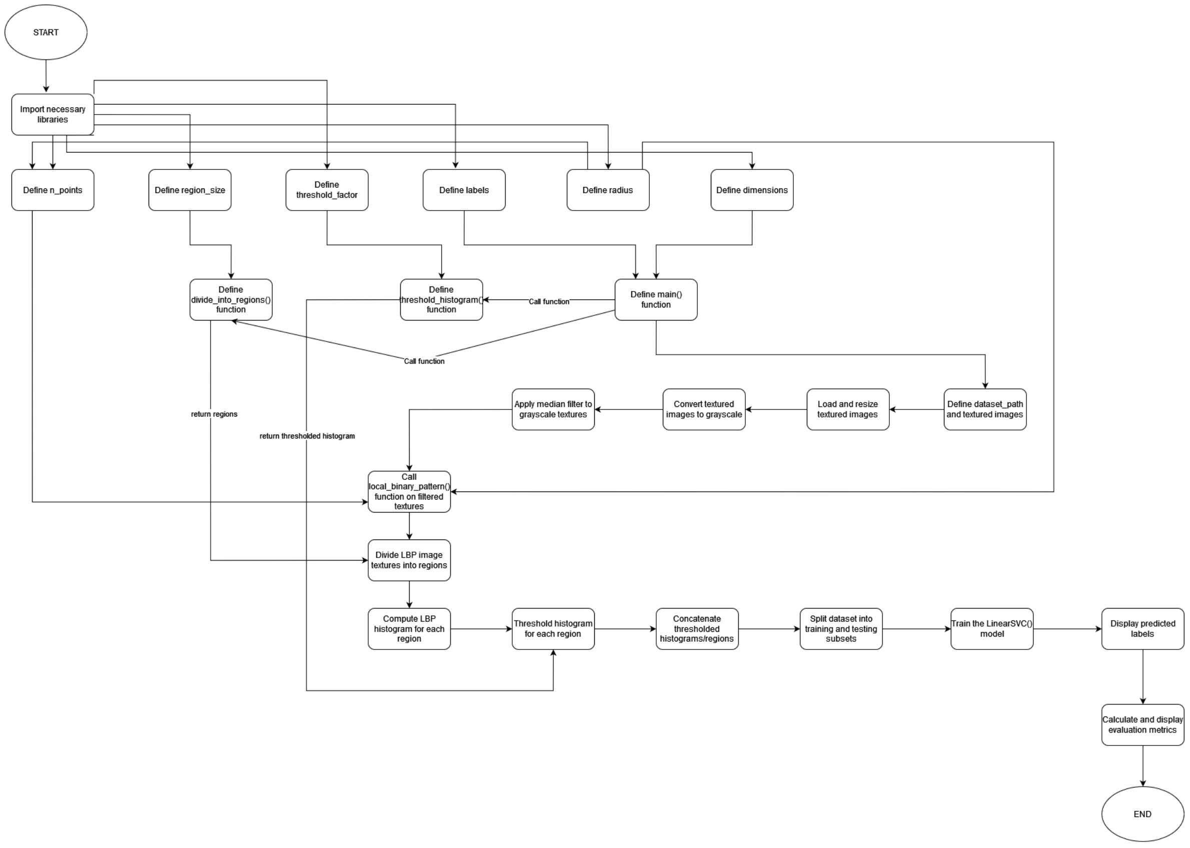

START
Import necessary libraries
Define n_points
Define region_size
Define threshold_factor
Define labels
Define radius
Define dimensions
Define divide_into_regions() function
Define threshold_histogram() function
Call function
Define main() function
Call function
return regions
return thresholded histogram
Apply median filter to grayscale textures
Convert textured images to grayscale
Load and resize textured images
Define dataset_path and textured images
Call local_binary_pattern() function on filtered textures
Divide LBP image textures into regions
Compute LBP histogram for each region
Threshold histogram for each region
Concatenate thresholded histograms/regions
Split dataset into training and testing subsets
Train the LinearSVC() model
Display predicted labels
Calculate and display evaluation metrics
END

Fig. 1: Flow Diagram of Methodology

a. Comparing the intensity of each pixel with its neighboring pixels, and

b. Encoding the result as a 'binary pattern'

This process is repeated for each pixel in the image, and the resulting patterns are used for the classification task, using an SVM classifier. First, the LBP features are computed for each preprocessed image to capture texture information. The scikit-image library is used to achieve this. The method 'local_binary_pattern' is defined as follows: -
@params:

`filtered`: The filtered grayscale image as input.

`n_points`: The number of sampling points to be used for LBP calculation. It is set to `8 * radius` as a common choice.

`radius`: The radius parameter defines the distance from the central pixel to the surrounding sampling points.

`method='uniform'`: This specifies the LBP method to use, with the 'uniform' method being a common choice that simplifies LBP patterns.

The LBP function returns a 2D array, known as the LBP feature map, where each pixel in the map represents the LBP code of the corresponding pixel in the input image. The LBP feature map is added to the `lbp_images` list, which stores these feature maps for all filtered images. Algorithm 3 illustrates the function discussed above.

We now move onto the next step in this research, that is, the region-based thresholding. Here, we take local binary description a step further. Dividing each LBP feature map or, each LBP image into non-overlapping regions using the `divide_into_regions` function defined in Algorithm 4, each region is processed to compute the LBP histogram. The LBP histogram counts the occurrence of each LBP code in the region. The LBP code is a binary code that encodes the local texture information of a pixel and its neighbors in a circular neighborhood.

To achieve this, the code calls the `divide_into_regions` function described in Algorithm 4, for each LBP feature map. The `divide_into_regions` function:

- Accepts the LBP feature map (`lbp`) and a specified region size (`region_size`) as input.
- Divides the LBP feature map into smaller regions of the specified size without overlap.
- Returns a list of these regions.

For each region within the list returned by `divide_into_regions`, the code calculates a histogram of LBP values. These histograms are represented as arrays. The histograms are then thresholded using the `threshold_histogram` function illustrated in Algorithm 5.

Imagine each region $'x'$ comprising of a number of pixels. Let the intensity of the central pixel be I_c, and the mean of the intensity of the neighbouring pixels be I_{x-1}. Now, instead of simply comparing these two intensity values, we apply a `threshold_factor` T_h and multiply it to I_c. This is based on the notion that the closer the values of I_c and I_{x-1} are, the higher is the probability of that region being noisy.

Algorithm 4: Method *divide_into_regions*

```
Input  : lbp feature map
Output: list of non-overlapping regions
1 $height, width \leftarrow$ _lbp.shape;
2 for $i \in range(0, height, \_region\_size$ do
3     for $j \in range(0, width, \_region\_size$ do
4         $region \leftarrow \_lbp[i : i + \_region\_size, j : j + \_region\_size] regions.append(region)$ end
5     end
6     return $regions$
```

Thus, for I_c values only slightly greater than I_{x-1}, the corresponding bit in the LBP code was assigned to 1, resulting in representation of noise for that region. However now, the intensity value of the central pixel was subjected to a threshold factor lesser than 1, and corresponding bit in the LBP code was assigned to 0, resulting in elimnation of the noise. Figure 2 clearly illustrates this concept.

The new equations in place are as below,

$$(T_h * I_c) - I_{x-1} > 0 \rightarrow assign = 1$$

$$(T_h * I_c) - I_{x-1} < 0 \rightarrow assign = 0$$

This function `threhold_histogram` normalizes the histogram to have a unit L1 norm and applies a threshold to make it binary. As discussed, the threshold is determined by multiplying the specified `threshold_factor` by the mean of the normalized histogram. In other words, values below this threshold are set to 0, and values equal to or above the threshold are set to 1.

Algorithm 5: thresholding histograms

```
Input  : list of non-overlapping histograms
Output: thresholded histograms
1 $threshold \leftarrow$_threshold_factor * np.mean(_histogram) $\_histogram[\_histogram < threshold] \leftarrow 0$;
2 $\_histogram[\_histogram >= threshold] \leftarrow 1$;
3 return $\_histogram$
```

The primary purpose of normalizing a histogram is to ensure that its values represent relative frequencies or probabilities, making it more interpretable and invariant to the scale of the data. In this specific context, the function is used to normalize the LBP histogram for a region of an image. Normalization, being a common practice in image processing and feature extraction tasks, enhances the robustness and comparability of features, ensuring that the LBP histograms are transformed into normalized representations that are both scale-invariant and more suitable for subsequent texture analysis and classification.

After thresholding each region's histogram, the binary histograms are collected in a list to form a thresholded representation of the entire LBP feature map. The thresholded histograms of LBP values for each region are then concatenated to create a single feature vector representing the entire image. Algorithm 6 summarizes the above.

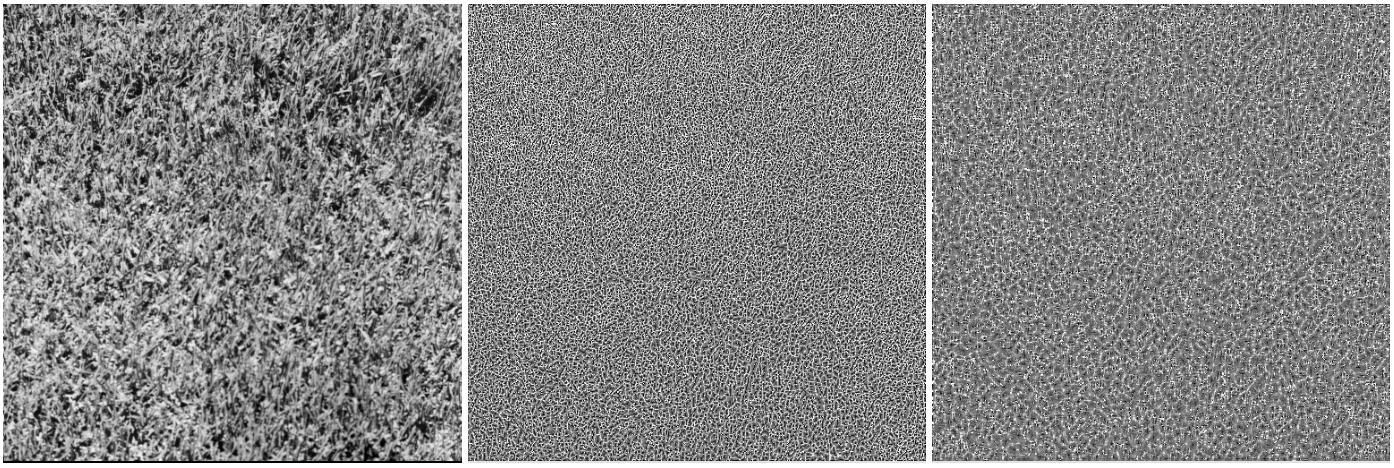
Algorithm 6: fetching thresholded images

```
Input  : thresholded histograms
Output: thresholded images
1  for $lbp \in lbp\_images$ do
2      $regions \leftarrow divide\_into\_regions(lbp, region\_size);$
3      for $region \in regions$ do
4          $histogram, \_ \leftarrow np.histogram(region, bins = np.arange(0, 10), density = True);$
5          $histogram \leftarrow threshold\_histogram(histogram,$
6          $threshold\_factor)$
7          $thresholded\_regions.append(histogram);$
8      end
9      $thresholded\_image \leftarrow np.concatenate(thresholded\_regions);$
10     $thresholded\_images.append(thresholded\_image);$
11 end
12 $thresholded\_images \leftarrow np.array(thresholded\_images);$
```

For each LBP feature map (`lbp`), the code divides it into non-overlapping regions using the `divide_into_regions` function. The regions are defined by the `region_size` parameter. This step aims to segment the LBP feature map into smaller areas to capture local texture patterns. Within the loop for each LBP feature map, there is an inner loop that iterates over the regions obtained in the previous step. Each `region` variable represents a specific segment of the LBP feature map.

Inside the inner loop, a histogram of LBP values is computed for the current `region`. This histogram is created using the `np.histogram` function. The `np.histogram` function takes the `region` as input and specifies the number of bins using `bins=np.arange(0, 10)`, which creates 10 equally spaced bins. The `density=True` parameter normalizes the histogram, ensuring that it represents relative frequencies. After calculating the histogram for the current region, it is passed to the `threshold_histogram` function for thresholding. As discussed, the `threshold_histogram` function normalizes the histogram and applies a threshold based on the `threshold_factor`.The histogram's values that fall below the threshold are set to zero, values that meet or surpass the threshold, however, are set to 1, as we already discussed.

The thresholded histograms for all regions within the current LBP feature map are collected in a list called `thresholded_regions`. After processing all regions within the current LBP feature map, the code concatenates the thresholded histograms from all regions into a single 1D array called `thresholded_image`. This array represents the thresholded LBP feature map for that image. The `thresholded_image` is appended to the `thresholded_images` list, which accumulates the thresholded representations of LBP feature maps for all images in the dataset. After processing all LBP feature maps in the outer loop, the `thresholded_images` list contains the thresholded representations of LBP feature maps for all images in the dataset. Finally, the code converts `thresholded_images` into a NumPy array using



(a) Original texture image (b) Corresponding LBP image (c) Thresholded LBP image

Fig. 2: An illustration of region-based thresholding

`np.array(thresholded_images)` to facilitate classification.

The next set of tasks revolve around training and testing the model. We use two different types of datasets for this purpose, to cater to two different use cases. For the material science use case, we employ the Brodatz textured dataset, and for medical imaging, we use the OASIS Brain MRI database. In case of the Brodatz dataset, each image file has a filename that includes information about its category or class. Labels are assigned to the images and the assigned labels are stored in an array called `labels`. The directory structure for the OASIS MRI database is different. There are 6 sub-directories, the first four containing the images from four different classes. There are 2 sub-directories containing a mix of images from all four classes, specifically for testing purposes.

Once the Brodatz images are labeled, it's crucial to split the dataset into training and testing sets. This separation is vital for assessing how well the trained model generalizes to unseen data and avoids over-fitting (where a model memorizes the training data but performs poorly on new, unseen examples).

```
X_train, X_test, y_train, y_test =
train_test_split(thresholded_images,
labels, test_size=0.15)
model = LinearSVC()
model.fit(X_train, y_train)
```

In the provided code, the dataset is split into two subsets using the `train_test_split` function from scikit-learn. By convention, one portion is designated as the training set, while the other serves as the testing set. The machine learning model is trained with the training set, allowing it to learn the relationships between the extracted texture features and their corresponding labels. The model's performance on unseen data is assessed using the testing set. This step simulates real-world scenarios where the model encounters new textures it hasn't seen during training.

For the OASIS MRI database, the images are already labelled. We loop through the entire directory, storing the extracted lbp features in a list named `features`, and their corresponding labels (categories) in another list called `labels`. The following code illustrates the above,

```
for category in range(1, 5):
    category_path =
    os.path.join(texture_directory,
    f'OASIS_Cross_{category}_converted')
    for filename in os.listdir(category_path):
        if filename.endswith('.jpg'):
            image_path = os.path.join(category_
            path, filename)
            feature = extract_features(image_
            path)
            features.append(feature)
            labels.append(category)
```

Next step is to train the model, in this case, using a Linear Support Vector Machine. Let us see why a LinearSVC is specifically employed over any other classifying algorithm, for this texture recognition task.

LinearSVCs are well-suited for problems where the decision boundary is linear, or can be effectively approximated as linear. In texture recognition, LBP features often capture linearly separable patterns within images. Considering a classification task where the goal is to distinguish between "wood" and "brick" textures, or between "Cross 1" MRI and "Cross 2" MRI, LinearSVC can effectively learn a linear decision boundary between these two classes if they exhibit linearly separable characteristics in the LBP feature space.

Now, making predictions on the testing data with `model.predict`, the model's accuracy, precision, recall, and F1-score is calculated to evaluate performance, using scikit-learn's metrics functions. The discussion is continued in the next section.

4. Results

So far, we have looked at the motivation, idea, and methodology behind the noise-resistant LBP-based texture recognition model, involving the following key steps:

1. Feature Extraction: Local Binary Pattern (LBP) descriptors were computed for each image to capture the texture patterns.
2. Model Training: A Linear Support Vector Machine (LinearSVC) classifier was trained on a subset of the dataset. And finally,
3. Model Evaluation: The trained model was evaluated on the remaining subset of the Brodatz dataset to measure its classification performance.

In this section, let us look at the results of the texture recognition experiments using Local Binary Pattern (LBP) descriptors and Linear Support Vector Machine (LinearSVC) classification. This study aimed to assess the effectiveness of this approach in accurately classifying different texture patterns. The Brodatz dataset used in the training process and experiments comprises a diverse collection of texture images, each belonging to a specific texture class. The textures have descriptions that include ”Brodatz - Straw (D15),” ”Brodatz - Plastic bubbles (D112),” etc., measure 512x512 in pixels, and 256 KB in size. Figure 3 illustrates a couple of examples of textures. The dataset was preprocessed to ensure uniform dimensions and quality across all images.

Over a number of subsequent runs on the Brodatz dataset, it was observed that the model achieved an average accuracy score of 80.00%. In other words, it correctly predicted the true labels of 80% of texture images from the test subset. This indicates the effectiveness of our approach in noise-resistant texture classification tasks.

On the other hand, the complex OASIS MRI database consists of four sets or categories of brain MRI images, namely Cross 1, Cross 2, Cross 3, Cross 4. The intricate structure of the dataset makes it a little more difficult to apply our method on it. Examples of the MRI images are shown in Figure 4.

We tried running the model with a different `threshold_factor` each time, and logged the respective performance metrics. These consist of F1-score, recall, accuracy, and precision. These measurements offer comprehensive insights into the model's ability to classify textures accurately, minimize false positives, avoid false negatives, and achieve a balanced performance.

Table I displays the results of all 10 runs of the model. It can be clearly observed that the model achieves the highest accuracy of 72.62% at `threshold_factor=0.3`. This indicates optimum threshold for the classification task. Without the threshold, that is, for `threshold_factor=1.0` the stood at 57.14%. The Precision metric recorded for `threshold_factor=0.3` was 79.43%, and Recall score was found to be 72.62% as well. The F1 Score for this threshold was logged at 71.14%, indicates a balanced trade-off between Precision and Recall, often desirable in such classification and recognition tasks.

5. Conclusion

In this study, we discussed a possible, robust solution for texture recognition, a fundamental practice in computer vision with applications spanning diverse domains, from image analysis to industrial automation. This research focused on harnessing the power of Local Binary Pattern (LBP) descriptors in conjunction with Linear Support Vector Machine (LinearSVC) classification to achieve accurate and robust texture recognition. In this section, we reflect on the idea, the methodologies explored, the insights gained, and the implications of the findings.

At the outset of the research, the primary objectives were clear: to develop an effective texture recognition system and to evaluate its performance rigorously. The exploration began with the foundation of feature extraction through LBP descriptors. LBP, a widely recognized and versatile texture analysis technique, was employed to capture intricate texture patterns within images. We discussed the theory behind LBP, its robustness to various transformations, and its ability to encode spatial relationships among pixel intensities effectively. Emphasizing on the adaptability of LBP, we looked at its ability to accommodate variations in texture scale, rotation, and illumination. This adaptability is a critical asset in real-world applications where textures may exhibit substantial variations.

With feature extraction in place, we turned our attention to classification, a pivotal component of any texture recognition system. Linear Support Vector Machine (LinearSVC) emerged as the classifier of choice, elucidating the reasons behind this selection. The linear nature of LinearSVC aligns harmoniously with the discriminative power of LBP descriptors, especially in scenarios where texture patterns exhibit linearly separable characteristics. We discussed LinearSVC's computational efficiency, robustness to high-dimensional feature spaces, and regularization capabilities as attributes that significantly contribute to its suitability for the texture recognition task.

To validate the effectiveness of this approach, a series of meticulously designed experiments were conducted. A couple of dataset comprising various texture classes from different fields of work were obtained, each item representing distinct visual patterns. The datasets were standardized through preprocessing steps that reduced aberrations. Experiments involved comprehensive data splitting into training and testing sets, a crucial step in assessing model generalization. The models were rigorously trained on a subset of the data and subsequently evaluated on an independent testing dataset.

In the pursuit of a comprehensive evaluation, a suite of performance metrics were employed, especially for the OASIS MRI database, each designed to shed light on different facets of the model's capabilities. Accuracy provided an overarching measure of model's correctness in predictions. Precision, recall, and the F1-score complemented accuracy, offering insights into the model's ability to make accurate positive predictions, correctly identify relevant instances, and balance precision and recall in scenarios with class imbalances. These metrics provide a holistic perspective on the model's performance.

Experimental results revealed compelling insights into the efficacy of the LBP-based texture recognition model. The table showcasing model performance metrics for multiple runs of the model over the brain MRI dataset underscored the reliability of the approach. Our discussions delved into the nuances of these results, elucidating their implications in various contexts.

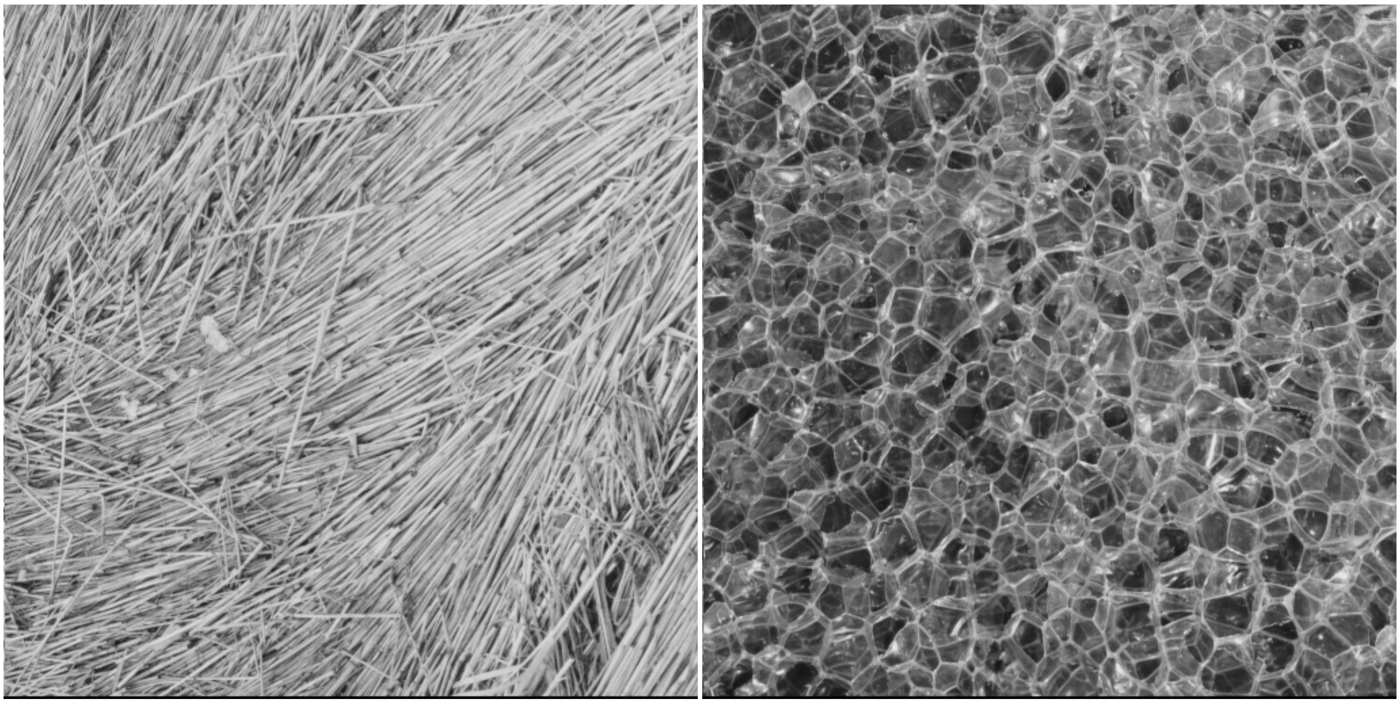

(a) Brodatz - Straw (D15)
(b) Brodatz - Plastic bubbles (D112)

Fig. 3: Different Brodatz textures used

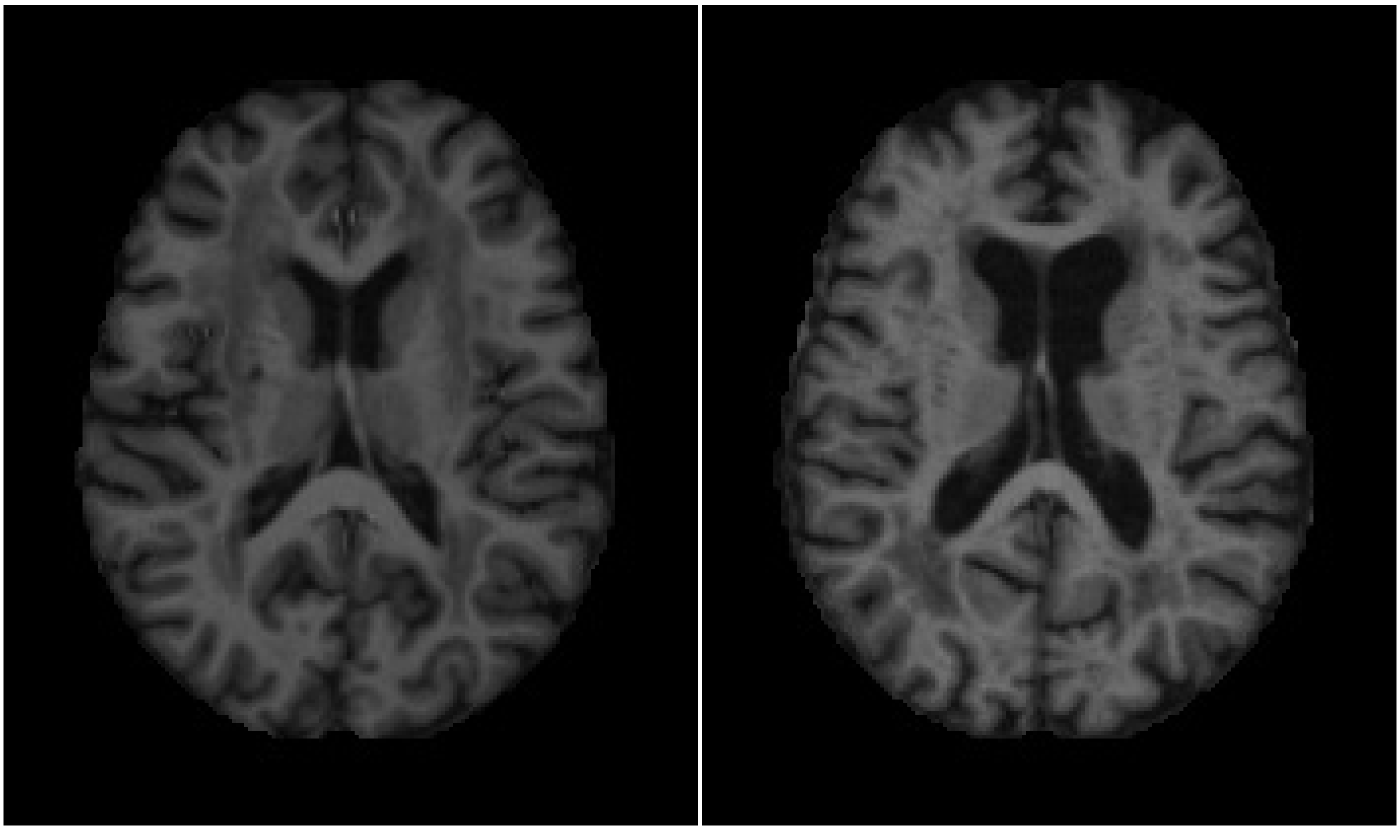

(a) Cross 2 MRI Image
(b) Cross 4 MRI Image

Fig. 4: OASIS MRI images used

SNo	Threshold factor	Accuracy	Precision	Recall	F1 Score
1	0.1	38.10	40.87	38.10	36.54
2	0.2	40.48	40.01	40.48	37.00
3	0.3	72.62	79.43	72.62	71.14
4	0.4	51.19	58.37	51.20	48.72
5	0.5	53.57	56.10	53.58	53.82
6	0.6	60.71	61.99	60.72	60.83
7	0.7	63.10	63.13	63.10	62.99
8	0.8	55.95	56.55	55.96	55.94
9	0.9	53.57	55.27	53.58	53.07
10	1.0	57.14	60.07	57.15	54.04

Table 1: Model Performance Metrics for 10 Runs

We looked at the model's accuracy at the optimum threshold, as well as without the threshold, which demonstrated its proficiency in distinguishing between diverse texture classes when noise-resistant thresholding was specifically applied. Additionally, the precision, recall, and F1-score values were gathered and a table was compiled, emphasizing their significance in applications where minimizing false positives or false negatives holds importance.

While this research has yielded promising results, the journeyis far from complete. Several avenues for future exploration beckon:

Deep Learning Integration: The integration of deep learning techniques, such as Convolutional Neural Networks (CNNs), could enhance the LBP-based texture recognition model's performance, especially when dealing with complex and diverse texture patterns.
Semantic Segmentation: Extending this approach to semantic segmentation, where texture patterns within images are delineated and classified, opens doors to applications in object detection and scene understanding.
Transfer Learning: Leveraging transfer learning by pretraining on large-scale textured image datasets can boost the Local Binary Pattern Descriptor's ability to recognize textures with limited training data.
Multimodal Data Fusion: Combining texture information with other modalities, such as color or depth data, can further enhance the discriminative power of this system.

This research underscores the significance of feature extraction techniques like LBP in encoding intricate texture information and the importance of robust classification algorithms like LinearSVC in making accurate predictions. As we look to the future, the integration of advanced techniques and the pursuit of interdisciplinary collaborations hold the promise of advancing LBP-centered texture recognition further and broadening its impact across multiple domains.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[2] G. Zhao and M. Pietikainen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 915–928, 2007.

[3] H. Roy and D. Bhattacharjee, "Local-gravity-face (lg-face) for illumination-invariant and heterogeneous face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 7, pp. 1412–1424, 2016.

[4] H. Roy, D. Bhattacharjee, and O. Krejcar, "Interpretable local frequency binary pattern (lfrbp) based joint continual learning network for heterogeneous face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2125–2136, 2022.

[5] S. Koley, H. Roy, S. Dhar, and D. Bhattacharjee, "Cross-modal face recognition with illumination-invariant local discrete cosine transform binary pattern (ldctbp)," *Pattern Analysis and Applications*, vol. 26, no. 3, pp. 847–859, 2023.

[6] J. Lu, V. E. Liong, and J. Zhou, "Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1979–1993, 2018.

[7] B. Zhang, Y. Gao, S. Zhao, and J. Liu, "Local derivative pattern versus local binary pattern: Face recognition with high-order local pattern descriptor," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 533–544, 2010.

[8] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[9] D. Bhattacharjee and H. Roy, "Pattern of local gravitational force (plgf): A novel local image descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 595–607, 2021.

[10] Q. Zheng, A. Kumar, and G. Pan, "A 3d feature descriptor recovered from a single 2d palmprint image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 6, pp. 1272–1279, 2016.

[11] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikäinen, X. Chen, and W. Gao, "Wld: A robust local image descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1705–1720, 2010.

[12] S. Murala, R. P. Maheshwari, and R. Balasubramanian, "Local tetra patterns: A new feature descriptor for content-based image retrieval," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2874–2886, 2012.

[13] H. Roy and D. Bhattacharjee, "A novel quaternary pattern of local maximum quotient for heterogeneous face recognition," *Pattern Recognition Letters*, vol. 113, pp. 19–28, 2018.

[14] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE transactions on image processing*, vol. 19, no. 6, pp. 1635–1650, 2010.

[15] S. Chakraborty, S. K. Singh, and P. Chakraborty, "Local gradient hexa pattern: A descriptor for face recognition and retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 1, pp. 171–180, 2018.

[16] Z. Lei, M. Pietikäinen, and S. Z. Li, "Learning discriminant face descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 2, pp. 289–302, 2014.

[17] S. Koley, H. Roy, and D. Bhattacharjee, "Gammadion binary pattern of shearlet coefficients (gbpsc): An illumination-invariant heterogeneous

face descriptor," *Pattern Recognition Letters*, vol. 145, pp. 30–36, 2021.

[18] J. Lu, V. E. Liong, X. Zhou, and J. Zhou, "Learning compact binary face descriptor for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2041–2056, 2015.

[19] S. Bhattacharya, G. S. Nainala, S. Rooj, and A. Routray, "Local force pattern (lfp): Descriptor for heterogeneous face recognition," *Pattern Recogn. Lett.*, vol. 125, no. C, p. 63–70, jul 2019.

[20] H. Roy and S. Koley, "Local-friis-radiation-pattern (lfrp) for face recognition," *Sensing and Imaging*, vol. 22, 2021.

[21] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1224–1244, 2018.

[22] R. He, X. Wu, Z. Sun, and T. Tan, "Wasserstein cnn: Learning invariant features for nir-vis face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1761–1773, 2018.

[23] X. Liu, L. Song, X. Wu, and T. Tan, "Transferring deep representation for nir-vis heterogeneous face recognition," in *2016 international conference on biometrics (ICB)*. IEEE, 2016, pp. 1–8.

[24] L. Guo, H. Bai, and Y. Zhao, "A lightweight and robust face recognition network on noisy condition," *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1964–1969, 2019.

[25] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *IEEE transactions on information forensics and security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[26] Shahabaz, D. K. Somwanshi, A. K. Yadav, and R. Roy, "Medical images texture analysis: A review," in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, 2017, pp. 436–441.

[27] P. S. Purkait, N. Ghosh, S. Dey, H. Roy, and S. Dhar, "A comprehensive approach to classify the skin cancer disease using latest cnn model (yolov8)," in *Doctoral Symposium on Intelligence Enabled Research*. Springer, 2023, pp. 159–169.

[28] C. Kaur and U. Garg, "Artificial intelligence techniques for cancer detection in medical image processing: A review," *Materials Today: Proceedings*, vol. 81, pp. 806–809, 2023, international Virtual Conference on Sustainable Materials (IVCSM-2k20).

[29] 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.

[30] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 971–987, 2002.

[31] R. Maani, S. Kalra, and Y.-H. Yang, "Rotation invariant local frequency descriptors for texture classification," *IEEE Transactions on Image Processing*, vol. 22, no. 6, pp. 2409–2419, 2013.

[32] 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.

[33] M. Topi, O. Timo, P. Matti, and S. Maricor, "Robust texture classification by subsets of local binary patterns," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 3, 2000, pp. 935–938 vol.3.

[34] M. Topi, P. Matti, and O. Timo, "Texture classification by multi-predicate local binary pattern operators," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 3, 2000, pp. 939–942 vol.3.

[35] S. Banerji, A. Verma, and C. Liu, "Novel color lbp descriptors for scene and image texture classification," in *15th international conference on image processing, computer vision, and pattern recognition, Las Vegas, Nevada*, 2011, pp. 537–543.

[36] L. Nanni, A. Lumini, and S. Brahnam, "Survey on lbp based texture descriptors for image classification," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3634–3641, 2012.

[37] N. Kazak and M. Koc, "Some variants of spiral lbp in texture recognition," *IET Image Processing*, vol. 12, no. 8, pp. 1388–1393, 2018.

[38] W.-H. Liao and T.-J. Young, "Texture classification using uniform extended local ternary patterns," *2010 IEEE International Symposium on Multimedia*, pp. 191–195, 2010.

[39] T. Mäenpää and M. Pietikäinen, "Multi-scale binary patterns for texture analysis," in *Scandinavian Conference on Image Analysis*, 2003.

[40] J. Sima, Y. Dong, T. Wang, L. Zheng, and J. Pu, "Extended contrast local binary pattern for texture classification," 2018.

[41] L. Liu, S. Lao, P. W. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Median robust extended local binary pattern for texture classification," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1368–1381, 2016.