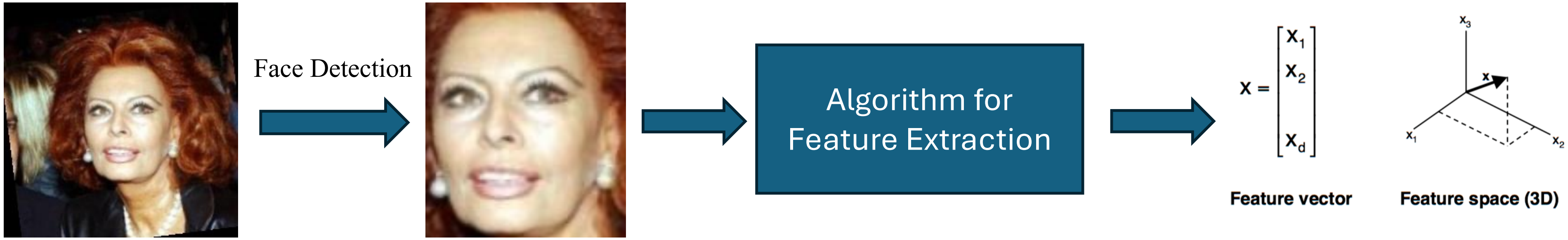
Face Classification

2° Assignment

Face Description

- Faces are one of many forms of biometrics used to identify individuals and to verify their identity.
- Feature extraction is a very important step in face verification.
- Different feature extraction techniques like Principal Component Analysis (PCA), Fisher Linear Discriminant analysis (FLD) and Local Binary Patterns are generally used.



Face Detection

Algorithm for Feature Extraction

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \\ x_d \end{bmatrix}$$

Feature vector

Feature space (3D)

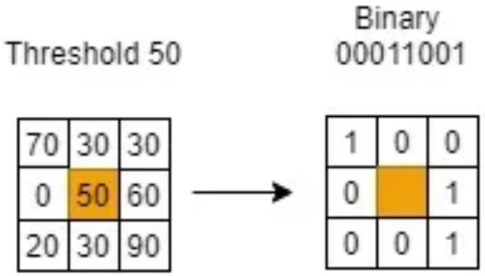
Local Binary Patterns

- The **local binary pattern (LBP)** is a popular texture descriptors based on appearance features.
- It describes **local structures of an image** and is invariant to changes in illumination. LBP was first introduced in 1994 (*) and has been used in a wide range of applications, especially face detection and recognition
- LBP is:
  - robust to illumination variations (it can capture texture information in images that have different lighting conditions)
  - computationally efficient method
  - invariant to image rotation and scale (in uniform modality).
  - highly discriminative for texture analysis
- However, LBP:
  - is sensitive to noise in the image.
  - captures local texture information in the immediate vicinity of each pixel
  - cannot capture rotational information (cannot distinguish between textures that differ in their rotational patterns)
- LBP is applied to **grayscale images** and does not capture color information in the texture patterns.

*T. Ojala, M. Pietikäinen, and D. Harwood (1994), "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions", Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582–585.

Local Binary Patterns – default modality

- LBP **introduces a re-encoding techniques** in local neighborhood. It compares the intensity of a central pixel in a small neighborhood with the intensity of its surrounding pixels.
- Each pixel in the neighborhood is assigned a binary value based on whether its intensity is greater than or less than the intensity of the central pixel (threshold). These binary values are then concatenated into a binary number and transformed to a decimal value.
- Generally, a **histogram of the values** is used to capture the texture distribution within an image.

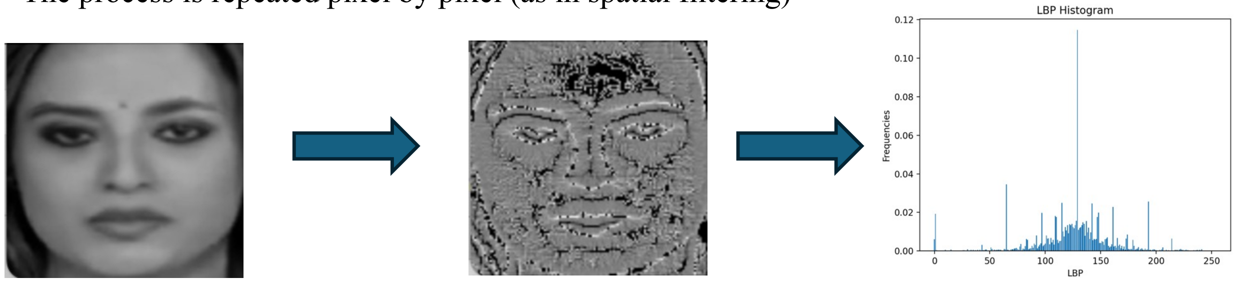


70	30	30
0	50	60
20	30	90

1	0	0
0		1
0	0	1

To the central pixel (50) it corresponds the value 25 (in binary 00011001)

The process is repeated pixel by pixel (as in spatial filtering)



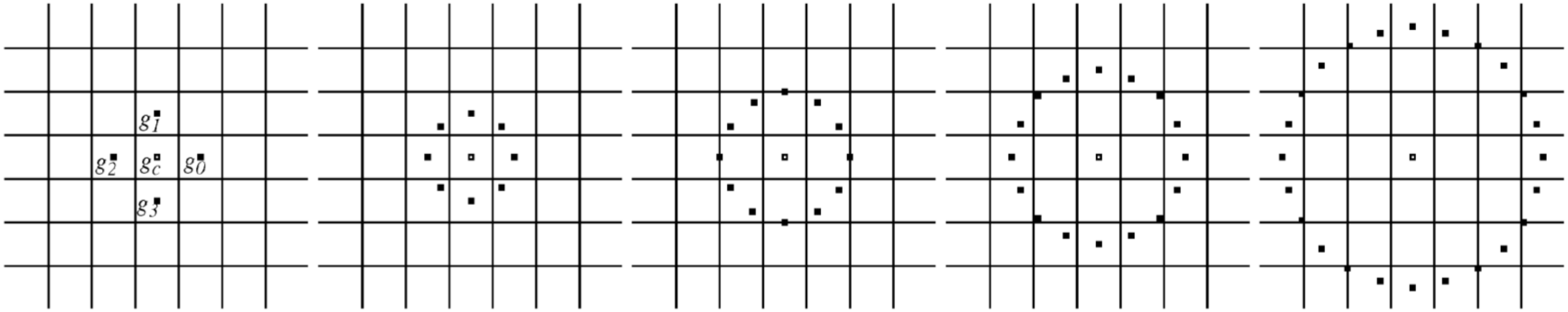
Grayscale image

LBP image

Feature descriptor (histogram)

Local Binary Patterns – Uniform modality

- Traditional LBP is not invariant to rotation and the computation is linked with the neighborhood grid.
- An improved way to compute LBP is to look at P points surrounding a central point distributed over a circle of radius R. The P points do not fall inside the pixels and interpolation of the gray levels is used to find their values. Then, **the number of changes 0-1 in the P points is determined**.

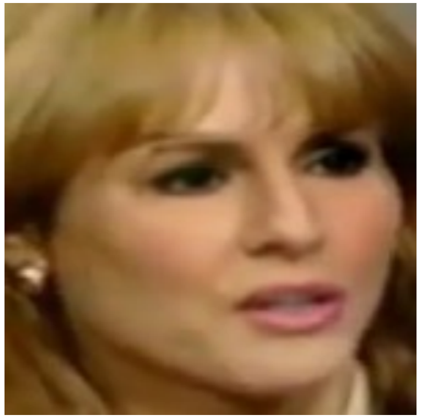


(*P*=4,*R*=1.0) (*P*=8,*R*=1.0) (*P*=12,*R*=1.5) (*P*=16,*R*=2.0) (*P*=24,*R*=3.0)

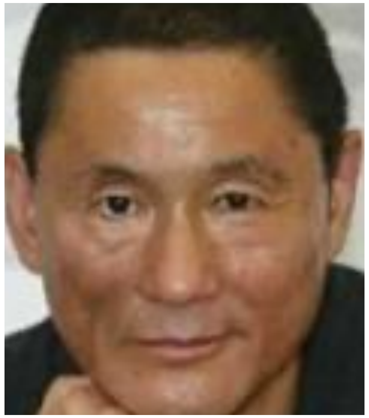
T. Ojala, M. Pietikainen, T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, July 2002 DOI:10.1109/TPAMI.2002.1017623

Face Forensics and DeepFake

- The generation and manipulation of synthetic images are recently making enormous progress and such progress raises significant concerns, especially in terms of security, loss of trust in digital content and the possible spread of false information or news.
- Face forensics is the study of techniques aimed at detecting synthetic faces and validating the veracity of a face (i.e., detecting the absence of facial manipulation). It is related to face verification.



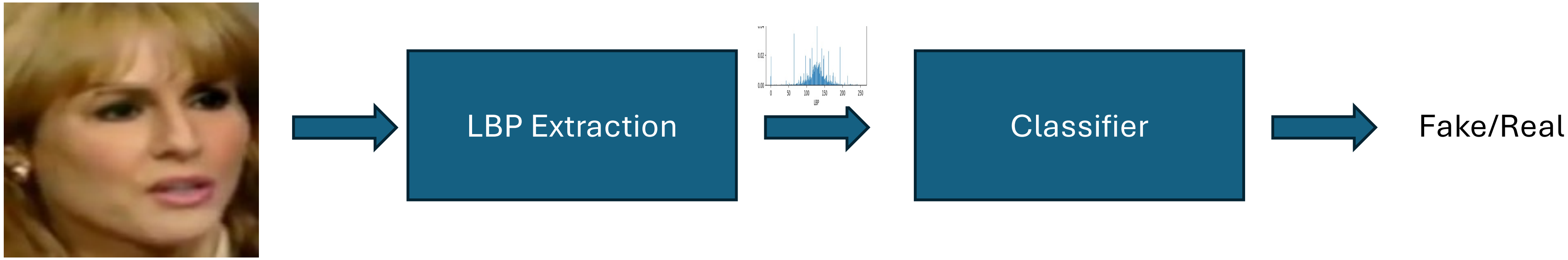
Fake or Real?



Fake or Real?

Assignment 2: Fake Face detection

- The goal is that of training, validating and testing a classifier to detect if a face is fake or real



LBP Extraction

Classifier

Fake/Real

Fake Face Detection - sketch

1. You will prepare a dataset based on publicly available images
2. For each uncropped image, **detect the face region** (if more than a face is detected, discard the image –> dataset cleaning)

   from openCV use: the faceDetector (*cv2.CascadeClassifier*)
3. use scikit-image to compute the **LBP image** and compute an histogram (**use numpy for that**)

   skimage.feature.local_binary_pattern()
4. Since extracting features from all images can take some time (1 hour on my pc), I suggest to use *pickle* to save the list of feature vectors you get from previous steps
5. You will adopt a ***fixed subjects-based partition experimental protocol*** by considering 60% of the ***subjects*** *(not images!! careful on that!)* to prepare the training set, 20% of the subjects as validation set and 20% as test set
6. You will train **several classifiers** from the scikit-learn library on the training set and use the validation set to choose the best classifier and data pre-processing (model selection)
7. Finally, you will test the selected classifier on the test and report the results

All experiments must be commented and reported in slides to be presented during the exam!

Dataset

- For the fake faces, you may wanna use a subset of the Faceforensycs dataset
- The dataset has images of 998 subjects, with already cropped fake faces

https://www.kaggle.com/datasets/greatgamedota/faceforensics?resource=download

- For the real faces, you may wanna use the Faces in the Wild dataset, also available from Kaggle. On this set, you must use the opencv facedetector to crop images in the same way!
- https://www.kaggle.com/datasets/jessicali9530/lfw-dataset/data

- Remember to assign labels 1 to fake faces and 0 to real faces

Model selection

- The validation set must be used to **select the best features and classifier** for solving our problem
- **The function to extract LBP takes several parameters in input**. You need to select the best method (default, uniform) with the following parameter configurations:
    - method = default, P=256, R=1 (original implementation)
    - method = uniform, P=8, R=1
- You will have to validate the effect of standardization on data (use StandardScaler). Of course, **mean and covariance must be computed only on the training set** and applyed on the validation and test set.
- You will have to choose between: RandomForestClassifier, LogisticRegression and LinearSVC (use defaul parameters, training of the models was pretty fast on my computer...)

- You can select the best model and features by considering the accuracy values in classification

(N° of expected experiments = 12)

I suggest parameterizing your code to improve readability and make it easier to run experiments with different configurations.

Model Selection

- After doing experiments on the validation set, you have to prepare a table showing the accuracy values that your methods have achieved with different features, pre-processing and classifiers (RandomForest=RF, LogisticRegression=LR...)

Features	Standardization	acc. RF	acc. LR	acc. linearSVC
LBP - default	NO			
LBP - uniform	NO			
LBP - default	YES			
LBP - uniform	YES			

- Then, you will have to comment the results and select the model you think works better on the validation set

Test

- Once you have selected a model (and features + data preprocessing) you will have to use the selected trained model on your test data and estimate the accuracy value and the confusion matrix.
- Then, comment your results

Discussion

- Finally, discuss your work by highlighting pros and cons of the approach and of the experiments you have performed
- What could have been improved?
- If you had more time, what would you have liked to change???

- Remember to include the list of *materials* **you studied** to develop the assignment.
- Remember that, **during the exam**, I will ask you to explain to me ***each line of your code***!!