**MỤC LỤC**

[**MỤC LỤC** 1](#_Toc102309033)

[**CHƯƠNG 1: TỔNG QUAN NGHIÊN CỨU** 2](#_Toc102309034)

[**1.1.** **Thuật toán trích chọn đặc trưng** 2](#_Toc102309035)

[**1.2.** **Texture feature (đặc trưng kết cấu)** 2](#_Toc102309036)

[**1.3.** **GLCM** 2](#_Toc102309037)

[1.3.1. Lịch sử phát triển 2](#_Toc102309038)

[1.3.2. Chi tiết về GLCM 2](#_Toc102309039)

[1.3.3. Ý nghĩa 2](#_Toc102309040)

[1.3.4. Ưu, nhược điểm của thuật toán GLCM 2](#_Toc102309041)

[**1.4.** **Bài toán phân loại trong học máy** 2](#_Toc102309042)

[**1.5.** **Công cụ sử dụng** 3](#_Toc102309043)

[1.5.1. VS Code 3](#_Toc102309044)

[1.5.2. Jupyter Notebook 3](#_Toc102309045)

[**CHƯƠNG 2: DEMO THUẬT TOÁN VÀ BÀI TOÁN PHÂN LOẠI HÌNH ẢNH** 4](#_Toc102309046)

[**2.1. Demo thuật toán GLCM** 4](#_Toc102309047)

[**2.2. Bài toán phân loại hình ảnh** 7](#_Toc102309048)

# **CHƯƠNG 1: TỔNG QUAN NGHIÊN CỨU**

## **Thuật toán trích chọn đặc trưng**

* Trích chọn đặc trưng là một trong những thuật toán cốt lõi trong các bài toán về nhận diện (recognition), phát hiện (detection), phân loại (classification), phân cụm (clustering),… trong học máy. Đây thực chất là một quá trình chọn lọc một tập con chứa các thuộc tính liên quan để sử dụng trong quá trình xây dựng mô hình. Các kỹ thuật trích chọn đặc trưng được dùng cho một số lý do:
* Đơn giản hóa các mô hình để giúp các nhà nghiên cứu/người dùng diễn dịch dễ dàng hơn
* Giảm thời gian huấn luyện
* Giảm chiều dữ liệu
* Tăng cường tổng quát hóa bằng cách giảm sự quá khớp (overfitting)

Các đặc trưng thường gặp trong các bài toán học máy có thể được chia làm 4 nhóm chính, bao gồm: Geometric features (các đặc trưng hình học), Statistical features (các đặc trưng thống kê), Texture features (các đặc trưng kết cấu) ,and Color features (các đặc trưng màu sắc

## **Texture feature (đặc trưng kết cấu)**

## **GLCM**

### Lịch sử phát triển

### Chi tiết về GLCM

### Ý nghĩa

### Ưu, nhược điểm của thuật toán GLCM

## **Bài toán phân loại trong học máy**

Đây là một trong những bài toán điển hình của học có giám sát (Supervised Learning). Một bài toán được gọi là phân loại (Classification) nếu các nhãn (label) của dữ liệu đầu vào được chia thành một số hữu hạn nhóm. Bộ lọc thư rác là ví dụ tiêu biểu cho tác vụ này: nó được huấn luyện với nhiều mẫu thư điện tử cùng với nhãn tương ứng (thư rác hoặc thư thông thường), từ đó học cách phân loại các thư điện tử mới.

***Ví dụ:*** Gmail xác định xem một email có phải là spam hay không; các hãng tín dụng xác định xem một khách hàng có khả năng thanh toán nợ hay không. Ba ví dụ phía trên được chia vào loại này

## **Công cụ sử dụng**

### VS Code

### Jupyter Notebook

# **CHƯƠNG 2: DEMO THUẬT TOÁN VÀ BÀI TOÁN PHÂN LOẠI HÌNH ẢNH**

## **Demo thuật toán GLCM**

1. ***Tạo ma trận đồng xuất hiện mức xám***

* Nhập hình ảnh
* Chuyển đổi hình ảnh thành ảnh đa mức xám
* Sử dụng hàm ***graycomatrix*** trong thư viện ***skimage.feature*** của python để tạo ma trận glcm:

(function) graycomatrix: (image: Any, distances: Any, angles: Any, levels: Any | None = None, symmetric: bool = False, normed: bool = False) -> (Any | NDArray[bool\_] | ndarray)

Trong đó:

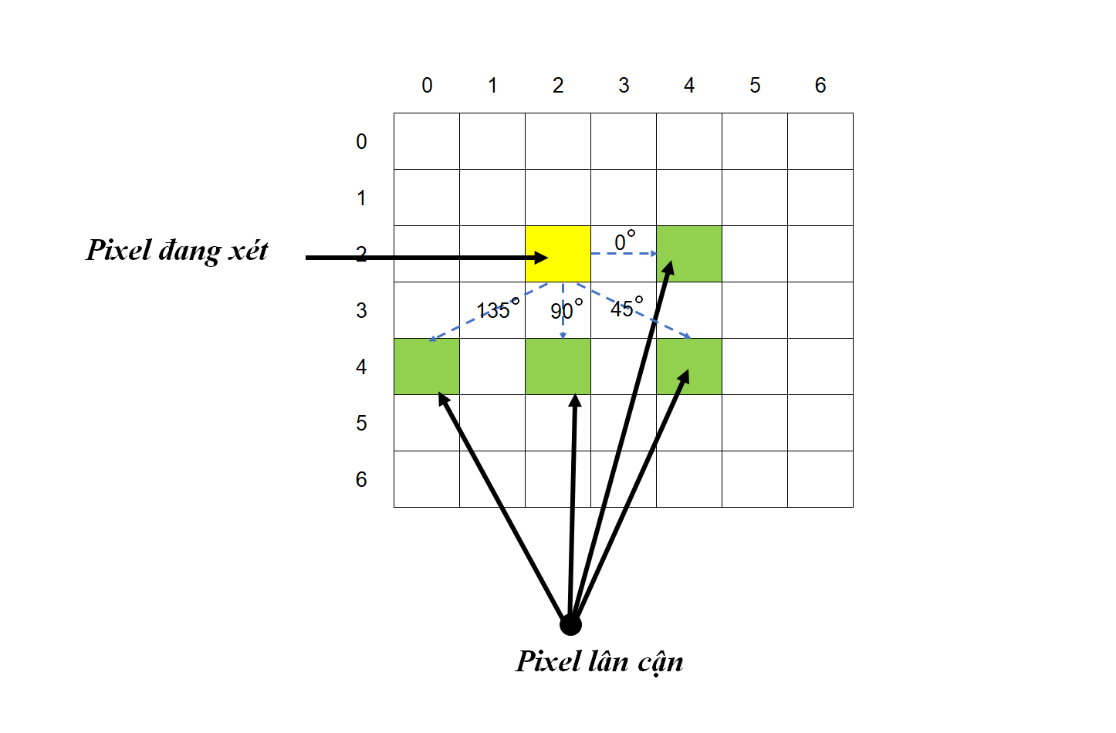
***image***: dữ liệu hình ảnh đầu vào (ảnh đa mức xám)

***distances***: khoảng cách giữa pixel đang xét với các pixel lân cận với nó (neighbors), có thể chỉ định nhiều khoảng cách khác nhau bằng cách truyền vào một list. Ví dụ:

graycomatrix(...,distances=[1,2,3],...)

***angles***: số đo góc hợp bởi pixel đang xét và pixel lân cận của nó cùng với đường nằm ngang. Có thể chỉ định nhiều số đo góc khác nhau bằng cách truyền vào một list.

graycomatrix(...,angles=[0,np.pi/2,np.pi/4],...)



***levels***: quy định kích thước của ma trận GLCM. Lưu ý: levels > giá trị mức xám cao nhất trong hình ảnh đầu vào

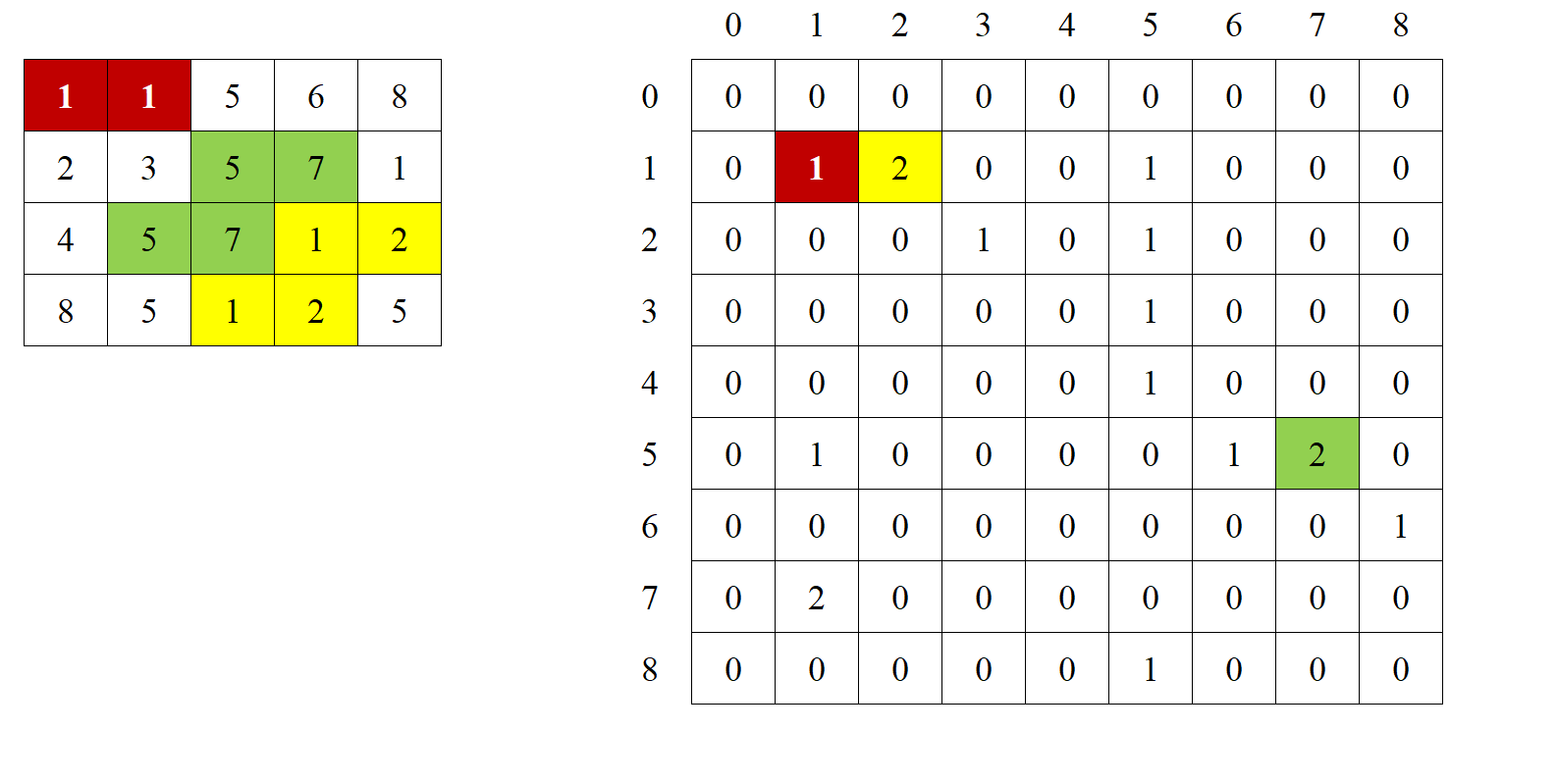
* Ngoài ra còn có các tham số khác như symmetric (tạo ma trận GLCM đối xứng, normed (chuẩn hóa ma trận GLCM), v..v..
* Code minh họa:

import numpy as np

from skimage.feature import graycomatrix

img=np.array([[1,1,5,6,8],[2,3,5,7,1],[4,5,7,1,2],[8,5,1,2,5]],dtype='uint8')

glcm=graycomatrix(image=img,distances=[1],angles=[0],levels=9)



1. ***Tính toán 5 giá trị đặc trưng mô tả các kết cấu của hình ảnh***

*\*Các tham số truyền vào các hàm được viết dưới đây là mảng 2 chiều*

* Entropy

def cal\_entropy(glcm):

    s\_glcm=glcm+glcm.T

    n\_glcm=s\_glcm/np.sum(s\_glcm)

    entropy\_matrix=-np.log(n\_glcm+1e-6)

    return np.sum(entropy\_matrix\*n\_glcm)

* Contrast

def cal\_contrast(glcm):

    s\_glcm=glcm+glcm.T

    n\_glcm=s\_glcm/np.sum(s\_glcm)

    contrast=0

    n=glcm.shape[0]

    for i in range(n):

        for j in range(n):

            contrast=contrast+n\_glcm[i][j]\*(i-j)\*\*2

    return contrast

* Correlation

def cal\_correlation(glcm):

    mu=np.array([0,0])

    sig=np.array([0,0])

    s\_glcm=glcm+glcm.T

    n\_glcm=s\_glcm/np.sum(s\_glcm)

    n=n\_glcm.shape[0]

    for i in range(n):

        for j in range(n):

            mu=mu+np.array([i\*n\_glcm[i,j],j\*n\_glcm[i,j]])

    for i in range(n):

        for j in range(n):

            sig=sig+n\_glcm[i,j]\*((i-mu)\*\*2)

    corr=0

    for i in range(n):

        for j in range(n):

            corr=corr+n\_glcm[i,j]\*(i-mu[0])\*(j-mu[1])/np.sqrt(sig[0]\*sig[1])

    return corr

* Energy

def cal\_energy(glcm):

    s\_glcm=glcm+glcm.T

    n\_glcm=s\_glcm/np.sum(s\_glcm)

    energy=np.sum(n\_glcm\*\*2)

    return np.sqrt(energy)

* Homogeneity

def cal\_homogeneity(glcm):

    s\_glcm=glcm+glcm.T

    n\_glcm=s\_glcm/np.sum(s\_glcm)

    n=glcm.shape[0]

    homo=0

    for i in range(n):

        for j in range(n):

            homo=homo+n\_glcm[i][j]/(1+(i-j)\*\*2)

    return homo

* Trong python có cung cấp hàm *graycoprops* để tính các giá trị đặc trưng kết cấu (nhưng không có entropy)

contrast = graycoprops(glcm,'contrast')

correlation = graycoprops(glcm,'correlation')

energy = graycoprops(glcm,'energy')

homogeneity = graycoprops(glcm,'homogeneity')

* Code minh họa thuật toán trích chọn đặc trưng:

img=cv2.imread('natural\_images/test/airplane/airplane\_0701.jpg',0)

img=cv2.resize(img,(SIZE\_IMAGE,SIZE\_IMAGE))

glcm=graycomatrix(img,[1],[0])

print('Kích thước ma trận hình ảnh: {}x{}'.format(img.shape[0],img.shape[1]))

df = pd.DataFrame()

df['entropy'] = [cal\_entropy(glcm[:,:,0,0])]

df['contrast'] = graycoprops(glcm,'contrast')[0]

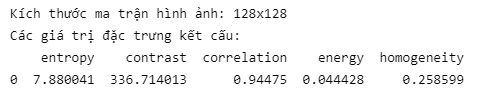
df['correlation'] = graycoprops(glcm,'correlation')[0]

df['energy'] = graycoprops(glcm,'energy')[0]

df['homogeneity'] = graycoprops(glcm,'homogeneity')[0]

print('Các giá trị đặc trưng kết cấu: ')

print(df)



## **Bài toán phân loại hình ảnh**

***Tập dữ liệu***: dữ liệu [natural](https://www.kaggle.com/datasets/prasunroy/natural-images) trên Kaggle, bao gồm 8 lớp phân biệt: airplane, car, cat, dog, flower, fruit, motorbike, person

***Thuật toán học tập***: Light Gradient Boosting Machine (dựa trên thuật toán cây quyết định, thường được sử dụng để xếp hạng, phân loại và nhiều tác vụ học máy khác). Đây là một trong những thuật toán rất phổ biến hiện nay nhờ vào hiệu suất tuyệt vời của nó

***Thuật toán trích chọn đặc trưng được sử dụng***: GLCM, và từ ma trận GLCM trích xuất ra 5 đặc trưng là entropy, contrast, correlation, energy, homogeneity

def feature\_extractor(dataset):

    image\_dataset = pd.DataFrame()

    for image in range(dataset.shape[0]):

        df = pd.DataFrame()

        img = dataset[image, :,:]

        GLCM = graycomatrix(img, [1], [0])

        GLCM\_Energy = graycoprops(GLCM, 'energy')[0]

        df['Energy'] = GLCM\_Energy

        GLCM\_corr = graycoprops(GLCM, 'correlation')[0]

        df['Corr'] = GLCM\_corr

        GLCM\_hom = graycoprops(GLCM, 'homogeneity')[0]

        df['Homogen'] = GLCM\_hom

        GLCM\_contr = graycoprops(GLCM, 'contrast')[0]

        df['Contrast'] = GLCM\_contr

        GLCM\_entropy = cal\_entropy(GLCM[:,:,0,0])

        df['Entropy'] = GLCM\_entropy

        GLCM2 = graycomatrix(img, [3], [0])

        GLCM\_Energy2 = graycoprops(GLCM2, 'energy')[0]

        df['Energy2'] = GLCM\_Energy2

        GLCM\_corr2 = graycoprops(GLCM2, 'correlation')[0]

        df['Corr2'] = GLCM\_corr2

        GLCM\_hom2 = graycoprops(GLCM2, 'homogeneity')[0]

        df['Homogen2'] = GLCM\_hom2

        GLCM\_contr2 = graycoprops(GLCM2, 'contrast')[0]

        df['Contrast2'] = GLCM\_contr2

        GLCM\_entropy2 = cal\_entropy(GLCM2[:,:,0,0])

        df['Entropy2'] = GLCM\_entropy2

        GLCM3 = graycomatrix(img, [5], [0])

        GLCM\_Energy3 = graycoprops(GLCM3, 'energy')[0]

        df['Energy3'] = GLCM\_Energy3

        GLCM\_corr3 = graycoprops(GLCM3, 'correlation')[0]

        df['Corr3'] = GLCM\_corr3

        GLCM\_hom3 = graycoprops(GLCM3, 'homogeneity')[0]

        df['Homogen3'] = GLCM\_hom3

        GLCM\_contr3 = graycoprops(GLCM3, 'contrast')[0]

        df['Contrast3'] = GLCM\_contr3

        GLCM\_entropy3 = cal\_entropy(GLCM3[:,:,0,0])

        df['Entropy3'] = GLCM\_entropy3

        GLCM4 = graycomatrix(img, [0], [np.pi/4])

        GLCM\_Energy4 = graycoprops(GLCM4, 'energy')[0]

        df['Energy4'] = GLCM\_Energy4

        GLCM\_corr4 = graycoprops(GLCM4, 'correlation')[0]

        df['Corr4'] = GLCM\_corr4

        GLCM\_hom4 = graycoprops(GLCM4, 'homogeneity')[0]

        df['Homogen4'] = GLCM\_hom4

        GLCM\_contr4 = graycoprops(GLCM4, 'contrast')[0]

        df['Contrast4'] = GLCM\_contr4

        GLCM\_entropy4 = cal\_entropy(GLCM4[:,:,0,0])

        df['Entropy4'] = GLCM\_entropy4

        GLCM5 = graycomatrix(img, [0], [np.pi/2])

        GLCM\_Energy5 = graycoprops(GLCM5, 'energy')[0]

        df['Energy5'] = GLCM\_Energy5

        GLCM\_corr5 = graycoprops(GLCM5, 'correlation')[0]

        df['Corr5'] = GLCM\_corr5

        GLCM\_hom5 = graycoprops(GLCM5, 'homogeneity')[0]

        df['Homogen5'] = GLCM\_hom5

        GLCM\_contr5 = graycoprops(GLCM5, 'contrast')[0]

        df['Contrast5'] = GLCM\_contr5

        GLCM\_entropy5 = cal\_entropy(GLCM5[:,:,0,0])

        df['Entropy5'] = GLCM\_entropy5

        image\_dataset = image\_dataset.append(df)

    return image\_dataset

Tài liệu tham khảo