**University of Kasdi Merbah, Ouargla**

**Faculty of New Information and Communication Technologies**

**Department of Computer Science and Information Technology**

Thesis Submitted to the Department of computer science and Information in Candidacy for the Degree of Master in computer science, option “informatique industrielle”.

***Presented by:***

***BENSID Kawthar & HAMEL Rekia***

***<u>Title</u>:***

# *On Combinig Deep and Handcrafted Features for Off-line Writer Identification*

| | | |
|---|---|---|
| Dr. AIADI Oussama | Supervisor | UKM Ouargla |
| Dr. AMEUR Khadidja | President | UKM Ouargla |
| Dr. YOUSSFA Abdelmajid | Examiner | UKM Ouargla |

**Academic Year: 2020– 2021**

# *Acknowledgments*

*I dedicate this modest work*

*To my very dear parents for them supports and encouragement during all my years of studies and without them I would have never succeeded.*
*That god gets them good health and long life.*

*To all my professors and teachers that I had during all my school program and which allowed me to succeed in my studies.*

*To my best friends*

*To my Mather and father, to my brothers , I wish all the best for them.*

*To all my sisters.*

*To every person having contributed to this work, closely or remotely.*

## Abstract:

The purpose of this project is to identify the handwritten texts of the individual. In fact, the text is recognized as a means of verifying the identity of the person and will enable the realization of a biometric system of quick and effective control of handwriting, in order to reduce the risk of fraud in a significant way.

In this project, we realized a desktop application that allows the user to recognize the handwriting of the individual. We used the histogram of oriented gradients (Hog) and the co-occurrence histograms of oriented gradients (CoHog) function to extract the parameters before applying identification techniques, which will be explained later. These techniques will give us a result that reflects the process followed by our application. The latter has been tested and validated on a test basis.

**Key words:**

Handwriting, biometric system, identification, CoHog , Hog , Deep(VGG).

# TABLE OF CONTENTS

## TABLE OF FIGURES

# CHAPTER 1: GENERAL INTRODUCTION

## Introduction :

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Biometrics is a sub-field of computer vision, which aiming at recognizing a person's identity based on an individual's physical or behavioral attributes such as the face, fingerprints, voice, or iris. In Biometric systems, a variety of physical or behavioral characteristics are used, including fingerprints, face, hand / finger geometry, retina, signature, gait, palm print, ear, vein or DNA information to establish their identity. In the biometric literature, these characteristics are called traits, indicators, identifiers or also modalities.

Analysis of handwriter is an established area in the forensic sciences and, in addition to other applications, analysis of handwriter for forensic applications has continued to be an attractive problem for document examiners. The individuality of handwriter has been studied in a number of works and evidences provided by handwritten documents have been employed in courts for many years . Traditionally, the task of forensic writer identification (that involves identifying the writer of a questioned document) is carried out by forensic document experts using standard methodologies as outlined in Refs. In the past, there has been a hesitancy from the domain experts to employ computerized solutions for this problem . The advances in different areas of image analysis and pattern classification however, have led to acceptability of computerized solutions by forensic experts. It is important to mention that such solutions are aimed at facilitating and not replacing the human experts. Considering the scale of forensic databases (order of 104 samples), automatic systems can be employed to reduce the search space so that human experts can focus on a manageable sized list of potential candidates. Schomaker et al. establish that the target performance of computerized writer identification systems is near to 100% identification rate in hit list of 100 writers from writing collections matching the size of real world forensic databases.

## 2.Problematic:

The problem of identifying the writer of a handwritten document image has been an active research area over the last few years and enjoys applications in forensic and historical document analysis. We have developed an effective method for automatic writer identification and verification from unconstrained handwritten text images. Our method relies on two different aspects of writing: the presence of redundant patterns in the writing and its visual attributes. Based on the hypothesis that handwriting carries certain patterns that an individual would use frequently as he writes, we look to extract these patterns by analyzing small writing fragments and grouping similar patterns into clusters. In fact this corresponds more to the redundancy of writing gestures than writing shapes. These clusters are determined either for each of the writers separately or, for a group of writers generating a universal set of patterns. The writing in question is then compared to the produced clusters. We next exploit two important visual attributes of writing, the orientation and curvature, which enable to distinguish one writing from another. These attributes are extracted by computing a set of features from writing samples at different levels of observation. Two writings are then compared by computing distances between their respective features. Finally, we combine the two facets of handwriting to characterize the writer of a handwritten sample. The proposed methodology, evaluated on modern as well as ancient writings exhibited promising results on tasks of writer recognition and handwriting classification.

## 3.Motivations:

The Histogram of Oriented Gradient represent the internal structure of an object using the information of the gradient, there by overcoming the problems associated with the appearance of the object: pose, lighting, occlusion, texture, etc; Using the sliding window approach, the filter is applied at all positions and scales of an image followed by the non-maxima suppression.

Co-HOG, a variant of HOG, uses a pair of gradient orientations as its basic building block, unlike HOG which uses a single gradient. Using the pair of orientations, the co-occurrence matrix is computed and histograms are calculated. However, in Co-HOG, gradient direction alone is considered and magnitude is ignored. It is believed that gradient magnitude also carries significant information about features.

In addition to the above reasons, we noted that few attention has been devoted to use deep neural networks (e.g., Convolutional Neural Network (CNN)), in spite of their impressive achievements on different state of the art tasks. Deep networks have received minor attention compared the handcrafted features. We believe that the architecture of CNN may be the reason behind that. Early layers encodes low-level image features such as edges, corners, flat lines…etc. while mid-level layers can capture parts of objects e.g., mouth from face image. The last layers are interested with the entire object. Thus, feeding an input handwriting sample to a deep neural network until reaching up to the soft-max layer may be not suitable in the case of handwriting recognition. Therefore, extracting features learned by early CNN layers, and supplying those features to a conventional classifier could be more useful than the former case.

## 4.Contributions:

The contributions of this thesis can be summarized as follows investigating the performance of Hog's variables in the context of author identification. For each Hog variant, we try to reveal its pros as well as its drawbacks to open the door for further improvements. Using CoHog, we compare the performance of those variants with deep features. Using CNN and Vgg(16) to classify handwritten numbers in the IAM dataset. We run experiments on the well-known IAM dataset of 13,353 isolated lines of text written by 657 writers.

It should be noted that the Hog variables are determined according to the following criteria: publication date (preferably newer over older versions), number of citations (Google Scholar citations), proficiency: most variables are originally intended to identify people via biometric traits eg face, ear ...etc. Therefore, having successful performance in such tasks might release a good indicator to extend this performance for the simplicity of our context and method, and efficient versions with a low computation budget are highly recommended.

# CHAPTER 2: WORK BACKGROUND

## 1. Introduction:

Image processing can be defined as the set of methods and technique operating on the image in order to extract the most relevant information or quite simply to provide an image more perceptible to the human eye. In this chapter we present some basic nations of the field of digital image processing such as: image definition, image types.

Feature engineering is a game-changer in the world of machine learning algorithms. It’s actually one of my favorite aspects of being a data scientist! This is where we get to experiment the most – to engineer new features from existing ones and improve our model’s performance.

Some of the top data scientists in the world rely on feature engineering to boost their leader board score in hackathons. I’m sure you would even have used various feature engineering techniques on structured data.

## 2.The image:

An image may be defined as a two dimensional function, f(x,y), where x and y are spatial(plane) coordinates, and the amplitude of at any pair of coordinates(x,y) is called the intensity or gray level of the image at that point.

When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a digital image [6]**.**

## 3. Type of images:

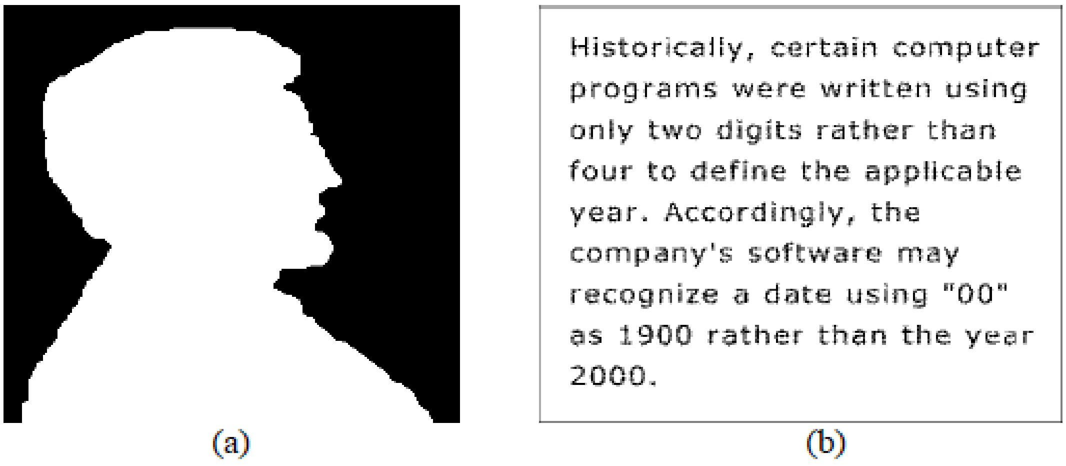
The images types we will consider are:

1. Binary.
2. Gray-scale.
3. Color.
4. Multispectral.

## 3.1 Binary images:

Binary images are the simplest type of images and can take on two values, typically black and white, or 0 and 1. A binary image is referred to as a 1-bit image because it takes only 1 binary digit to represent each pixel. These types of images are frequently used in applications where the only information required is general shape or outline, for example optical character recognition(OCR).

Binary images are often created from the gray-scale images via a threshold operation, where every pixel above the thrzshold value is turned white (‘1’), and those below it are turned black (‘0’). In the figure below, we see examples of binary images [1].



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

***Figure 1:*** ***Examples of binary images.***

## 3.2 Gray-scale images:

Gray-scale images are referred to as monochrome (one-color) images. They contain gray-level information, no color information. The number of bits used for each pixel determines the number of different gray levels available. The typical gray-scale image contains 8bits/pixel data, which allows us to have 256 different gray levels. The figure below shows example of gray-scale images.

In applications like medical imaging and astronomy, 12 or 16 bits/pixel images are used. These extra gray levels become useful when a small section of the image is made much larger to discern details [1].

*Figure 2: Examples of gray-scale images*

## 3.3.Color images :

Color images can be modeled as three-band monochrome image data, where each band of data corresponds to a different color. The actual information stored in the digital image data is the gray-level information in each spectral band.

Typical color image are represented as red, green, and blue (RGB images). Using the 8-bit monochrome standard as a model, the of the three color bands red, green and blue). The figure below illustrates a representation of a typical RGB color image[1].

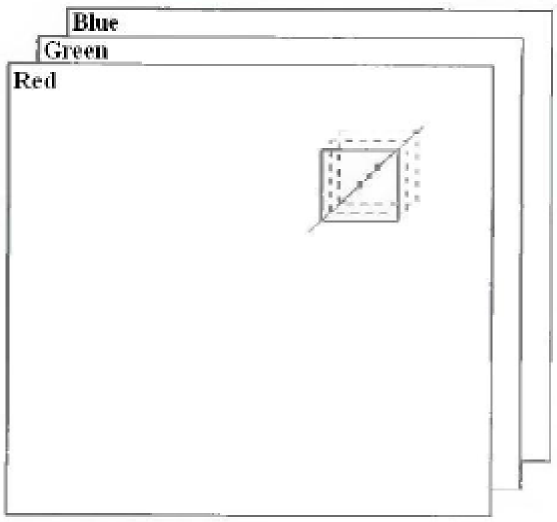


*Figure 3: Illustrates a representation of a typical RGB color image.*

### 3.4.Multispectral images:

Multispectral images typically contain information outside the normal human perceptual range. This may include infrared, ultraviolet, X-ray.

Acoustic , or radar data. These are not images in the usual sense because the information represented is not directly visible by the humansystem. However, the information is often represented in visual from by mapping the different spectral bands to RGB components[1].

## 4. Edge Detection:

Edge detection is a technique of image processing used to identify points in a digital image with discontinuities, simply to say, sharp changes in the image brightness. These points where the image brightness varies sharply are called the edges (or boundaries) of the image.

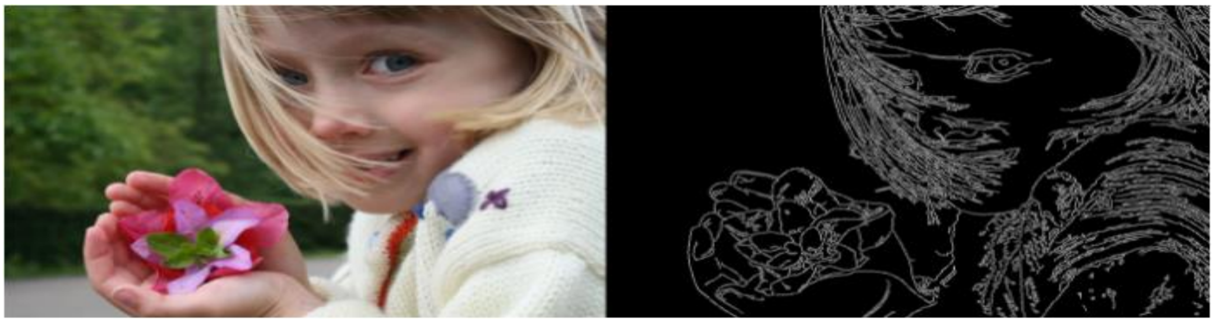


***Figure 4: Exemple Edge Detection.***

It is one of the basic steps in image processing, pattern recognition in images and computer vision. When we process very high-resolution digital images, convolution techniques come to our rescue. Let us understand the convolution operation.

we are using 3*3 Prewitt filter as shown in the above image. As shown below, when we apply the filter to perform edge detection on the given 6*6 image (we have highlighted it in purple for our understanding) the output image will contain in the purple square. We repeat the convolutions horizontally and then vertically to obtain the output image [2].

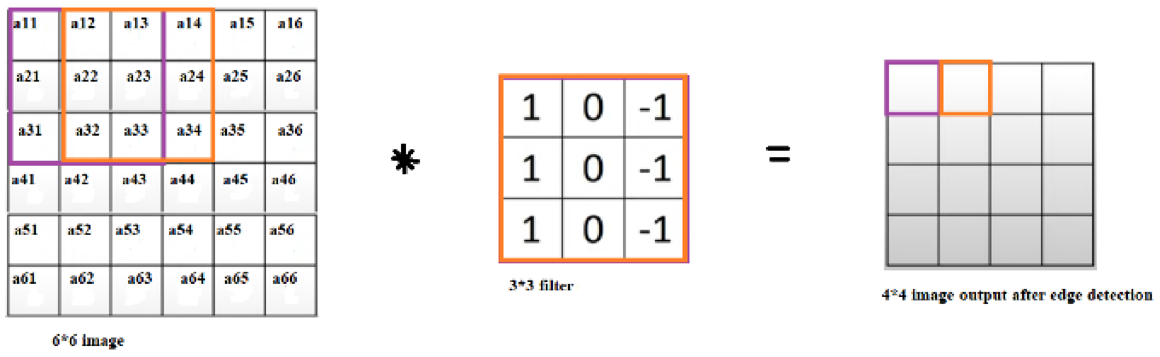


***Figure 5: represented in the below image using.***

We would continue the above procedure to get the processed image after edge-detection. But, in the real world, we deal with very high-resolution images for Artificial Intelligence applications. Hence we opt for an algorithm to perform the convolutions, and even use Deep Learning to decide on the best values of the filter.

There are various methods in edge detection, and the following are some of the most commonly used methods-

- Prewitt edge detection
- Sobel edge detection
- Laplacian edge detection
- Canny edge detection

## 4.1.Prewitt Edge Detection :

This method is a commonly used edge detector mostly to detect the horizontal and vertical edges in images. The following are the Prewitt edge detection filters-

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

***Figure 6: Prewitt filter for vertical and horizontal edge detection.***

## 4.2. Sobel Edge Detection:

This uses a filter that gives more emphasis to the centre of the filter. It is one of the most commonly used edge detectors and helps reduce noise and provides differentiating, giving edge response simultaneously. The following are the filters used in this method-

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

*Figure 7: Sobel filter for vertical and horizontal edge detection.*

## 4.3. Laplacian Edge Detection:

The Laplacian edge detectors vary from the previously discussed edge detectors. This method uses only one filter (also called a kernel). In a single pass, Laplacian edge detection performs second-order derivatives and hence are sensitive to noise. To avoid this sensitivity to noise, before applying this method, Gaussian smoothing is performed on the image.

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

**Common Laplacian edge detection filters**

*Figure 8: Common Laplacian edge detection filters.*

The above are some of the commonly used Laplacian edge detector filters that are small in size.

### 4.4.Canny Edge Detection:

This is the most commonly used highly effective and complex compared to many other methods. It is a multi-stage algorithm used to detect/identify a wide range of edges. The following are the various stages of the Canny edge detection algorithm-

- Convert the image to grayscale
- Reduce noise – as the edge detection that using derivatives is sensitive to noise, we reduce it.
- Calculate the gradient – helps identify the edge intensity and direction.
- Non-maximum suppression – to thin the edges of the image.
- Double threshold – to identify the strong, weak and irrelevant pixels in the images.
- Hysteresis edge tracking – helps convert the weak pixels into strong ones only if they have a strong pixel around them.

The following are the original minion image and the image after applying this method.

## 5. computer vision:

Computer vision is the field of computer science that focuses on creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process an image at a pixel level and understand it. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms.

## 6. Applications of computer vision :

### 6.1.Content organization :

Computer vision systems already help us organize our content. Apple Photos is an excellent example. The app has access to our photo collections, and it automatically adds tags to photos and allows us to browse a more structured collection of photographs. What makes Apple Photos great is that the app creates a curated view of your best moments for you.

### 6.2.Augmented reality :

Computer vision is a core element of augmented reality apps. This technology helps AR apps to detect physical objects (both surfaces and individual objects within a given physical space) in real-time and use this information to place virtual objects within the physical environment.

### 6.3.Health :

Image information is a key element for diagnosis in medicine because it accounts for 90 percent of all medical data. Many diagnoses in health are based on image processing—X-rays, MRI, and mammography, just to name a few. And image segmentation proved its effectiveness during medical scans analysis. For example, computer vision algorithms can detect diabetic retinopathy, the fastest-growing cause of blindness. Computer vision can process pictures of the back of the eye (see below) and rate them for disease presence and severity.

### 6.4.Agriculture :

Many agricultural organizations employ computer vision to monitor the harvest and solve the common agricultural problems such as weeds emergence or nutrient deficiency. Computer vision systems process images from satellites, drones, or planes, and attempt to detect the problems in the early phase, which helps to avoid unnecessary financial losses.

## 7.Systme biométrique :

A biometric system is a pattern recognition system, it uses an individual's biometric data to operate in enrollment mode or in verification or identification (recognition) mode. It captures the biometric signal, processes it and then extracts a set of representative characteristics called biometric models which will then be compared with the model previously stored on a database.

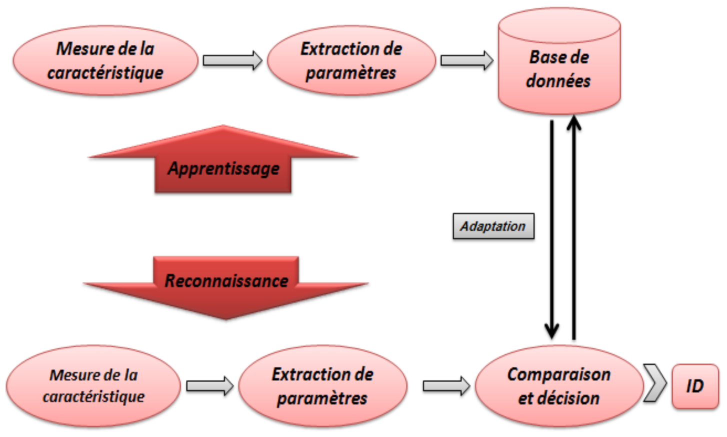


*Figure 9: The architecture of biometric systems.*

## 8.Writer Identification:

Handwriting is a behavioral biometric (other examples are voice, signature, gait) that varies between individuals. When writing letters of a word, each person is used to writing in unique, characteristic shapes, called allographs. As an example, we can see a clear variation of handwriting from three different writers in Figure 10. It is this clear variation that permits writer identification even when the same text is written.

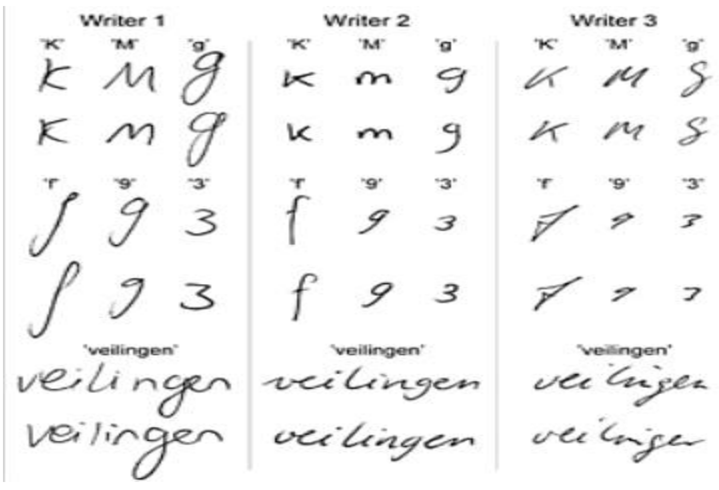


***Figure 10: An obvious difference of character shapes and writing style can be seen between three different writers. Image from.***

## 9.Feature Extraction:

### 9.1.COLOR FEATURE EXTRACTION:

Color space represents the color in the form of intensity value. We can specify, visualize and create the color by using color space method. There are different color feature extraction methods [9].

Color feature extraction methods:

- Histogram Intersection Method.
- Zernike Chromaticity Distribution Moments.
- Color Histogram.

### 9.2.SHAPE FEATURE EXTRACTION:

Shape is main source of information which is used for object recognition. Without shape visual content object cannot be recognize properly. Image is incomplete without recognizing shape. The two objects cannot have exact same shape but by using various algorithms we can recognize similar shape easily. The main problem to recognize shape that the setting location the object shows different shape from different location, so it is measure problem to recognize the actual shape of the object [3].

### 9.3.TEXTURE FEATURE EXTRACTION:

Texture contains significant information about the basic arrangement of the surface that is clouds, leaves, bricks, fabric etc. It also defines surface with environment relationship. Texture feature also describes the physical composition of surface. There are different methods of texture feature extraction [3]:

- The Grey Level Co-occurrence Matrix.
- Edge Detection.
- Laws Texture Energy Measures.

## 10. Conclusion:

All pattern recognition system has physical data as a common point. At from which information is derived and mathematically modeled in order to facilitate their digital processing and make them evolve in a virtual space and possibly predict their behavior.

Processing an image (after scanning) improves its quality, by reducing noise and detecting the presence of certain shapes in order to batter approximate the original image, so that it is ready to be used in other application.

# CHAPTER 3: MATERIAL AND METHODS

## 1.Introduction:

The goal of this chapter is to provide an overview as histogram of oriented gradients (HOG) a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts co-occurrences of gradient orientation in localized portions of an image – detection and co-occurrence histograms of oriented gradients(CoHog).

The architecture of a convolutional neural network is a multi-layered feed-forward neural network, made by stacking many hidden layers on top of each other in sequence. It is this sequential design that allows convolutional neural networks to learn hierarchical features.

## 2.The HOG Feature Descriptor:

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. It is widely used in computer vision tasks for object detection.

Let's look at some important aspects of HOG that makes it different from other feature descriptors:

The HOG descriptor focuses on the structure or the shape of an object. Now you might ask, how is this different from the edge features we extract for images? In the case of edge features, we only identify if the pixel is an edge or not. HOG is able to provide the edge direction as well. This is done by extracting the **gradient and orientation** (or you can say magnitude and direction) of the edges [4].

Additionally, these orientations are calculated in 'localized' portions. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated. We will discuss this in much more detail in the upcoming sections

Finally the HOG would generate a Histogram for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'

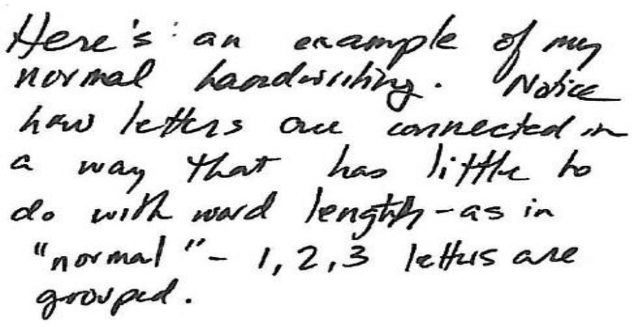
To put a formal definition to this:

The HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image.

Process of Calculating the Histogram of Oriented Gradients (HOG)

We should now have a basic idea of what a HOG feature descriptor is. It's time to delve into the core idea behind this article. Let's discuss the step-by-step process to calculate HOG.

Consider the below image of size (180 x 280). Let us take a detailed look at how the HOG features will be created for this image:



Here's: an example of my normal handwriting. Notice how letters are connected in a way that has little to do with word length - as in "normal" - 1,2,3 letters are grouped.

***Figure 11: image handwriting simple.***

**Step 1:** Preprocess the Data (64 x 128)

This is a step most of you will be pretty familiar with. Preprocessing data is a crucial step in any machine learning project and that's no different when working with images.

We need to preprocess the image and bring down the width to height ratio to 1:2. The image size should preferably be 64 x 128. This is because we will be dividing the image into 8*8 and 16*16 patches to extract the features. Having the specified size (64 x 128) will make all our calculations pretty simple. In fact, this is the exact value used in the *original paper*.

Coming back to the example we have, let us take the size 64 x 128 to be the standard image size for now. Here is the resized image:

**Step 2:** Calculating Gradients (direction x and y)

The next step is to calculate the gradient for every pixel in the image. Gradients are the small change in the x and y directions. Here, I am going to take a small patch from the image and calculate the gradients on that:

We will get the pixel values for this patch. Let's say we generate the below pixel matrix for the given patch (the matrix shown here is merely used as an example and these are not the original pixel values for the given patch):

| | | | | |
|---|---|---|---|---|
| 121 | 10 | 78 | 96 | 125 |
| 48 | 152 | 68 | 125 | 111 |
| 145 | 78 | 85 | 89 | 65 |
| 154 | 214 | 56 | 200 | 66 |
| 214 | 87 | 45 | 102 | 45 |

***Table 1: The below pixel matrix for the given patch.***

I have highlighted the pixel value 85. Now, to determine the gradient (or change) in the x-direction, we need to subtract the value on the left from the pixel value on the right. Similarly, to calculate the gradient in the y-direction, we will subtract the pixel value below from the pixel value above the selected pixel.

Hence the resultant gradients in the x and y direction for this pixel are:

Change in X direction ($G_x$) = 89 – 78 = 11

Change in Y direction ($G_y$) = 68 – 56 = 8

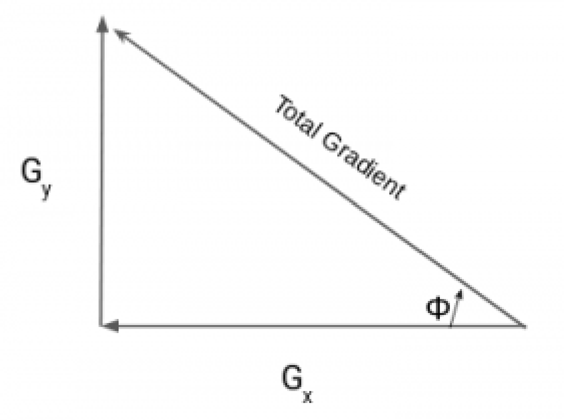
This process will give us two new matrices – one storing gradients in the x-direction and the other storing gradients in the y direction. This is similar to using a Sobel Kernel of size 1. The magnitude would be higher when there is a sharp change in intensity, such as around the edges.

We have calculated the gradients in both x and y direction separately. The same process is repeated for all the pixels in the image. The next step would be to find the magnitude and orientation using these values.

**Step 3:** Calculate the Magnitude and Orientation

Using the gradients we calculated in the last step, we will now determine the magnitude and direction for each pixel value. For this step, we will be using the Pythagoras theorem (yes, the same one which you studied back in school!).

Take a look at the image below:



The gradients are basically the base and perpendicular here. So, for the previous example, we had Gx and Gy as 11 and 8. Let's apply the Pythagoras theorem to calculate the total gradient magnitude:

$$\text{Total Gradient Magnitude} = \sqrt{G_x^2 + G_y^2}$$

Next, calculate the orientation (or direction) for the same pixel. We know that we can write the tan for the angles:

$$\tan \emptyset = \frac{Gy}{Gx}$$

Hence, the value of the angle would be:

$$\tan \emptyset = \arctan \frac{Gy}{Gx}$$

The orientation comes out to be 36 when we plug in the values. So now, for every pixel value, we have the total gradient (magnitude) and the orientation (direction). We need to generate the histogram using these gradients and orientations.

But hang on – we need to take a small break before we jump into how histograms are created in the HOG feature descriptor. Consider this a small step in the overall process. And we'll start this by discussing some simple methods of creating Histograms using the two values that we have – gradients and orientation.

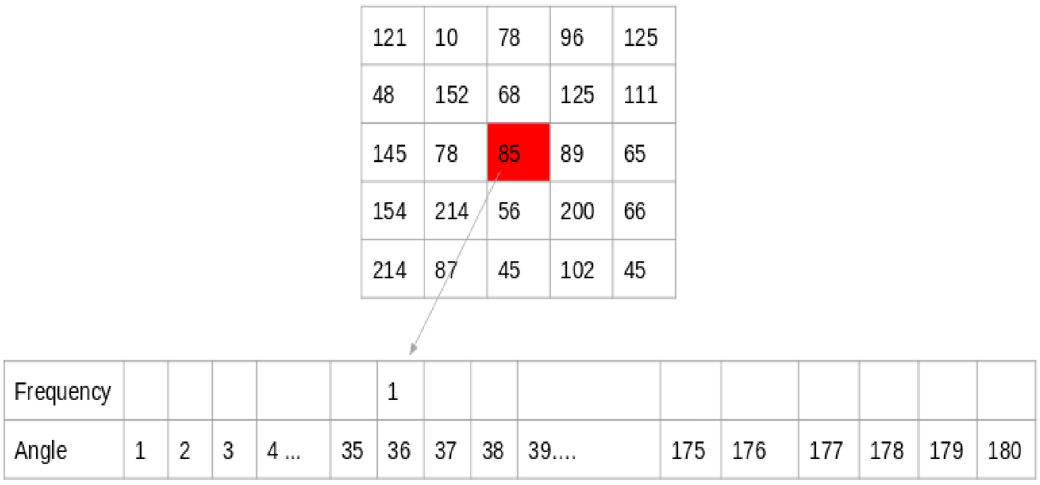
Different Methods to Create Histograms using Gradients and Orientation

A histogram is a plot that shows the frequency distribution of a set of continuous data. We have the variable (in the form of bins) on the x-axis and the frequency on the y-axis. Here, we are going to take the angle or orientation on the x-axis and the frequency on the y-axis.

**Method 1:**

Let us start with the simplest way to generate histograms. We will take each pixel value, find the orientation of the pixel and update the frequency table.

Here is the process for the highlighted pixel (85). Since the orientation for this pixel is 36, we will add a number against angle value 36, denoting the frequency:



| | | | | |
|---|---|---|---|---|
| 121 | 10 | 78 | 96 | 125 |
| 48 | 152 | 68 | 125 | 111 |
| 145 | 78 | 85 | 89 | 65 |
| 154 | 214 | 56 | 200 | 66 |
| 214 | 87 | 45 | 102 | 45 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Frequency | | | | | | 1 | | | | | | | | | | |
| Angle | 1 | 2 | 3 | 4 ... | 35 | 36 | 37 | 38 | 39.... | 175 | 176 | 177 | 178 | 179 | 180 | |

***Table 2: The  Frequency Table.***

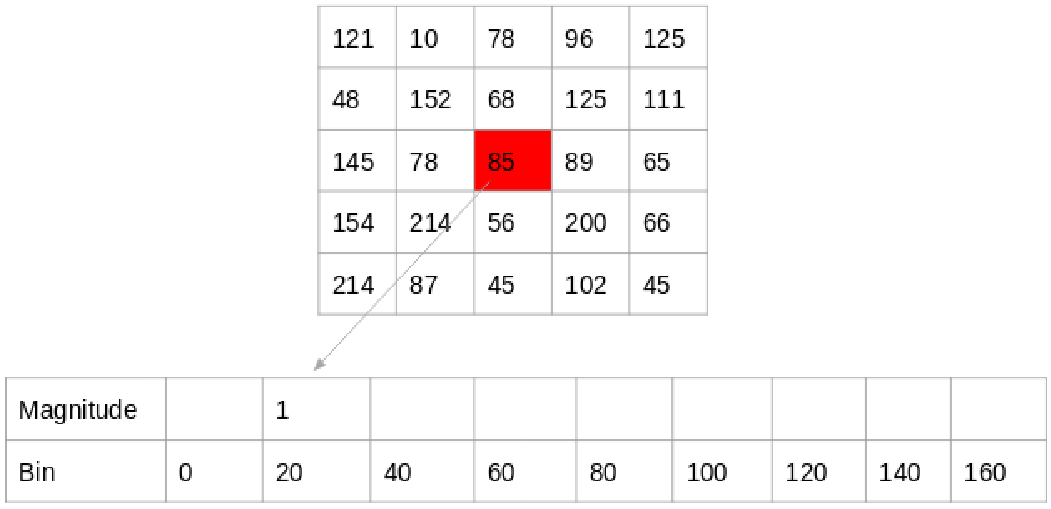
The same process is repeated for all the pixel values, and we end up with a frequency table that denotes angles and the occurrence of these angles in the image. This frequency table can be used to generate a histogram with angle values on the x-axis and the frequency on the y-axis.

That’s one way to create a histogram. Note that here the bin value of the histogram is 1. Hence we get about 180 different buckets, each representing an orientation value. Another method is to create the histogram features for higher bin values.

**Method 2:**

This method is similar to the previous method, except that here we have a bin size of 20. So, the number of buckets we would get here is 9.

Again, for each pixel, we will check the orientation, and store the frequency of the orientation values in the form of a 9 x 1 matrix. Plotting this would give us the histogram:



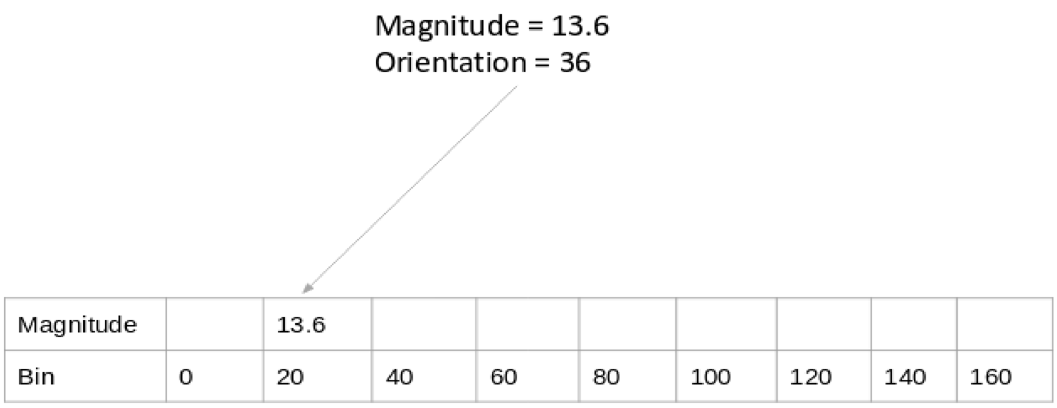
| | | | | |
|---|---|---|---|---|
| 121 | 10 | 78 | 96 | 125 |
| 48 | 152 | 68 | 125 | 111 |
| 145 | 78 | 🟥 85 | 89 | 65 |
| 154 | 214 | 56 | 200 | 66 |
| 214 | 87 | 45 | 102 | 45 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Magnitude | | 1 | | | | | | | |
| Bin | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

***Table 3: Table of magnitude.***

**Method 3:**

The above two methods use only the orientation values to generate histograms and do not take the gradient value into account. Here is another way in which we can generate the histogram – instead of using the frequency, we can use the gradient magnitude to fill the values in the matrix. Below is an example of this:


Magnitude = 13.6
Orientation = 36

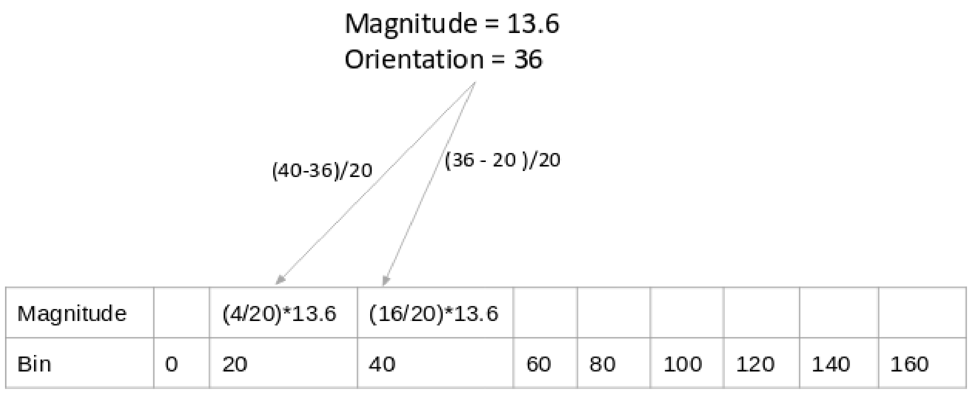
| Magnitude | | 13.6 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Bin | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

***Table 4: Table of orientation.***

You might have noticed that we are using the orientation value of 30, and updating the bin 20 only. Additionally, we should give some weight to the other bin as well.

**<u>Method 4:</u>**

Let’s make a small modification to the above method. Here, we will add the contribution of a pixel’s gradient to the bins on either side of the pixel gradient. Remember, the higher contribution should be to the bin value which is closer to the orientation.


Magnitude = 13.6
Orientation = 36
(40-36)/20
(36 - 20 )/20

| Magnitude | | (4/20)*13.6 | (16/20)*13.6 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Bin | 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

***Table 5 :Table of orientation.***

This is exactly how histograms are created in the HOG feature descriptor.

**Step 4:** Calculate Histogram of Gradients in 8×8 cells (9×1)

The histograms created in the HOG feature descriptor are not generated for the whole image. Instead, the image is divided into 8×8 cells, and the histogram of oriented gradients is computed for each cell. Why do you think this happens?

By doing so, we get the features (or histogram) for the smaller patches which in turn represent the whole image. We can certainly change this value here from 8 x 8 to 16 x 16 or 32 x 32.

If we divide the image into 8×8 cells and generate the histograms, we will get a 9 x 1 matrix for each cell. This matrix is generated using method 4 that we discussed in the previous section.

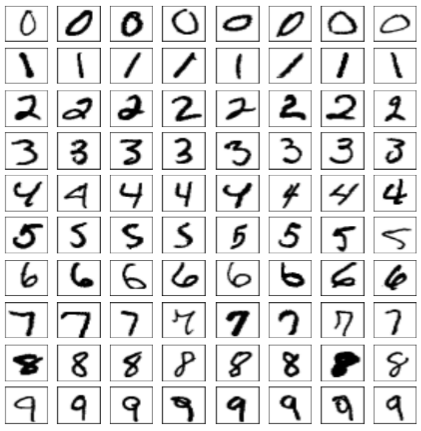


*Figure 12: Image into 8×16 cells.*

Once we have generated the HOG for the 8×8 patches in the image, the next step is to normalize the histogram.

**Step 5:** Normalize gradients in 16×16 cell (36×1)

Before we understand how this is done, it’s important to understand why this is done in the first place.

Although we already have the HOG features created for the 8×8 cells of the image, the gradients of the image are sensitive to the overall lighting. This means that for a particular picture, some portion of the image would be very bright as compared to the other portions.

We cannot completely eliminate this from the image. But we can reduce this lighting variation by normalizing the gradients by taking 16×16 blocks. Here is an example that can explain how 16×16 blocks are created:

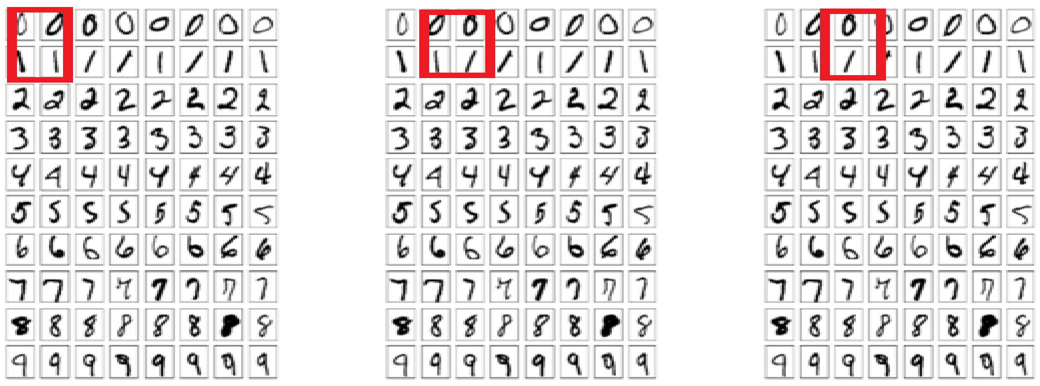


*Figure 13: Orientation histograms for overlapping  2×2 blocks of cells.*

Here, we will be combining four 8×8 cells to create a 16×16 block. And we already know that each 8×8 cell has a 9×1 matrix for a histogram. So, we would have four 9×1 matrices or a single 36×1 matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector V:

$$V = [a_1, a_2, a_3, \dots, a_{36}]$$

We calculate the root of the sum of squares:

$$K = \sqrt{a_1^2 + a_2^2 + a_3^2 + \dots + a_{36}^2}$$

And divide all the values in the vector V with this value k:

$$\text{Normalised Vector} = \left(\frac{a_1}{k}, \frac{a_2}{k}, \frac{a_3}{k}, \dots \frac{a_{36}}{k}\right)$$

The resultant would be a normalized vector of size 36×1.

**Step 6:** Features for the complete image

We are now at the final step of generating HOG features for the image. So far, we have created features for 16×16 blocks of the image. Now, we will combine all these to get the features for the final image.

Can you guess what would be the total number of features that we will have for the given image? We would first need to find out how many such 16×16 blocks would we get for a single 64×128 image:

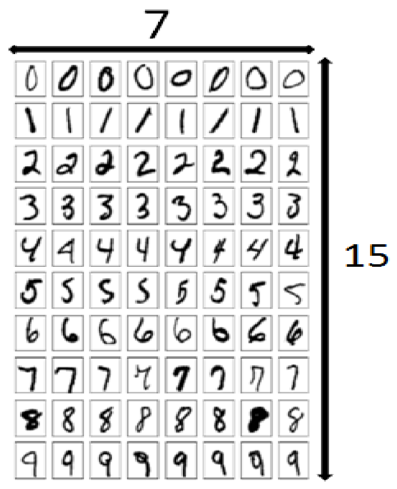


***Figure 14: Image into 8×8 cells.***

We would have 105 (7×15) blocks of 16×16. Each of these 105 blocks has a vector of 36×1 as features. Hence, the total features for the image would be 105 x 36×1 = 3780 features.

We will now generate HOG features for a single image and verify if we get the same number of features at the end.

## 3.Co-occurrence Histograms of Oriented Gradients (CoHOG) :

We propose a high-dimensional feature "Co-occurrence Histograms of Oriented Gradients (CoHOG)". Our feature uses pairs of gradient orientations as units, from which it builds the histograms. The histogram is referred to as the cooccurrence matrix, hereafter. The co-occurrence matrix expresses the distribution of gradient orientations at a given offset over an image as shown in Fig. 14. The combinations of neighbor gradient orientations can express shapes in detail. It is informative for pedestrian classification. Mathematically, a co-occurrence [5].

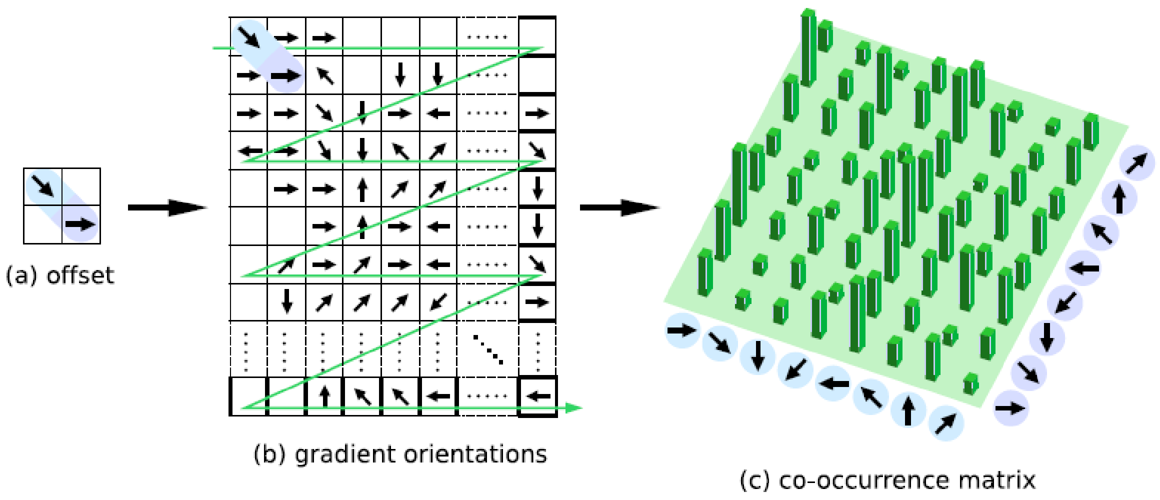


*Figure 15: Co-occurrence matrix of gradient orientations. It calculates sums of all pairs of gradient orientations at a given offset.*

matrix C is defined over an n× m image I, parameterized by an offset (x, y), as:

$$C_{(i,j)}(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1, \text{if } I(p,q) = i \text{ and } I(p+x, q+y) = j \\ \quad\quad 0, \text{otherwise.} \end{cases}$$

CoHOG has robustness against deformation and illumination variance for the same reasons as HOG, because CoHOG is gradient based histogram feature descriptor.

We describe the processes of CoHOG calculation shown in Fig. 15. Firstly, we compute gradient orientations from an image by

$$O = \arctan \frac{v}{h}$$

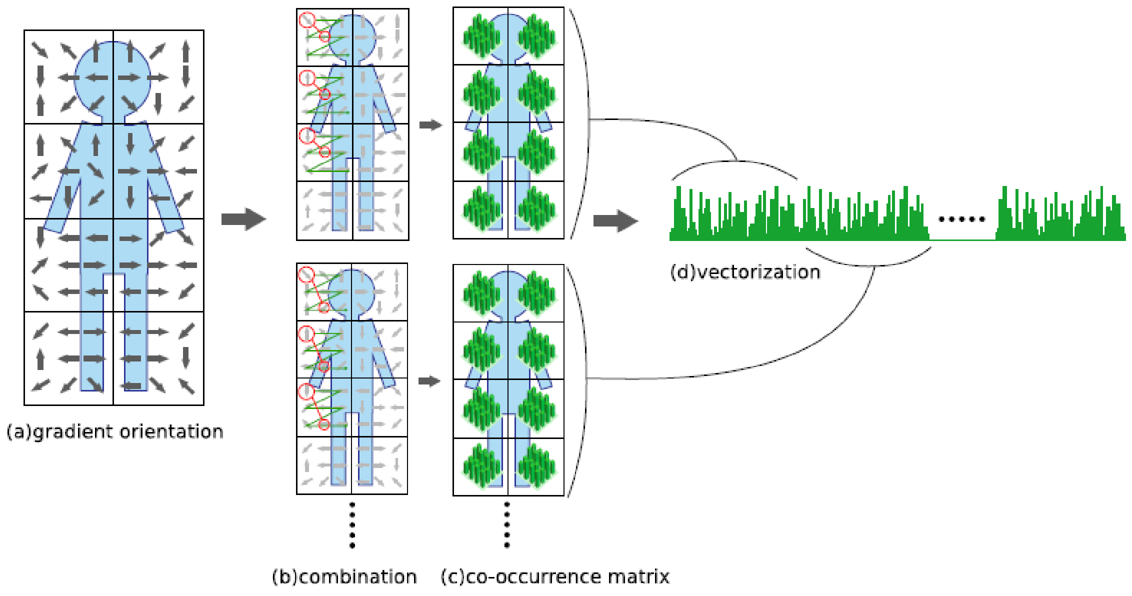


*Figure 16: Overview of CoHOG calculation.*

where v and h are vertical and horizontal gradient respectively calculated by Sobel filter, Roberts filter, etc. We label each pixel with one of eight discrete orientations or as no-gradient (Fig. 15(a)). All 0◦ – 360◦ orientations are divided into eight orientations per 45◦. No-gradient means √v2 + h2 is smaller than a threshold. Secondly, we compute co-occurrence matrices by Eq. (1) (Fig. 15(b)). By using short-range and long-range offsets, the co-occurrence matrix can express local and global shapes. We do not use half of the offsets, because they behave as same as the others in calculation of co-occurrence matrix as shown in Fig. 16.

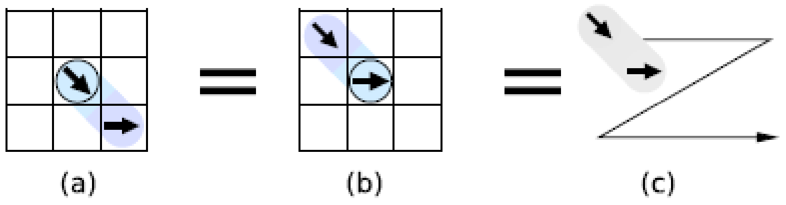


*Figure 17: Offset values of (a) (1, 1) and (b) (−1,−1) are different, but they behave as same as the other in the calculation of co-occurrence matrix.*

```
1: given $I$: an image of gradient orientation
2: initialize $H \leftarrow 0$
3: for all positions $(p, q)$ inside of the image do
4:     $i \leftarrow I(p, q)$
5:     $k \leftarrow$ the small region including $(p, q)$
6:     for all offsets $(x, y)$ such that corresponds neighbors do
7:         if $(p + x, q + y)$ is inside of the image then
8:             $j \leftarrow I(p + x, q + y)$
9:             $H(k, i, j, x, y) \leftarrow H(k, i, j, x, y) + 1$
10:        end if
11:    end for
12: end for
```

***Figure 18: Implementation of CoHOG calculation. The bins of histogram H are initialized to zero before voting. All pixels in the gradient orientation image I are scanned, and bins of H corresponding to pixels are incremented.***

The dashed-circle is the maximum range of offsets. We can get 31 offsets including a zero offset. The co-occurrence matrices are computed for each small region (Fig. 15(c)). The small rectangular regions are tiled N × M, such as 3 × 6 or 6 × 12, with no overlapping. Finally, the components of all the co-occurrence matrices are concatenated into a vector (Fig. 15(d)). Since CoHOG expresses shapes in detail, it is high-dimensional. The dimension is 34, 704, when the small regions are tiled 3 × 6. From one small region, CoHOG obtains 31 co-occurrence matrices. A co-occurrence matrix has 64 components (Fig. 14(c)). The co-occurrence matrix calculated with zero offset has only eight effective values because non-diagonal components are zero. Thus CoHOG obtains $(64 \times 30 + 8) \times (3 \times 6) = 34,704$ components from an image. In fact, the effective values are fewer than 34, 704, because co-occurrence matrices have multiple zero valued components. Zero valued components are not used in classification, because their inner product is zero at all times. Nevertheless, CoHOG is a more powerful feature descriptor than HOG. The implementation of CoHOG is simple. An example of CoHOG implementation is shown in Fig. 17. We can calculate CoHOG by only iterating to increment the components of co-occurrence matrices, whereas HOG calculation includes more procedures, such as orientation weighted voting, histogram normalization, region overlapping, and etc. CoHOG can achieve high performance without those complex procedures.

## 4.Convolutional Neural Networks (CNN):

One of the most effective models for deep learning is the Convolutional Neural Network (CNN) [3].The CNN comprises two unique types of layers, pooling layers and convolution. Layers of CNN contain well-designed filters to handle with input data. They convolve the range of input values, and finally get smaller range of them. And then, CNN can detect essential or specific features within the range we acquired before. The CNN generally consists of the input layer, convolution layer and Rectified Linear Unit. Rectified Linear Unit (ReLU) is mathematically expressed as Max(0, x), Max pooling and the final output layer [3], as shown in Fig.2. Convolution layer can produce a matrix of smaller dimension than input matrix, and max pooling can transmit the maximum value from amongst a rather small batch of data of the input matrix to the output. The output layer is fullyconnected, and based on the activation function [6].

Total input to the j-th feature map of layer l at position (x,y):

$$\mathrm{V}_{\mathrm{j}}^{\mathrm{l}}(\mathrm{x},\mathrm{y}) = \sum_{i=1}^{l} \sum_{u,v=0}^{f-1} \mathrm{k}_{\mathrm{ji}}^{\mathrm{l}}(\mathrm{u},\mathrm{v}).\, \mathrm{O}_{\mathrm{i}}^{(\mathrm{l}-1)}(\mathrm{x}-\mathrm{u},\mathrm{y}-\mathrm{v}) + \mathrm{b}_{\mathrm{j}}^{(\mathrm{l})}$$

Convolution layer output:

$$O_i^{(l-1)}(x,y) = f\left(v_j^{(l)}(x,y)\right)$$

Pooling layer output:

$$O_i^{(l-1)}(x,y) = \max_{u,v=0\ldots G-1} O_i^{(l)}(x.s+u, y.s+v)$$

Where $O_i^{(l-1)}(i = 1, \ldots, l)$ represents featuremaps on the l+1 layer; where $k_{ji}^{(l)}(u,v)$: indicates trainable convolution kernel; $b_i^{(l)}$ means trainable bias; G is pooling size; and S denotes stride (spacing between adjacent pooling windows). Convolutional neural networks have achieved great performance in image classification targets [7]. Given input images and corresponding labels, CNN is learned to produce hierarchical representations of data. CNN can impose local connectivity on the raw input value. 1D input data includes signal, 2D input data includes image and voice cepstrum and 3D input data includes tomography and video. The table shows how CNN evolves in recent years.

| Year | Conference | Model |
|---|---|---|
| 2012 | NIPS | AlexNet |
| 2014 | ICLR | NIN |
| 2015 | ICLR | VGGnet |
| 2015 | CVPR | GoogleNet |
| 2015 | ICML | BN |
| 2016 | CVPR | InceptionV23 |
| 2016 | CVPR | ResNet |
| 2016 | ECCV | ResNet1001 |
| 2016 | BMVC | WRNs |
| 2017 | CVPR | ShuffleNet |
| 2017 | AAAI | Inception V4 |
| 2017 | CVPR | DenseNet |
| 2017 | CVPR | SENet |
| 2017 | CVPR | DRN |

The list is growing. Till now, CNN is still the state-of-the-art method in many realms.

*Table 6: List of Different Models.*

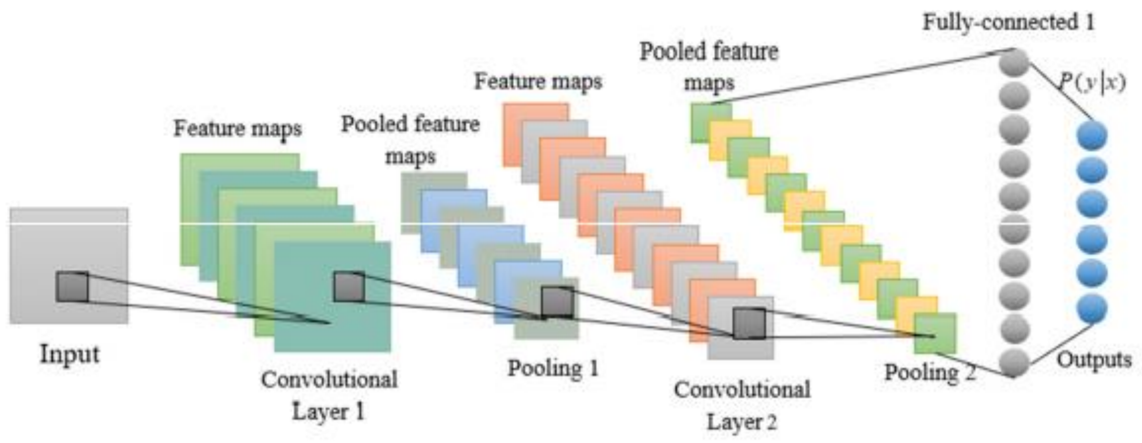


*Figure19:: Standard Convalutional Neural Network Architecture Generally.*

## 3.1.Convolution Layers :

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

### 3.1.1. Convolutional Layer :

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

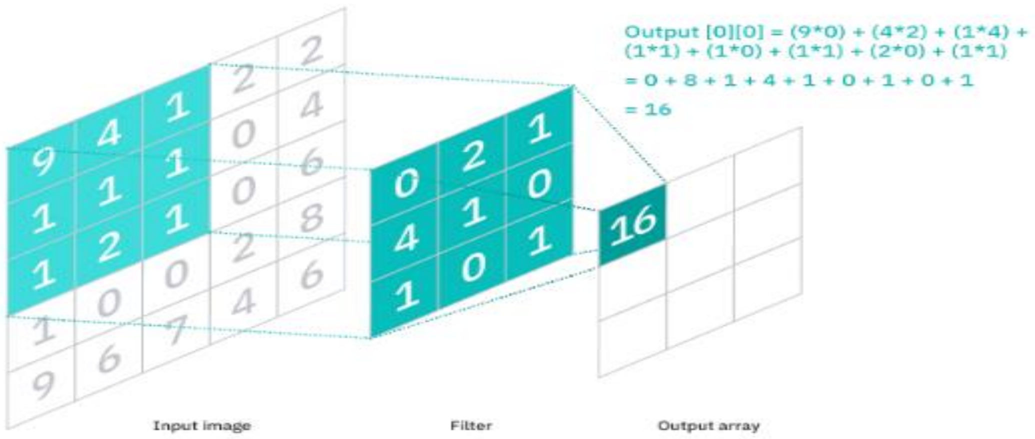

Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output array

*Figure20: Example of Convolutional Layer .*

### 3.1.2. Pooling Layer :

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

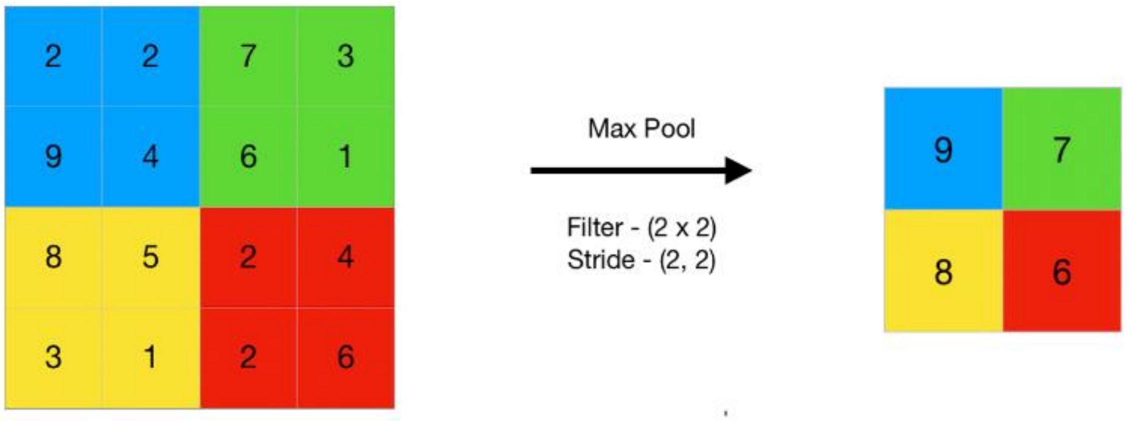


*Figure21: Example of max Pooling.*

### 3.1.3. Fully Connected Layer :

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

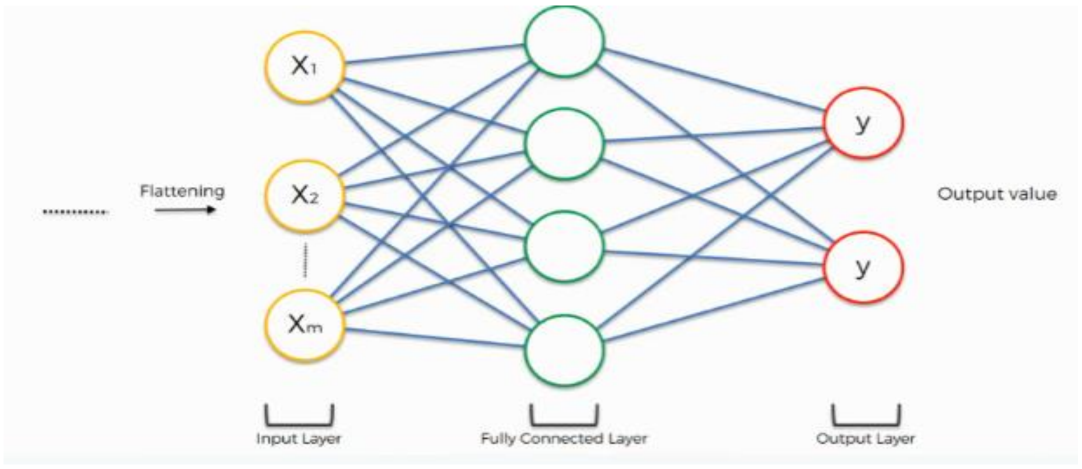


*Figure22: Example of Fully Connected Layer .*

### 3.1.4. Dropout :

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model’s performance when used on a new data.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

### 3.1.5. Activation Functions :

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used.

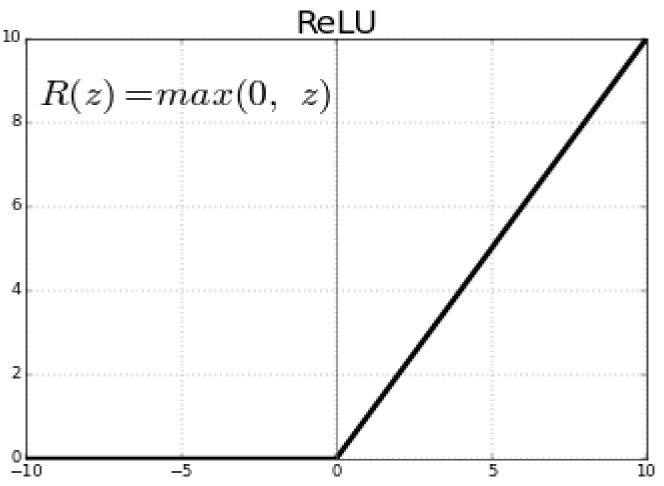


*Figure23: Example of max Activation Function.*

## 3.2.CNN Architectures: (VGG, GoogLeNet, ResNet)

### 3.2.1GoogLeNet/Inception(2014) :

The winner of the ILSVRC 2014 competition was GoogLeNet(a.k.a. Inception V1) from Google. It achieved a top-5 error rate of 6.67%! This was very close to human level performance which the organisers of the challenge were now forced to evaluate. As it turns out, this was actually rather hard to do and required some human training in order to beat GoogLeNets accuracy. After a few days of training, the human expert (Andrej Karpathy) was able to achieve a top-5 error rate of 5.1%(single model) and 3.6%(ensemble). The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. It used batch normalization, image distortions and RMSprop. This module is based on several very small convolutions in order to drastically reduce the number of parameters. Their architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million [12].

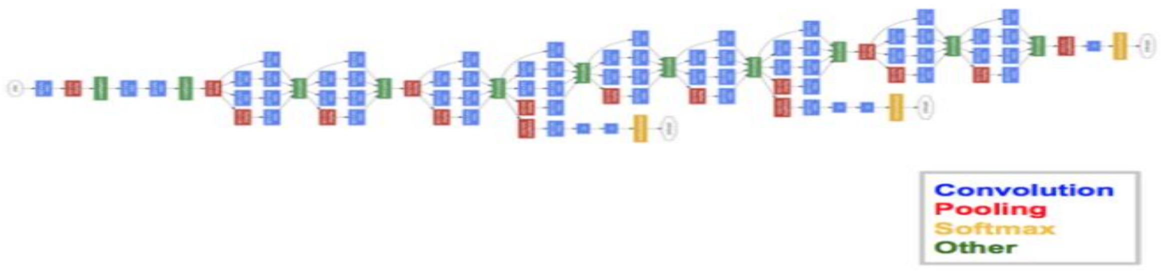


*Figure24: Example GoogleNet.*

### 3.2.2.VGGNet (2014) :

The runner-up at the ILSVRC 2014 competition is dubbed VGGNet by the community and was developed by Simonyan and Zisserman. VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture. Similar to AlexNet, only 3x3 convolutions, but lots of filters. Trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. However, VGGNet consists of 138 million parameters, which can be a bit challenging to handle [12].

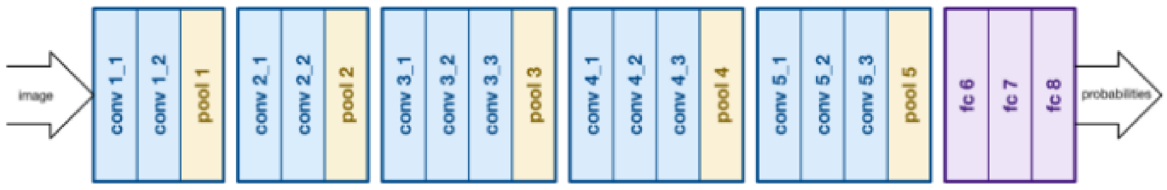


***Figure25: Example VGGNet.***

### 3.2.1.ResNet(2015):

At last, at the ILSVRC 2015, the so-called Residual Neural Network (ResNet) by Kaiming He et al introduced anovel architecture with “skip connections” and features heavy batch normalization. Such skip connections are also known as gated units or gated recurrent units and have a strong similarity to recent successful elements applied in RNNs. Thanks to this technique they were able to train a NN with 152 layers while still having lower complexity than VGGNet. It achieves a top-5 error rate of 3.57% which beats human-level performance on this dataset [12].

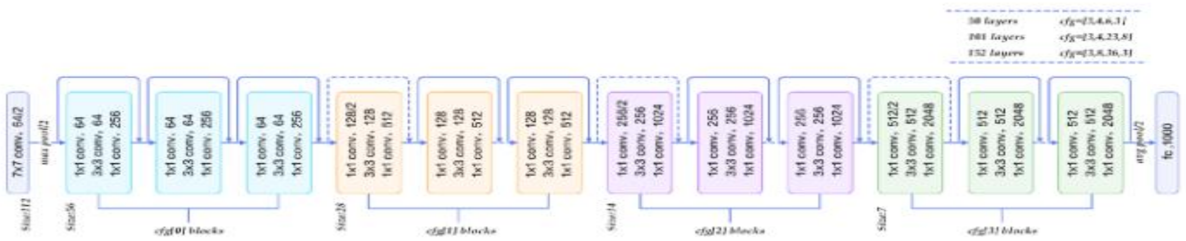


***Figure26: Example ResNet.***

## 4. Conclusion:

In this chapter, we have reviewed the Hog and CoHog variants that we considered for our application we hope that CoHog can be applied to other recognition problems. possible improvements are the application of some feature selection techniques to choose the best co-occurrence orientation from co-occurrence matrix; a sparse representation for CoHog may improve results.

# CHAPTER 4: EXPERIMENTAL RESULTS

## 1- Introduction:

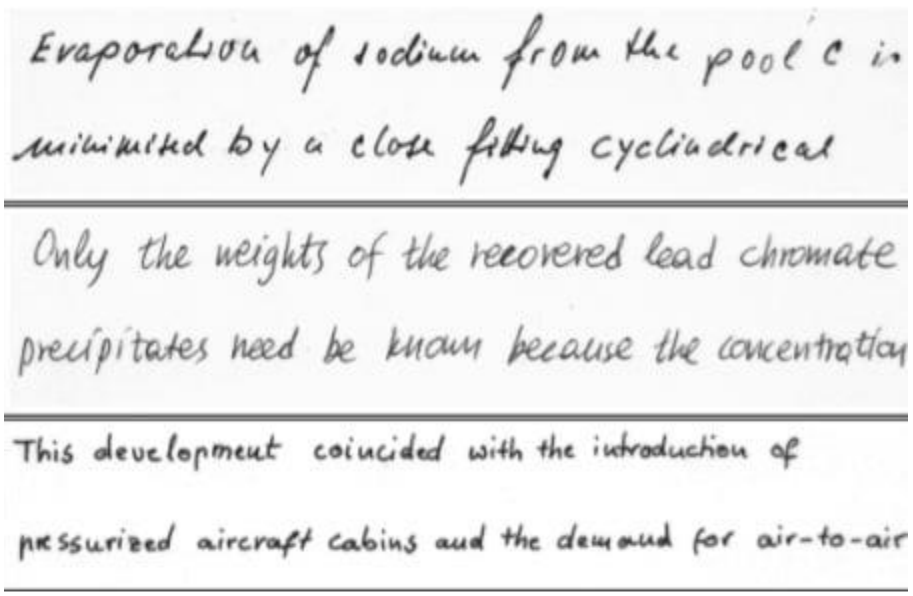
This chapter is to report our findings. We conduct experiments that measure the proposed method from different aspects. We start by introducing the dataset on which experiments are carried out, then, we present the performance metrics. Afterwards, experimental results are reported.

## 2- Experimental protocol:

### Dataset presentation:

The IAM Handwriting Database was first published at the ICDAR 1999 and is described in [50]. It contains 1539 pages of handwritten text from 657 writers who provide a different amount of pages. They also provide labeled text lines and labeled words using automatic segmentation and are manually verified, for solving text recognition tasks, which is not our case. Sample images from three different writers can be seen in Figure 6.2.[7]

The number of pages of each writer ranges from 1 to 66 pages in the original database.



Evaporation of sodium from the pool c is
minimized by a close fitting cyclindrical

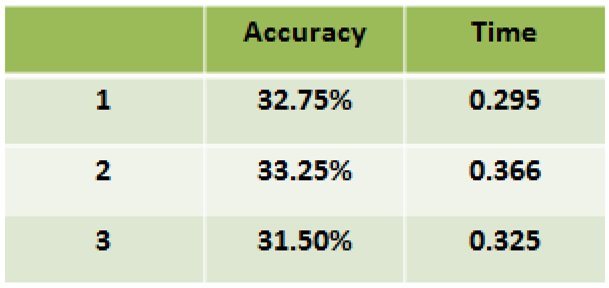
Only the weights of the recovered lead chromate
precipitates need be known because the concentration

This development coincided with the introduction of
pressurized aircraft cabins and the demand for air-to-air

*Figure27 : Three sample images from different writers extracted from the IAM dataset.*

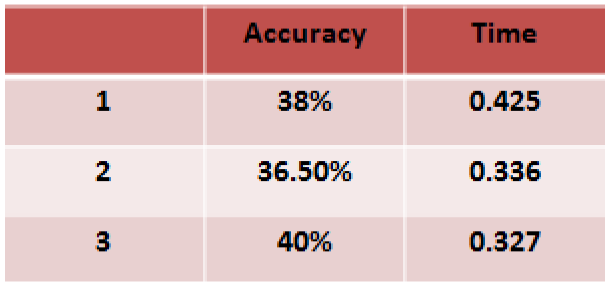
## 3. Experimental Results:

The following three tables represent the ratios and execution time of the three variables Hog, CoHog, and VGG-16.



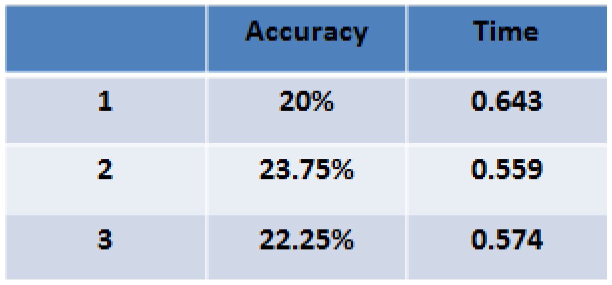
| | Accuracy | Time |
|---|---|---|
| 1 | 32.75% | 0.295 |
| 2 | 33.25% | 0.366 |
| 3 | 31.50% | 0.325 |

***Table 7: Results Hog.***



| | Accuracy | Time |
|---|---|---|
| 1 | 38% | 0.425 |
| 2 | 36.50% | 0.336 |
| 3 | 40% | 0.327 |

***Table 8: Results CoHog.***



| | Accuracy | Time |
|---|---|---|
| 1 | 20% | 0.643 |
| 2 | 23.75% | 0.559 |
| 3 | 22.25% | 0.574 |

***Table 9: Results VGG-16.***

In this experiment, we aim to compare the performance of the Hog and CoHog variants. To this end, we use pre-trained deep mesh, which is VGG-16. We freeze weights and take features from different hidden layers each time. The following table presents the results of the experiment.

| | Hog | Co-Hog | Deep (VGG) |
|---|---|---|---|
| Accuracy | 33.25% | 40% | 23.75% |
| Time | 0.366s | 0.327s | 0.559s |

*Table 10:Max Results Three Method.*

After executing each of Hog, CoHog and VGG16 we get the following results, where we notice that I CoHog gives better results than Hog and CoHog.

**4- Conclusion:**

In this chapter, we give about the experimental results of the experiments (hog, cohog, vgg16) that were conducted. Performed on the IAM dataset. We also studied the effect of distance on System performance. In the end, the handcrafted features are compared to the deep ones.

# GENERAL CONCLUSION:

Automatic writer identification is very intriguing research problem in the field of document analysis and handwriting recognition. The effective implementation of writer identification systems can be applicable in forensic and historical analysis, banks, check processing, signature analysis, graphology, legal documents, ancient manuscripts, digital rights administration, and document analysis methods. The objective of our study is to explore the visual patterns for automatic writer identification from unconstrained offline scanned text-lines images of handwriting.

we proposed a high-dimensional feature descriptor “Co-occurrence histograms of oriented gradients (CoHOG)” for pedestrian detection. Our feature descriptor uses pairs of gradient orientations as units, from which it builds histograms. Since the building blocks have an extensive vocabulary, our feature descriptor can express local and global shapes in detail. We compared the classification performance of our method and several previous methods on two famous datasets. The experimental results show that the performance of our method is better than that of the state-of-the-art methods or at least comparable, and consistently good on both datasets. Future work involves applying the proposed feature descriptor to other applications.

# REFERENCES

**[1]**Wasseem Nahy Ibrahem, Image Processing.

**[2]**EdgeDetection.*https://www.cs.auckland.ac.nz/compsci373s1c/PatricesLectures/Edge%20detection-Sobel_2up.pdf*

**[3**] Matevž Kunaver, Jurij Tasic, Image feature extraction - an overview February 2005 IEEE Xplore.

[**4**]Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor AISHWARYASINGH, SEPTEMBER4,2019.

**[5]** Tomoki Watanabe, Satoshi Ito, and Kentaro Yokoi. Co-occurrence Histograms of Oriented Gradients for Pedestrian Detection.

**[6]**Renu Khandelwa**l**, Convolutional Neural Network(CNN) Simplified.

**[7]** IBM Research Blog ,This is an archive from 2008 to February 2021**.**

**[8]** A. K. Jain, Arun Ross, Salil Prabhakar,b, '' An Introduction to Biometric Recognition '', January 2004

**[9] Anil K. Jain**, **Ruud Bolle, Sharath Pankanti**., "BIOMETRICS Personal Identification in Networked Society", Kluwer Academic Publishers New York, Boston, Dordrecht, London, Moscow, 2002.

**[10] PJames_L._Wayman,_Anil_K._Jain,_Davide_Maltoni**,_Da_Biometric Systems **James_L._Wayman,_Anil_K._Jain,_Davide_Maltoni**,_Da_Biometric Systems.

**[11]**AIADI Khaled**,** Deep VS handcrafted features for writer identification 2019/2020.

**[12}**Siddharth Das**,** CNN Architectures.

[13] Sheng He, Lambert Schomaker, Senior Writer Identification usingDeep Fragment Networks, IEEE

[14] Akshay Punjabi Punjabi, Writer identification in handwritten text images using deep neural networks, Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de València.