PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of higher education and scientific research

Ecole Nationale Polytechnique

Electronic department

Final year project

For obtaining the state engineer diploma in Electronics

Hybrid features fusion for writer identification using single handwritten words

Rayane KADEM

Yacine RABEHI

Under the supervision of

Dr. Nesrine BOUADJENEK, MCB at Ecole Nationale Polytechnique

Presented on July 7th, 2020

Composition of the Jury:

President	Pr.	Mourad ADNANE	Professor	ENP
Supervisor	Mme.	Nesrine BOUADJENEK	MCB	ENP
Examiner	Mr.	Sid-Ahmed BERRANI	MAB	ENP

ENP 2020

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of higher education and scientific research

Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات

Ecole Nationale Polytechnique

Electronic department

Final year project

For obtaining the state engineer diploma in Electronics

Hybrid features fusion for writer identification using single handwritten words

---

Rayane KADEM

Yacine RABEHI

Under the supervision of

Dr. Nesrine BOUADJENEK, MCB at Ecole Nationale Polytechnique

Presented on July 7th, 2020

Composition of the Jury:

President	Pr.	Mourad ADNANE	Professor	ENP
Supervisor	Mme.	Nesrine BOUADJENEK	MCB	ENP
Examiner	Mr.	Sid-Ahmed BERRANI	MAB	ENP

ENP 2020

ملخص

ثبت أن الكتابة اليدوية كجزء من القياسات الحيوية السلوكية لديها القدرة على التمييز بشكل كافٌ بين أي شخصين. لذلك ، في هذا العمل ، نقترح نظاماًا للتعرف على الكاتب باستخدام كلمات فردية مكتوبة بخط اليد. في هذا الصدد ، نقترح ، مقترناًا بمصنف آلة دعم السهم الموجه (SVM) ، دمج الميزات الهجينة التي تجمع بين الميزات المستخرجة من واصف جديد وهو التدرج المحلي المتعدد و الموجه (MLOG) والميزات التي تم إنشاؤها من الشبكة العصبية التلافيفية (CNN) VGG-16. تم تناول طريقتين معروفتين لتحديد الكاتب: الكاتب المعتمد والكاتب المستقل. أظهرت التجارب التي أجريت على مجموعتي بيانات قياسيتين نتائج مرضية وواعدة للغاية.
الكلمات المفتاحية: تعريف الكاتب ، الكتابة اليدوية ، MLOG ، الميزات الهجينة ، CNN.

Résumé

L'écriture manuscrite étant une caractéristique biométrique comportementale, a prouvé sa capacité à suffisamment pouvoir différencier deux individus. Par conséquent, dans ce travail, nous proposons un système d'identification d'écrivain à partir de documents manuscrits contenant un seul mot. À cet égard, nous proposons, associé au classifieur Machine à Vecteurs de Support (SVM), une fusion de caractéristiques hybrides, combinant des caractéristiques extraites d'un nouveau descripteur, à savoir le Gradient Local Multi-échelle Orienté (MLOG) et des caractéristiques générées à partir du réseau de neurones convolutif VGG-16. Deux approches connues d'identification des auteurs ont été abordées ; dépendant de l'auteur et indépendant de l'auteur. Les expériences menées sur deux ensembles de données standards ont montré des résultats satisfaisants et très prometteurs.
Mots-clés: identification de l'écrivain, écriture manuscrite, MLOG, caractéristiques hybrides, CNN.

Abstract

Handwriting as a part of behavioral biometrics has been proved to having the ability to sufficiently differentiate any two individuals. Therefore, in this work, we propose a system for writer identification using single handwritten words. In this regard, we propose, associated to Support Vector Machine (SVM) classifier, a hybrid features fusion that combined features extracted from a new descriptor namely, Multiscale Local Oriented Gradient (MLOG) and features generated from Convolutional Neural Network VGG-16. Two known approaches of writer identification were addressed: writer-dependent and writer-independent. Experiments conducted on two standard datasets, showed satisfying and very promising results.
Keywords : writer identification, handwriting, MLOG, hybrid features, CNN.

Acknowledgements

We would like to express our sincere gratitude to our supervisor Dr. Nesrine BOUADJENEK, who supported us all along this work. We thank her for her patience, motivation and unparalleled dedication to make this work a valuable one. She was always available whenever we needed her throughout this work. We could not have hoped for a better supervisor and mentor. We owe her immense and infinite gratitude.

We especially thank the members of the jury who despite their busy schedules, agreed to spare their time to examine this work: Mr. Mourad ADNANE, Professor at Ecole Nationale Polytechnique and Mr. Sid-Ahmed BERRANI, Doctor at Ecole Nationale Polytechnique.

We would like to direct our thanks and gratitude to all our instructors at Ecole Nationale Polytechnique, who guided us through our five years at the school.

We also thank all our colleagues at Ecole Nationale Polytechnique, whom we shared with precious memories and learned from during our years together.

Finally, We thank everyone that helped us in any form throughout our academic curriculum.

Dedications

I dedicate this work to my dear parents. All the words in the world can not express my endless gratitude for all the efforts and sacrifices you have made for me throughout my life.

To my siblings Lamia, Fouad and Mohamed who encouraged me through the years to be the best version of myself.

To my life mentor, my precious sister Cherifa who was always there for me and transmitted to my person her deep love for knowledge.

To my adorable nieces and nephews.

To my work partner Yacine, with whom I shared three years of studies, complicity and souvenirs. I thank you for your unique motivation and determination to make this work valuable.

To all those who are dear to me and who carry me in their hearts . . .

Rayane

Dedications

I dedicate this work to my parents who supported me along my life and spent immense efforts in my education: my father Naceur and my mother KAOULA Louiza.

To my sister Yasmine and my brother Amine whom I owe infinite gratitude for their relentless efforts in helping me in every single aspect of life.

To my sister in law Amel Yousnadj who helped me greatly and shared her knowledge with me.

To my work partner Rayane for sharing this work and all its hardships, ups and downs with me. I could not wish for a better work partner.

Yacine

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence.
CNN	Convolutional Neural Networks.
CPU	Central Processing Unit.
DNN	Deep Neural Networks.
ELU	Exponential Linear Unit.
FC	Fully Connected.
GLBP	Gradient Local Binary Paterns.
GMM	Gaussian mixture models.
GPU	Graphic Processor Unit.
HMM	Hidden Markov models.
KNN	K-Nearest Neighbors.
LBP	Local Binary Paterns.
LDGP	Local Directional Gradient Pattern.
MLOG	Multiscale Local Oriented Gradient.
MLP	Multi-Layer Perceptron.
RBF	Radial Basis Function.
ReLU	REctified Linear Unit.
RGB	Red Green Blue.
SOM	Self-Organizing Map.
SVM	Support Vector Machine.

Introduction

Since the beginning of human history, people have identified themselves by relying on specific characteristics such as faces or voices. Due to the industrial revolution, the world grew rapidly, where it was necessary to identify people and control access. Thus, was born the concept of a biometric system referring to the use of metrics related to the human characteristics as a personal identifier. Automated biometric systems have only become available in recent decades, due to significant advances in the field of computer processing. Recently, it has become extended to more specific fields such as the analysis of handwriting.

Researches have proved that handwriting is a biometric characteristic, just like DNA and fingerprints. It is an extension of the personality and a sign of individuality between people, which allows it to be used to distinguish and differentiate individuals. Indeed, although DNA and fingerprints are more reliable for person identification, handwriting remains a strong behavioral biometric, that does not require specialized hardware devices to collect handwriting samples, unlike DNA and fingerprints.

Writer identification refers to a document identification technology that determines the writer's identity through handwriting text analysis [1]. In the last few decades and with the technological development, the field of writer identification based on handwritten documents became a challenging topic for various researchers around the world due to the increase of its utilization. Among the applications where automatic writer identification system is found necessary, we note:

– Forensics and criminal investigations to determine the identity of alleged authors of handwritten documents such as wills, ransom notes, suicide letters, legal documents, etc.

– Authentication and classification of historical documents to preserve the cultural heritage from falsified documents [2].

– Monitor and regulate the access to certain confidential sites or data where large amounts of documents, forms, notes and meeting minutes are constantly being processed and managed, knowing the identity of the writer would provide an additional value [3].

– Checking authenticity of bank cheque.

Compared to electronic documents, handwriting holds discriminating information about the person who produced it. Indeed, the writing contours, corners, junction, and thickness shape are specific to each individual. Besides, a person will not produce exactly the same handwriting twice [4]. Therefore, a writer identification system, aims to extract characteristics, referred to as features, which define the handwriting style of the writer. These features should be able to express the indirect information of writer identity contained in a handwritten text. Consequently, due to the difficulty of characterizing the writing, the reliability of such a system depends strongly on the quality of the extracted features.

The literature reports only studies that addressed writer identification from a whole text document or lines. In this work, we propose to investigate writer identification from single handwritten words, which is a much more complicated and challenging task, since a word contains only very few information. As a solution to this particular problem, we come up with a hybrid features fusion associated to Support Vector Machine (SVM) classifier. Precisely, we combine features extracted from a brand new descriptor based on a Multiscale Local Oriented Gradient (MLOG) and features extracted from a pre-trained Convolutional Neural Network (CNN), namely, VGG16 (Visual Geometry Group) model, to form powerful features that characterize at best the handwriting style contained in a word. Experiments are conducted on two benchmark datasets and findings are very promising.

The dissertation is divided into four chapters that are structured as follows:

- Chapter 1 details the problematic and gives an overview on writer identification architecture by presenting writer-dependent and writer-independent approaches. It also gives a brief summary of the state of the art.

- Chapter 2 elaborates on some of the feature extraction and classification methods that proved efficiency in pattern recognition tasks. Especially, it introduces the proposed Multiscale Local Oriented Gradient (MLOG) descriptor.

- Chapter 3 gives a profound overview and description of Convolutional Neural Networks (CNN) and the predefined used architecture VGG16.

- Chapter 4 presents and discusses the obtained results for writer identification using single handwritten words, namely, results of descriptor-based systems and CNN-based system. Then after, the chapter exposes the findings of hybrid features for both writer-dependent and writer-independent approaches.

Finally, we achieve this dissertation with a conclusion and some perspectives for a future work.

Chapter 1

Overview on writer identification from handwriting

1.1 Introduction

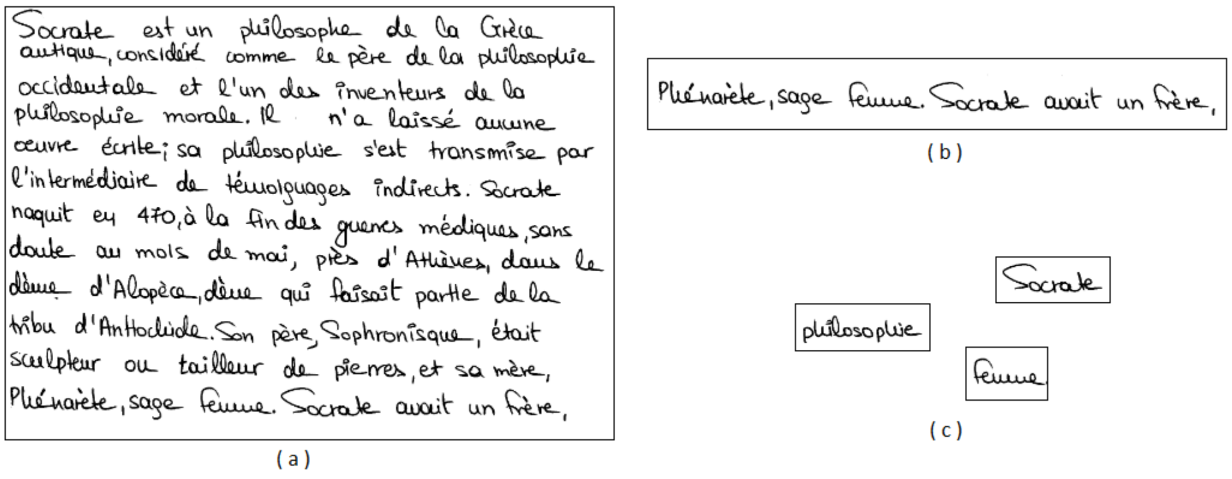
For many years, handwriting has been used as a behavioral biometric to identify individuals for different purposes. In 1989, Plamondon et al., published the first survey on automatic signature verification and writer identification [5], which proves that automatic writer identification started several decades ago. Since then, this discipline has received significant interest by the research community and various methods have been proposed. A typical writer identification system operates in two modes: offline and online. In the online mode, the writing behavior is captured from the writer in real-time through a writing tablet, which converts it into a list of signals and directions. On the other hand, for offline mode, scanned and digitized handwriting images are used to identify the writer.

In this chapter, we give an overview of the writer identification system by first, addressing the problematic tackled in this work. Secondly, we expose the handwriting as a behavioral biometrics. After that, we present the architectures of writer identification systems and finally, we conclude by presenting a summary of the state of the art to put the light on and build an idea on previously used techniques.

1.2 Problematic

Writer identity is an implicit (indirect) information of a handwritten document. There are many types of data used to identify the writer: texts, rows, and words as shown in Figure 1.1. The commonly used type of data is text blocks. This is due to the considerable amount of information it contains.

Nowadays, due to the digital revolution, handwriting is an increasingly rare activity, which requires a new approach to be able to identify the writer based on a very small amount of available text, which may be as little as a single word [6]. In this work, we address the writer identification problem based on single-word images, which is a challenging problem because the information contained in a single word is highly limited for modelling the writing style of an author. In order to solve this problem, we present different approaches of feature extraction and fusion, forming a solid ground for writing style discrimination and therefore writer identification.


Socrate est un philosophe de la Grèce
antique, considéré comme le père de la philosophie
occidentale et l'un des inventeurs de la
philosophie morale. Il n'a laissé aucune
oeuvre écrite; sa philosophie s'est transmise par
l'intermédiaire de témoignages indirects. Socrate
naquit en 470, à la fin des guerres médiques, sans
doute au mois de mai, près d'Athènes, dans le
dème d'Alopèce, dème qui faisait partie de la
tribu d'Antiochide. Son père, Sophronisque, était
sculpteur ou tailleur de pierres, et sa mère,
Phénarète, sage femme. Socrate avait un frère,

(a)

Phénarète, sage femme. Socrate avait un frère,

(b)

Socrate

philosophie

femme

(c)

Figure 1.1: Example of handwritten document types from ICDAR2011 dataset. (a): text,(b): row,(c): words

1.3 Handwriting as a behavioral biometric

In computer security, biometrics refer to identification and authentication tools that rely on measurable biological characteristics of the human body that can be automatically checked. As shown in Figure 1.2, biometric characteristics are generally divided into two categories; physiological and behavioral. Physiological characteristics include physiological properties of the human body such as iris, face, retina, DNA, and fingerprints. While behavioral characteristics gather the personal features developed by the behavior of the individual like voice, handwriting, gait and handwritten signature. However, fingerprint and face biometrics remain the most established and used biometrics for identification and verification.

Nevertheless, handwriting is universal and a daily used skill. The effectiveness of handwriting as a behavioral biometric has been debated by innumerable researchers [7]. Jain et al., [8] have identified seven factors determining the appropriateness of a biometric modality. In terms of “distinctiveness”, handwriting appears to be relatively less effective compared to DNA or fingerprints. Whereas in terms of “measurability”, acquiring handwriting samples is quite simple as it does not require specialized hardware devices. Therefore, to a certain extent, and due to its recent development and performance, handwriting has become a strong behavioral trait of identification, as powerful as DNA and fingerprints [9].

Indeed, each individual has his own particular writing style; people might be able to copy it, but never write it in an identical way. Even identical twins who share appearance and genetics do not have the same handwriting. In 2002, Srihari et al., [10] confirmed

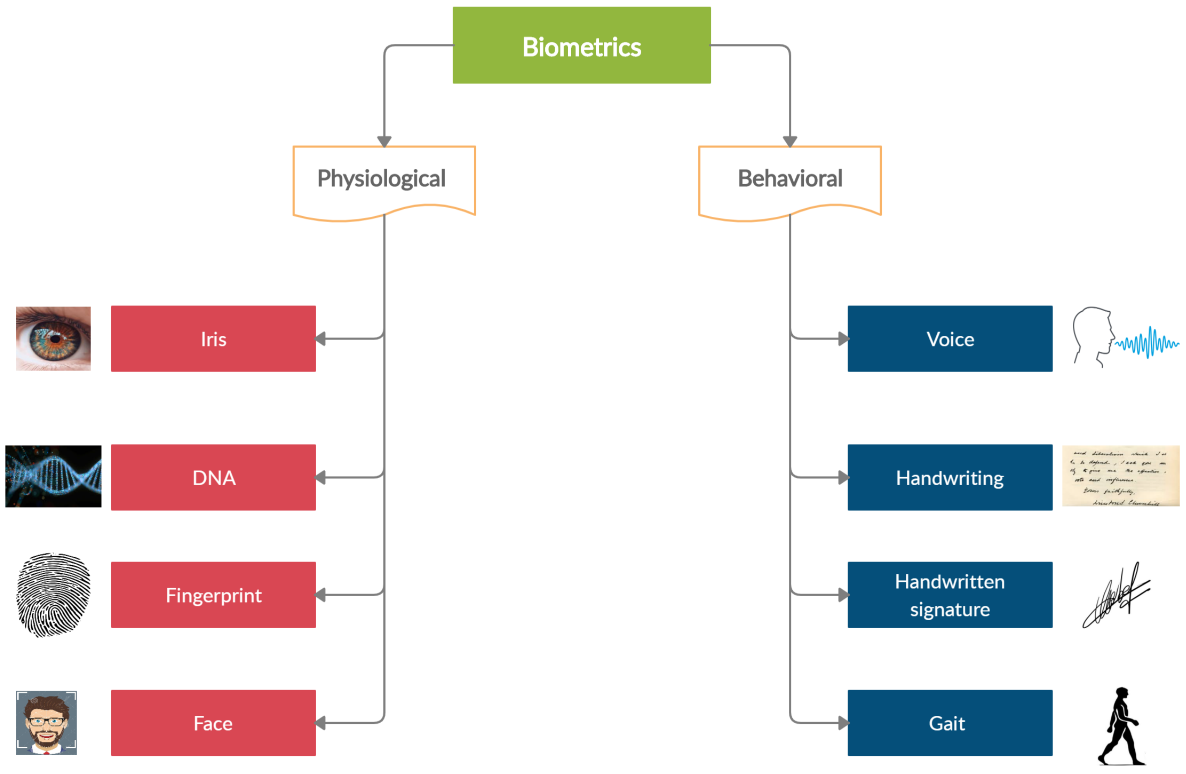


Figure 1.2: Biometric characteristics

the hypothesis that everyone writes differently. Each individual has a handwriting style that is distinct from the handwriting style of another individual as shown in Figure 1.3. Handwriting individuality lies in the specific shapes of letters like their roundness or sharpness, the spacing between words, the slope and average size of the letters, the pen pressure on the paper, etc.

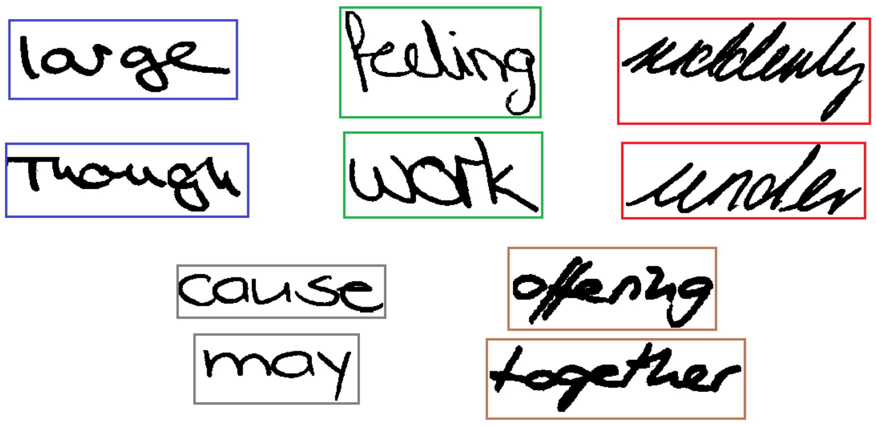


Figure 1.3: Examples of handwriting style for 5 different writers from IAM dataset

1.4 Architecture of a writer identification system

As any classification system, writer identification systems consist essentially of three main steps; namely, pre-processing, feature extraction, and classification, to build the model and therefore, decide on the authorship of an unknown handwritten word as depicted in Figure 1.4.

Writer identification techniques reported in the literature allow this task to be addressed through two approaches called writer dependent and writer independent approaches.

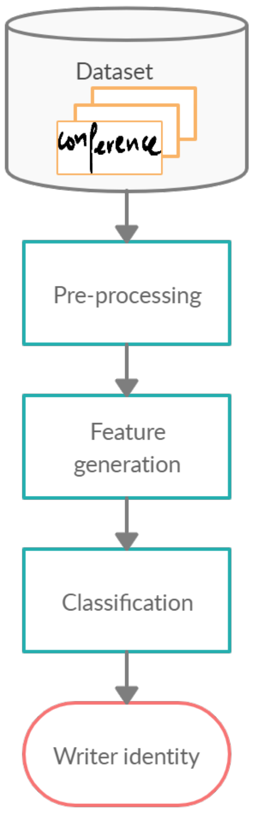


Figure 1.4: Architecture of writer identification system

1.4.1 Preprocessing

Preprocessing is an important step in writer identification systems. The aim of this step is to improve the readability of the text image and remove details that have no discriminatory power in the identification process. Recall that, documents are often digitized using a scanner device. Thus, the quality of the scanned document is influenced by several factors, namely, the performance of the scanner and the resolution (which is the number of pixels per inch). The preprocessing step usually includes several operations such as binarization, filtering and contrast enhancement. In this section, we only present the operations used in our work, which are binarization, segmentation and normalization.

- **Binarization**
  Binarization aims to isolate the writing from the background of the document. Applying this technique to a grayscale image gives a binary image where each pixel can have a value of 0 or 1.
  The process of binarization works by finding a threshold value in the image histogram, which separates between 2 entities, each representing one of two elements or an element and its background. The binarization algorithms are divided into two main categories called global binarization and local binarization. The former uses a unique threshold for the entire image, while the latter calculates local threshold values, pixel by pixel and region by region [11]. Within global binarization techniques, there exist several algorithms to find the optimum threshold value, but none of them can be referred to as the best one, as each method is optimal for certain applications. In our work, the algorithm that proved efficiency was the so called Otsu's binarization [12]. Through an iteration over all possible threshold values, it measures the spread of the pixel levels on both sides (foreground and background) of the considered threshold value. Then, it searches for the threshold value that leads to the minimum sum value of the foreground and background spreads [11].
  This binarization step allows to have a low memory space, and therefore, increases the processing speed and simplify feature extraction. Figure 1.5 illustrates a binarization example of an IAM word image using Otsu method.

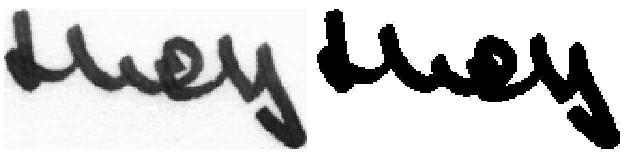


Figure 1.5: Image binarization using Otsu method

- **Segmentation**
  Segmentation is the operation that aims to extract the targeted entities of the text such as lines, words or fragments. In this work, word extraction is done by applying vertical and horizontal projection histograms. Vertical projection consists of summing the number of text pixels column by column. It delimits the first and last pixel of the text vertically. On the other hand, horizontal projection consists of summing the number of text pixels along the lines in order to detect the first and the last pixel of the text horizontally. Figure 1.6 illustrates this word extraction technique from a text.

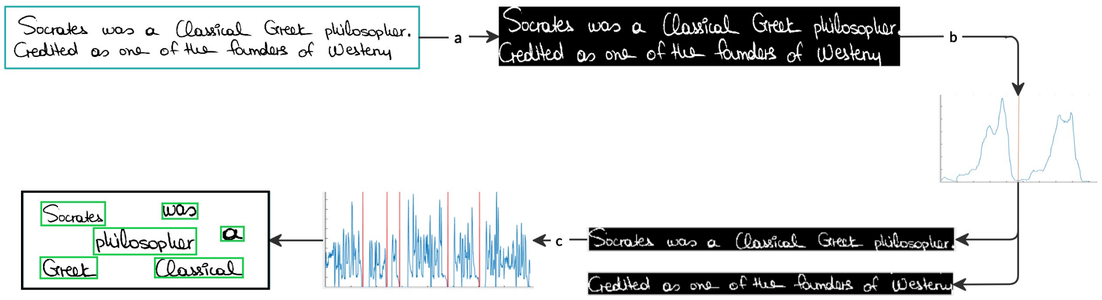


Figure 1.6: Words extraction. (a) Complement of the original image, (b) Horizontal projection, (c) Vertical projection

- **Normalization of the image size**

  After the segmentation step, word images systematically have variable sizes. However, certain methods of feature extraction provide feature vectors with a size that is proportional to that of the input image. Moreover, classifiers such as neural networks and Support Vector Machines (SVM) require data of the same size. Specially, size normalisation is essential to deep leaning neural networks. Hence, in such situations, a standardization process to a unique size is necessary before proceeding to feature generation. Normalization consists in resampling the image by decreasing or increasing the number of pixels according to the new targeted size, which can be larger or smaller than the original image size. Figure 1.7 shows an example of size normalization of a handwritten word.

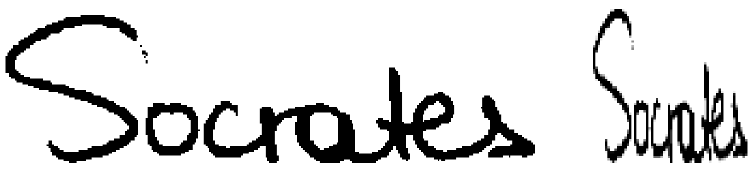


Figure 1.7: Example of size normalization of a handwritten word. (a) Original image, (b) Resized image 64x64 pixels

1.4.2 Feature extraction

Feature extraction or generation involves reducing the number of resources required to describe a large set of data. When performing analysis of complex data, one of the major

problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause the classification algorithm to overfit to training samples and generalize poorly to new ones (testing samples). Feature extraction aims to construct combinations of the variables to get around these problems while keeping an accurate description of the data. Many image processing practitioners believe that properly optimized feature extraction is the key to effective model construction [13].
Several feature extraction methods have been designed for writer identification in the last few decades. They can be divided into two main categories: texture-based and grapheme-based features [6].

Texture-based methods attempt to characterize repeating patterns of local variations in an image intensity. Statistical methods are widely used for achieving this purpose. Many descriptors are employed in this context for writer identification, such as Local Binary Patterns (LBP) [14, 15], Local Phase Quantization (LPQ) [14] and the run-length of LBP [16]. Filter-based features, such as Gabor [17] and oriented Basic Image Feature Columns (oBIF Columns) [18], have also been employed. Some other descriptors extract features from the contours of the ink trace, such as Hinge-based features [19, 20] which extract the properties of slant, curvature information and stroke width.

On the other hand, grapheme-based techniques aim to divide the ink on a set of pixels to multiple parts in order to segment the word into units of characters, or parts of characters [21]. These units are mapped on a common space called a codebook. Graphemes have been widely used in feature extraction methods dedicated to writer identification and verification [22, 23].

1.4.3 Classification

Classification consists of assigning a class label to a set of unclassified instances. In the case of supervised classification, where a set of labeled data is provided, the aim of classification is to analyze input data in order to develop an accurate description or model for each class, using a specific learning algorithm. In fact, the learning algorithm approximates a target function by mapping input variables to output variables according to some decision rule. It is what defines a classifier. The literature of writer identification reports the use of several classifiers such as k-Nearest Neighbors (KNN) [24], Gaussian Mixture Model (GMM) [25], Hidden Markov Model (HMM) [26], Support Vector Machine [27]. Nowadays, the trend is to use deep leaning for writer identification [6].

1.5 Writer-dependent approach

Writer-dependent approach considers one model per author. It is basically a classification problem of N classes where N is the number of writers. More precisely, it consists of creating a reference model for each writer, generated through his collected samples, whereas the queried sample is compared to each sample of each writer during the identification stage. It usually yields good results, but its drawback lies in the fact that a considerable amount of data is necessary to train a reliable system [14]. Besides, if a new writer is added to the dataset, re-training the model is necessary. This kind of approach has been the most commonly used in writer identification related works. However, it showed its limitations as the number of writers increases.

1.6 Writer-independent approach

An alternative to writer-dependent approach is the global approach or writer-independent model. It is based on the forensic questioned document examination approach and classifies the writing, in terms of authenticity, into genuine (positive) and forgery (negative), using for that one global model. This approach is based on a dichotomy transformation that makes it possible to reduce any innumerable classification problem to a 2-class problem [28].

In the case of writer identification, given a queried handwritten word and a reference handwritten word, the aim is to determine whether or not the two words were produced by the same writer. For that, a population of M writers is randomly selected from the dataset, where a set of references per writer is considered. The dichotomy transformation takes place through a dissimilarity measure computed between pairwise of references from the same writer to generate the positive (genuine) class called *within class dissimilarities.* As to the forgery (negative) class, a dissimilarity measure is calculated between each two references of different writers, which gives *between class dissimilarities.* Hence, we obtain a 2-class problem, which can be solved by building a model using a binary classifier.

More precisely, let V_i and U_i be two vectors in the feature space, labelled I_v and I_u respectively. Let D_i= $distance(V_i,U_i)$ be the dissimilarity measure where $distance(s,p)$ can be any of the known mathematical distances between two vectors s and p. Table 1.1 presents some of the commonly used distances. In the dissimilarity space, there are two classes that are independent of the number of writers: the within class (+) and the between class (-). The dissimilarity vector D_i is assigned the label I_D as follows [14]:

Distance function	Equation
Euclidean distance	$D_{euclidean}(a,b) = \left(\sum_{i=1}^{n} \lvert a_i - b_i \rvert^2\right)^{1/2}$
Manhattan distance	$D_{Manhattan}(a,b) = \sum_{i=1}^{n} \lvert a_i - b_i \rvert$
Chi-square distance	$D_{Chi-square}(a,b) = \sum_{i=1}^{n} \frac{(a_i-b_i)^2}{\lvert a_i+b_i \rvert}$
Canberra distance	$D_{Canberra}(a,b) = \sum_{i=1}^{n} \frac{\lvert a_i-b_i \rvert}{\lvert a_i+b_i \rvert}$

Table 1.1: Distance functions used in dissimilarity calculation

$$I_D = \begin{cases} + & if \quad I_q = I_v \\ - & Otherwise \end{cases} \tag{1.1}$$

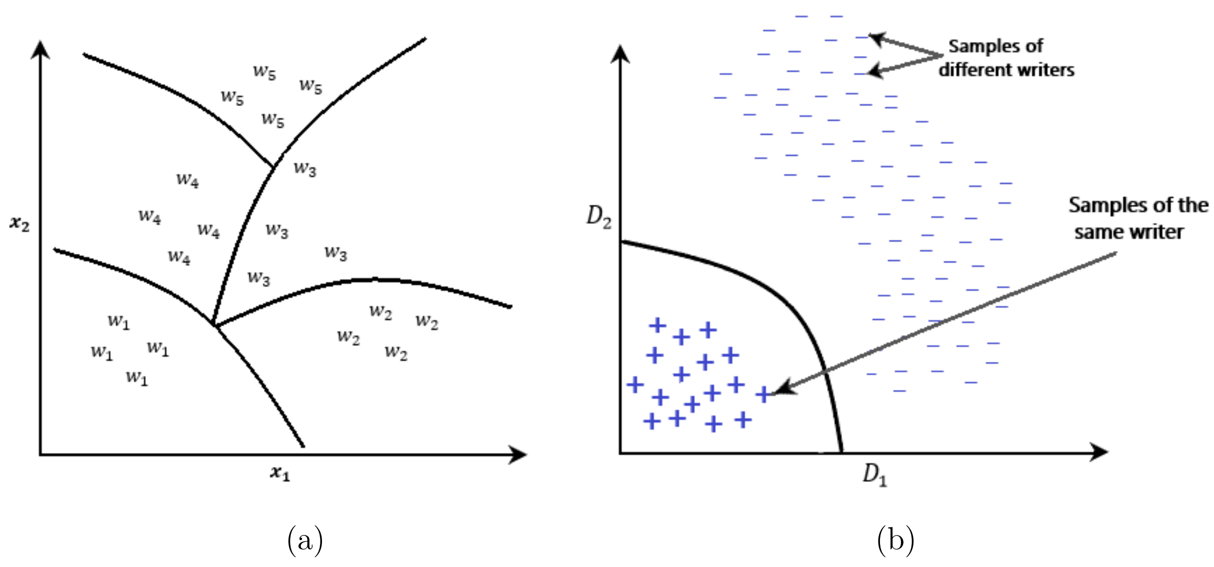


Figure 1.8: Dichotomy transformation: (a): samples in features space and (b): samples in the dissimilarity space

Figure 1.8 illustrates the dissimilarity-based transformation. Suppose there are five writers, $w_1, ..., w_5$, and each one of them provides four reference samples. The feature extraction process extracts a feature vector for each reference sample. Then a distance transformation takes place and computes the dissimilarity measure between the features of each pair of samples for the same writer and for different writers to form the dissimilarity space. As we can see, the distance transformation affects the geometry of the distribution. In the feature space, multiple boundaries are needed to separate all the writers. While, in the dissimilarity space, only one boundary is necessary, since the problem is reduced to a 2-class classification problem [14]. We can also see that, if both samples come from the same writer (genuine), then the dissimilarity measure should be close to 0. On the contrary, if the two samples come from different writers (a forgery), the dissimilarity measure should be far from 0. This is true under favourable conditions. However, the dissimilarity measure can be affected by intra-writer variability, which could generate values that are far from zero even if the dissimilarity is measured between samples produced by the same writer [14]. Note that, for a queried word, dissimilarities are computed with all the references of all the writers. The queried word belongs to the writer with the smallest dissimilarity measure. Figure 1.9 depicts the architecture of the writer-independent system.

The most important advantage of this approach, is that, even writers whose references were not used for training can be identified by the system. This characteristic is quite attractive, since it avoids having to train a new model every time a new writer is introduced.

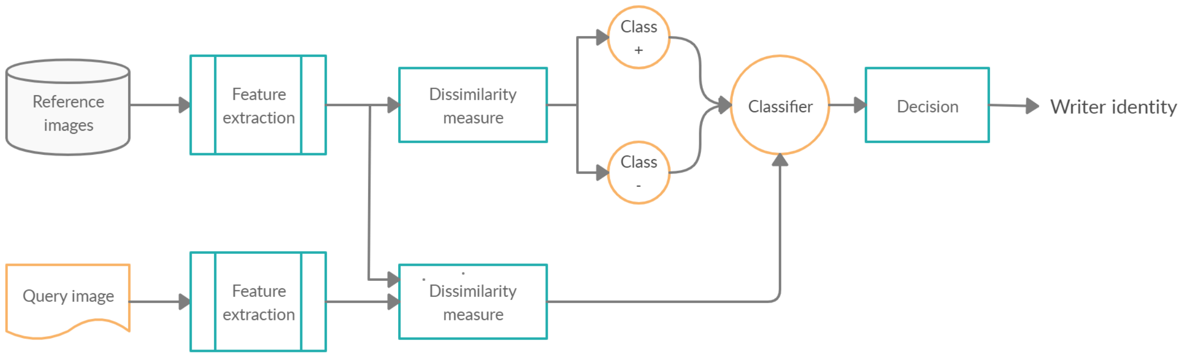


Figure 1.9: Architecture of a writer-independent system

1.7 Summary of the state of the art of writer identification

Up to now, developing a robust writer identification system still attracts many researchers in the pattern recognition community. Over the years, countless papers have been published but have all the particularity of dealing with writer identification based on a large amount of text (whole document), which is never the case in real-life applications. Unfortunately, writer identification based on a single handwritten word has not received enough interest mainly due to the high difficulty that it presents. A word is a very small sample, which makes it a highly challenging task, since the writer style information will be too limited. Indeed, very few authors ventured to use only one word, where the first work was introduced by Tomai et al., in 2004 [29]. The authors consider four different types of features that are gradient-structural & concavity features, word model recognizer features, shape curvature features and shape context features. Experiments were conducted through writer-dependent protocol on a private dataset of more than 1000 writers where the best accuracy reached was of 62%.

In [30], authors address writer verification problem based on the analysis of a unique sample of a handwritten word, which is slightly different from writer identification. The Levenshtein edit distance based on Fisher-Wagner algorithm was used to estimate the cost of transforming one handwritten word into another. In order to apply it to handwritten words, a segmentation module to generate the graphemes was developed. The approach was evaluated on part of the IAM database (100 writers), where half of them provided three samples only of the same word and a performance of 87% was reached.

The most recent and interesting work was published in 2018 by He et al.[6], where the authors presented a deep adaptive learning method for writer identification based on single-word images using multi-task learning. An auxiliary task is added to the training process to enforce the emergence of reusable features. The proposed method transfers the benefits of the learned features of a convolutional neural network from an auxiliary task such as explicit content recognition to the main task of writer identification in a single procedure. Specifically, they proposed a new adaptive convolutional layer to exploit the learned deep features. A multi-task neural network with one or several adaptive convolutional layers is trained end-to-end, to exploit robust generic features for a specific main task, i.e., writer identification. Three auxiliary tasks, corresponding to three explicit attributes of handwritten word images (lexical content, word length and character attributes), were evaluated on two benchmark datasets IAM and CVL. For IAM dataset, 35421 word images from 657 writers were used, the performances reached TOP-1 of 69.5% and TOP-5 of 86.1% using word recognition as an auxiliary task. As to CVL dataset, 99513 word images from 310 writers were used, where the performances reached TOP-1 of 79.1% and TOP-5 of 94.31% using word length recognition as an auxiliary task.

1.8 Conclusion

This chapter presented the individuality of handwriting as a behavioral biometric characteristic, and the challenging problematic of writer identification from a single handwritten word. It provided an overview on the architecture of a writer identification system detailing each of its component from acquisition to identification including writer-dependent and writer-independent approaches. From the analysis of the available state of the art, writer identification from single words can be further investigated in order

to improve performances. Consequently, the aim of this work is to fuse hybrid features for writer identification from a single handwritten word for both writer-dependent and writer-independent protocols.
The next chapter will be dedicated to the description of the adopted methods used for writer identification including our new proposed descriptor.

Chapter 2

Description of adopted methods for writer identification

2.1 Introduction

The development of a writer identification system is essentially based on the characterization of the writing style of the author. The first step to any efficient pattern recognition system is the feature extraction and then comes the classification step. Even if we choose the classifier wisely, without some powerful features, the system will perform poorly. Consequently, since writer identification from a single handwritten word is a highly challenging task, powerful features are necessary to capture the handwriting style from a handwritten word. Therefore, in this work we propose a new descriptor called Multiscale Local Oriented Gradient (MLOG) to characterize the writing style. To demonstrate fairly its performance, we compare it to two feature extraction methods, namely, pixel distribution and Gradient Local Binary Patterns (GLBP), that have proven their efficiency in many applications related to handwriting. Furthermore, this chapter gives an overview on Support Vector Machine classifier, since it was used for writer identification in this work.

2.2 Feature extraction methods

This section presents the feature extraction methods adopted for writer identification in this work, especially the new proposed descriptor Multiscale Local Oriented Gradient (MLOG), in addition to pixel distribution and Gradient Local Binary patterns as benchmark descriptors.

2.2.1 Pixel distribution

Pixel distribution is a topological descriptor that highlights the geometric continuous deformation properties of the handwriting within a given cell [31]. It has been used for signature verification [31] and soft-biometric prediction from handwriting [32]. Pixel distribution is based on four measures where each cell is determined by two projections; vertical and horizontal (see Figure 2.1). Afterwards, the height and width of the stroke are conducted in four directions limited by the mentioned projections [31]. These values are referred to with the letters C_1, C_2, C_3, C_4, where :

- C_1: represents the height of the left part of the stroke.
- C_2: represents width of the upper part of the stroke.
- C_3: represents height of the right part of the stroke.
- C_4: represents width of the lower part of the stroke.

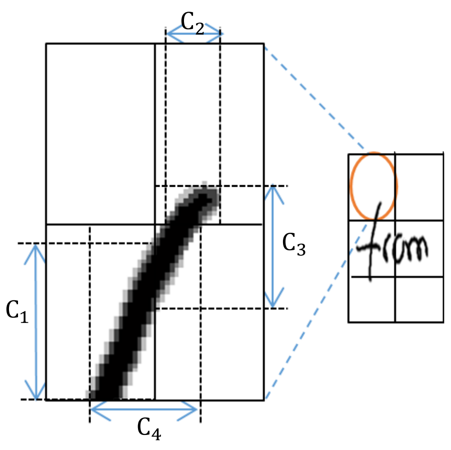


Figure 2.1: Pixel distribution within a cell [32]

2.2.2 Gradient Local Binary Patterns

Gradient Local Binary Patterns (GLBP) was first introduced for human detection as a descriptor that combines gradient information with texture information [33]. It has also been successfully employed in several applications related to handwriting such as the prediction of soft-biometrics from handwriting [32], writer retrieval [34] and word spotting [35].

The principle idea for GLBP calculation consists of exploiting uniform Local Binary Patterns (LBP) to compute the histogram of oriented gradients. The GLBP of a given image is elaborated as follows:

- First, Local Binary Patterns is calculated. LBP is used to perform a statistical and structural analysis of textural patterns on the image. It describes the gray level distribution by comparing the gray level value of a pixel with neighboring pixels. LBP takes 1 if the central pixel has a lower gray level, otherwise it takes 0 [32]. For L neighbors situated on a circle of radius R, the LBP code of a central pixel P is computed as follows:

$$LBP_{L,R}(P) = \sum_{l=0}^{L-1} F\left(g_l - g_p\right) 2^l \tag{2.1}$$

  where

$$F(s) = \begin{cases} 1, & s \geq 0 \\ 0, & s < 0 \end{cases} \tag{2.2}$$

  g_p is the gray value of the central pixel P and g_l is the gray value of the l^{th} neighbor.

- Secondly, neighbor pixels with the same binary value stick together to get several “1” area and “0” area, so that only uniform patterns are considered. Note that uniform patterns correspond to codes that contain two transitions from 1 to 0 (or 0 to 1) [32].

- Thirdly, the size of the GLBP matrix is defined by all possible angle and width values. Precisely, there are eight possible Freeman directions or angle values, while the number of "1" in the uniform patterns can vary from 1 to 7. This yields a 7x8 GLBP matrix in which gradient values are accumulated.
- At last, the GLBP matrix constituting the histogram is exploded into a vector of 56 elements. The flow of calculation is shown in Figure 2.2.

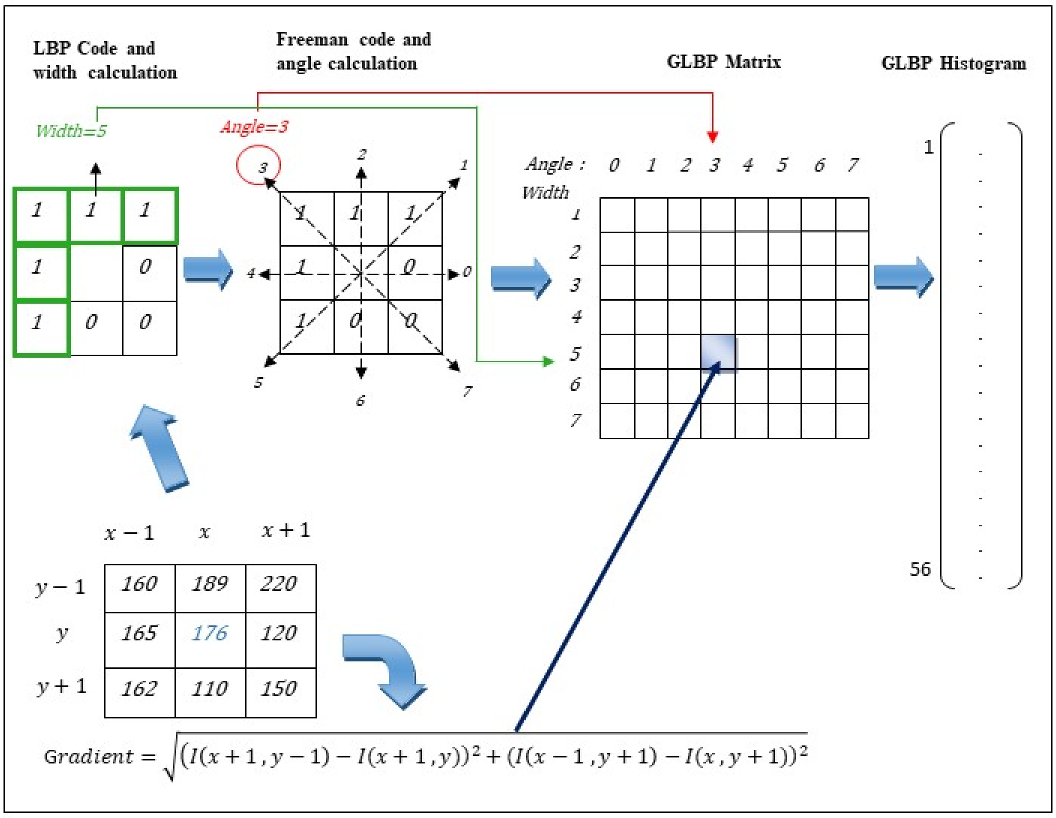


Figure 2.2: GLBP feature extraction for each pixel [36]

2.2.3 Multiscale Local Oriented Gradient (MLOG)

The proposed descriptor is inspired from the LDGP descriptor (Local Directional Gradient Pattern) which was developed for facial recognition. The LDGP descriptor has significantly reduced the extraction as well as matching time, while maintaining similar recognition rates as existing state of the art methods related to the application. It identifies the relationship between the high order derivatives of the referenced pixel in four different directions to compute the micropattern which corresponds to the local feature [37].

MLOG is a local descriptor that takes in consideration multiple neighborhoods with different radius for each pixel. Its multiscale particularity allows to encode n^{th} order derivatives for 3 sets of neighborhood in different directions.

- Set A: First order neighborhood with 5 directions of $0°, 45°, 90°, 135°$ and $180°(R = 1)$.
- Set B: Second order neighborhood with 5 directions of $0°, 45°, 90°, 135°$ and $180°(R = 2)$.
- Set C: Second order neighborhood with the 4 directions of $30°, 60°, 120°$ and $165°(R = 2)$.

Figure 2.3 illustrates the three sets of neighborhoods.

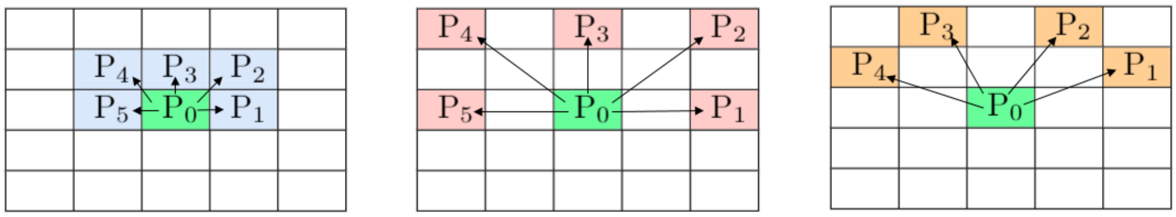


Figure 2.3: From left to right: Sets A, B and C of neighborhoods

For each one of these sets, we extract features that will be concatenated in order to form the final feature vector. These features are extracted as follows:

- **For the set A:**

  1. The first step is to compute the 5 shifted versions of the given image I in the directions $0°, 45°, 90°, 135°$ and $180°$, while preserving the dimensions with a padding. Thus obtaining the matrices $I_{0°}, I_{45°}, I_{90°}, I_{135°}$ and $I_{180°}$. Figure 2.4 illustrates a shifting example of a given image I in the directions $0°, 45°, 90°, 135°$ and $180°$.

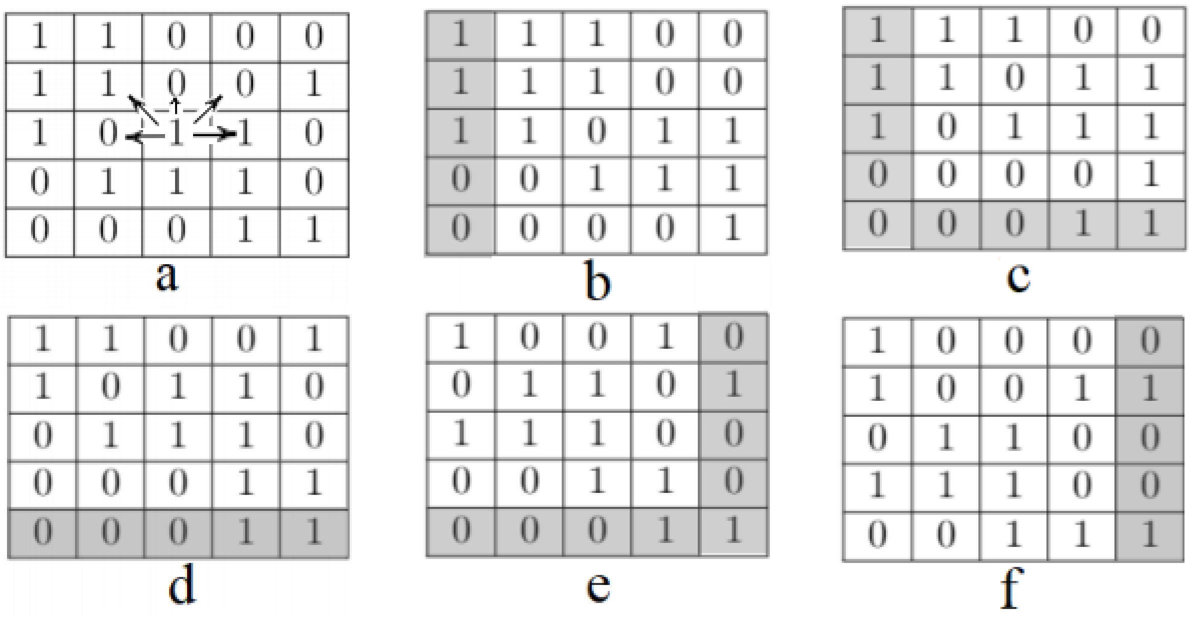


Figure 2.4: Example matrix ”a” shifted to form the matrices b, c, d, e and f, following the directions $0°, 45°, 90°, 135°$and $180°$, respectively

2. The second step is to compute the first order directional derivatives for each pixel P_0 from the given images I_θ, in their respective five directions of $0°, 45°, 90°, 135°$ and $180°$ as follows:

$$I^1_{0°}(P_0) = I_{0°}(P_0) - I_{0°}(P_1) \tag{2.3}$$
$$I^1_{45°}(P_0) = I_{45°}(P_0) - I_{45°}(P_2) \tag{2.4}$$
$$I^1_{90°}(P_0) = I_{90°}(P_0) - I_{90°}(P_3) \tag{2.5}$$
$$I^1_{135°}(P_0) = I_{135°}(P_0) - I_{135°}(P_4) \tag{2.6}$$
$$I^1_{180°}(P_0) = I_{180°}(P_0) - I_{180°}(P_5) \tag{2.7}$$

After processing all the pixels, we obtain five different matrices of the first order directional derivatives that are $I^1_{0°}, I^1_{45°}, I^1_{90°}, I^1_{135°}$ and $I^1_{180°}$.

3. The second order directional derivatives are computed for each pixel P_0 from first order derivatives as:

$$I^2_{0°}(P_0) = I^1_{0°}(P_0) - I^1_{0°}(P_1) \tag{2.8}$$
$$I^2_{45°}(P_0) = I^1_{45°}(P_0) - I^1_{45°}(P_2) \tag{2.9}$$
$$I^2_{90°}(P_0) = I^1_{90°}(P_0) - I^1_{90°}(P_3) \tag{2.10}$$
$$I^2_{135°}(P_0) = I^1_{135°}(P_0) - I^1_{135°}(P_4) \tag{2.11}$$
$$I^2_{180°}(P_0) = I^1_{180°}(P_0) - I^1_{180°}(P_5) \tag{2.12}$$

This computation of the directional derivatives continues until the n^{th} order directional derivatives, which are calculated from the $(n-1)^{th}$ order as:

$$I_\theta^n(P_0) = I_\theta^{n-1}(P_0) - I_\theta^{n-1}(P_i) \tag{2.13}$$

With $i = 1, \dots, 5$ and $\theta = 0°, 45°, 90°, 135°, 180°$ respectively.

Figure 2.5 depicts different order directional derivatives of a handwritten word.

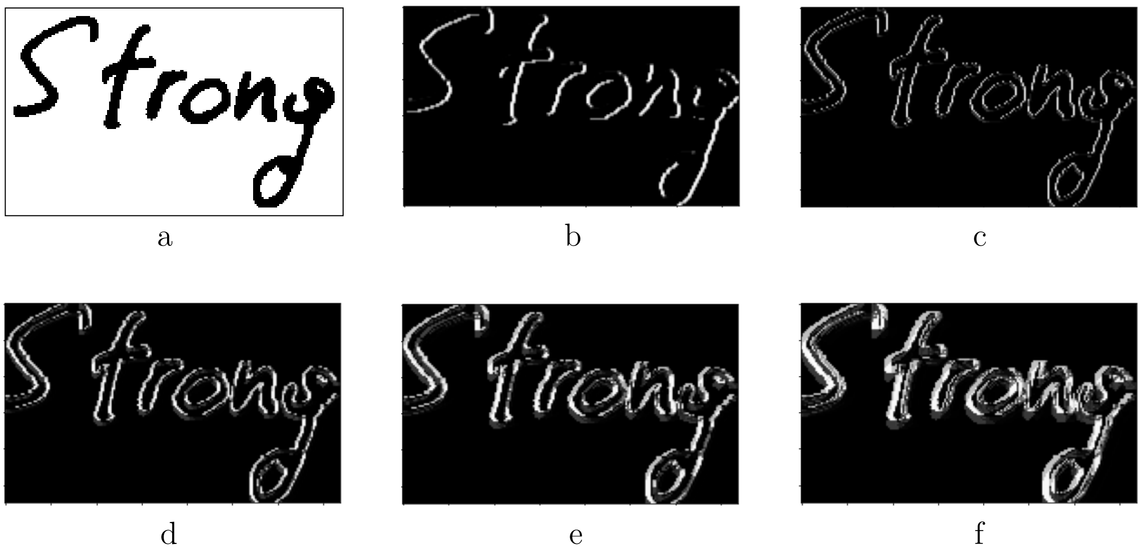


Figure 2.5: MLOG order directional derivatives of set A. (a) orignal image, (b) 1^{st} order derivative, (c) 2^{nd} order derivative, (d) 3^{rd} order derivative, (e) 4^{th} order derivative, (f) 5^{th} order derivative.

4. Thereafter, the n^{th} order directional derivatives are encoded through an encoding function $\mathcal{F}(.)$ to obtain $MLOG_A^n(P_0)$ as :

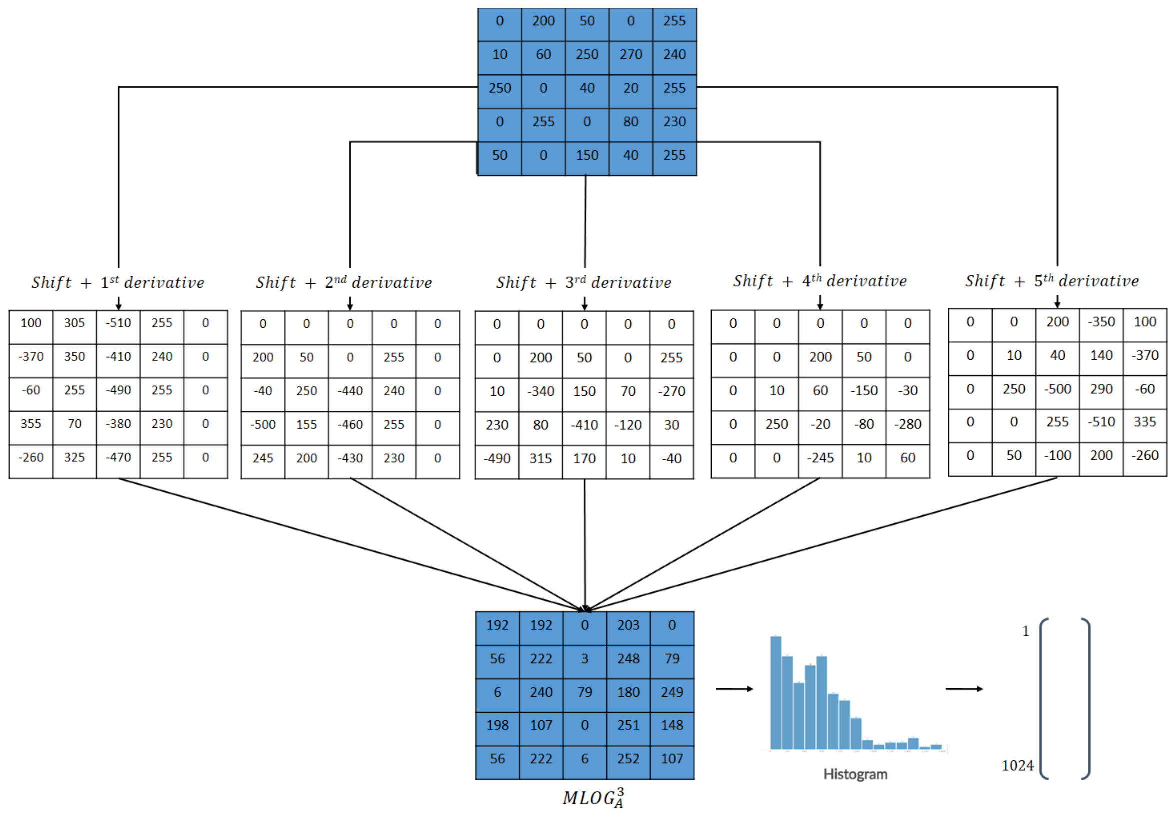
$$MLOG_A^n(P_0) = \begin{Bmatrix} \mathcal{F}(P_0, I_{0°}^n, I_{45°}^n), \mathcal{F}(P_0, I_{0°}^n, I_{90°}^n), \mathcal{F}(P_0, I_{0°}^n I_{135°}^n), \mathcal{F}(P_0, I_{0°}^n, I_{180°}^n), \mathcal{F}(P_0, I_{45°}^n, I_{90°}^n), \\ \mathcal{F}(P_0, I_{45°}^n, I_{135°}^n), \mathcal{F}(P_0, I_{45°}^n, I_{180°}^n), \mathcal{F}(P_0, I_{90°}^n, I_{135°}^n), \mathcal{F}(P_0, I_{90°}^n, I_{180°}^n), \\ \mathcal{F}(P_0, I_{135°}^n, I_{180°}^n) \end{Bmatrix} \tag{2.14}$$

Where $\mathcal{F}(.)$ defines the encoding function that compares the pair of the n^{th} derivative directions of the pixel P_0 as:

$$\mathcal{F}(P_0, I_\alpha^m, I_\beta^m) = \begin{Bmatrix} 1 & if \quad I_\alpha^m(P_0) \geq I_\beta^m(P_0) \\ 0 & otherwise \end{Bmatrix} \tag{2.15}$$

It is clear from equation 2.14 that $MLOG_A^n$ computes a 10 bits binary pattern for each pixel P_0 using the corresponding derivatives in $0°, 45°, 90°, 135°$ and $180°$ directions.

5. This 10 bits binary number is then converted into the equivalent decimal value yielding a maximum number of $2^{10} = 1024$ possible values. Spatial histogram of these decimal values is computed to represent the histogram feature of the handwritten word for the set A. The obtained vector of size 1024 is normalized according to the height and width of the image. Figure 2.6 shows an example of calculation of MLOG of order 3 for the set A.



0	200	50	0	255
10	60	250	270	240
250	0	40	20	255
0	255	0	80	230
50	0	150	40	255

Shift + 1st derivative

100	305	-510	255	0
-370	350	-410	240	0
-60	255	-490	255	0
355	70	-380	230	0
-260	325	-470	255	0

Shift + 2nd derivative

0	0	0	0	0
200	50	0	255	0
-40	250	-440	240	0
-500	155	-460	255	0
245	200	-430	230	0

Shift + 3rd derivative

0	0	0	0	0
0	200	50	0	255
10	-340	150	70	-270
230	80	-410	-120	30
-490	315	170	10	-40

Shift + 4th derivative

0	0	0	0	0
0	0	200	50	0
0	10	60	-150	-30
0	250	-20	-80	-280
0	0	-245	10	60

Shift + 5th derivative

0	0	200	-350	100
0	10	40	140	-370
0	250	-500	290	-60
0	0	255	-510	335
0	50	-100	200	-260

$MLOG_A^3$

192	192	0	203	0
56	222	3	248	79
6	240	79	180	249
198	107	0	251	148
56	222	6	252	107

Figure 2.6: Example of $MLOG_A^3$ calculation for set A

Figure 2.7 summarizes the feature extraction process for the set A.

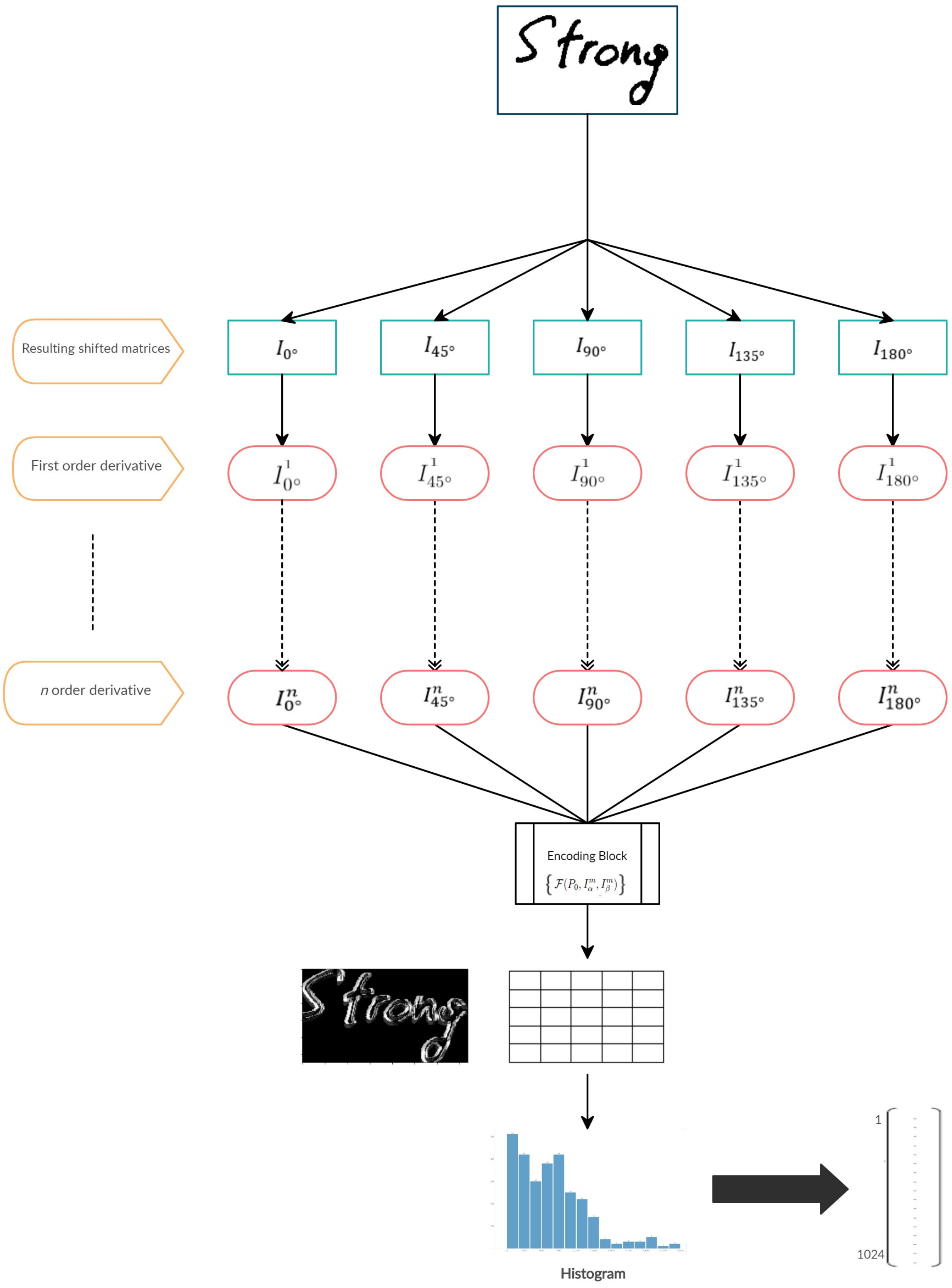


Figure 2.7: Feature generation from set A

- **For the set B:**

  We basically repeat the same procedure by considering the second neighborhood at directions $0°, 45°, 90°, 135°$ and $180°$ with a radius of 2 as depicted in Figure 2.3. This leads to a feature vector of 1024 elements.

- **For the set C:**
  With a radius of 2, we repeat the same procedure but this time with four directions that are $30°, 60°, 120°$, and $150°$ as presented it Figure 2.3, which leads to a feature vector of 64 elements.

Note that the resulting feature vectors from sets A, B and C have common 0 values. This allowed to reduce the size of the feature vectors as follows:

– Feature vector of set A: from 1024 to 72.

– Feature vector of set B: from 1024 to 72.

– Feature vector of set C: from 64 to 24.

Therefore, We obtain the final feature vector by concatenating the feature vectors from the 3 sets, where experimentally determined weightings, ψ and ϕ are applied to the vectors obtained from sets B and C, respectively. This results in a feature vector of 168 elements for a handwritten word image. Figure 2.8 illustrates this process.

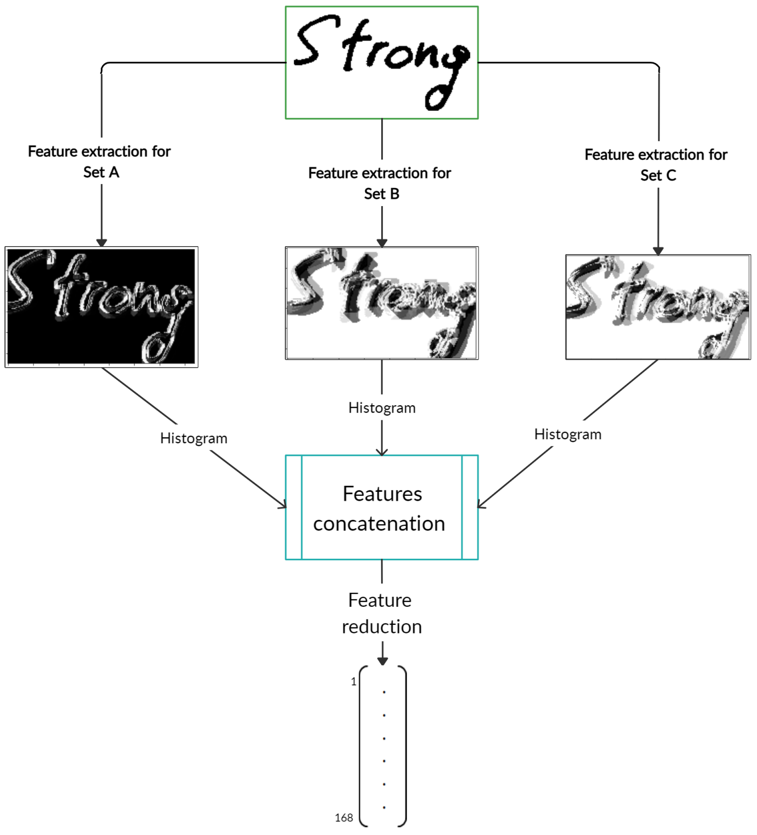

Feature extraction for Set A
Feature extraction for Set B
Feature extraction for Set C
Histogram
Histogram
Histogram
Features concatenation
Feature reduction
1
168

Figure 2.8: Features extraction using MLOG descriptor

2.3 Adopted classifier

Classification consists of assigning a class label to a set of unclassified instances. The literature of writer identification reports the use of several classifiers from a simple distance metric to the use of Convolutional Neural Networks. However, in this work, we chose to use Support Vector Machine for writer identification.

2.3.1 Choice of classifier

Amongst all the classifiers reported in the literature, our choice fell upon Support Vector Machine (SVM) classifier for writer identification from handwritten words. The main reasons that make SVM one of the most performant classifiers in pattern recognition, and thus justify this choice are as follows:

- Guaranteed optimality: Owing to the nature of convex optimization, the solution will always be a global minimum, not a local minimum.
- SVM can be used for linearly separable as well as non-linearly separable data.
- The kernel trick is the real strength of SVM. With an appropriate kernel function, we can solve almost any complex problem.
- It is effective in high dimensional spaces.
- SVM models have good generalization in practice, there is a lower risk of overfitting if the parameters are wisely chosen.

The next section will be a more in depth explanation about the functioning of this classifier.

2.3.2 Support Vector Machine (SVM)

SVM was first introduced in 1995 by Vladimir N. Vapnik, but it was not until 1998, that it began to be extensively used in pattern recognition, following the famous tutorial of Christopher J. C. Burges [38] which presents a simple and effective formulation for the use of SVM in pattern classification.

Support Vector Machine is a two-class supervised machine learning algorithm that arises from the statistical learning theory. The aim is to construct a hyperplane or a set of hyperplanes in a high or infinite dimensional space that guarantee an optimal separation between the two classes. Intuitively, a good separation between the two classes is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called margin function), since in general the larger the margin the lower the

generalization error of the classifier.
Indeed, as illustrated in Figure 2.9, there are several hyperplanes that correctly classify all samples. However, the optimal choice is the one that gives the maximum marging between the two classes. The Figure clearly shows that the margin of H_1 is greater than that of H_2, hence, the optimal hyperplane is the one that gives this maximum distance (in red).

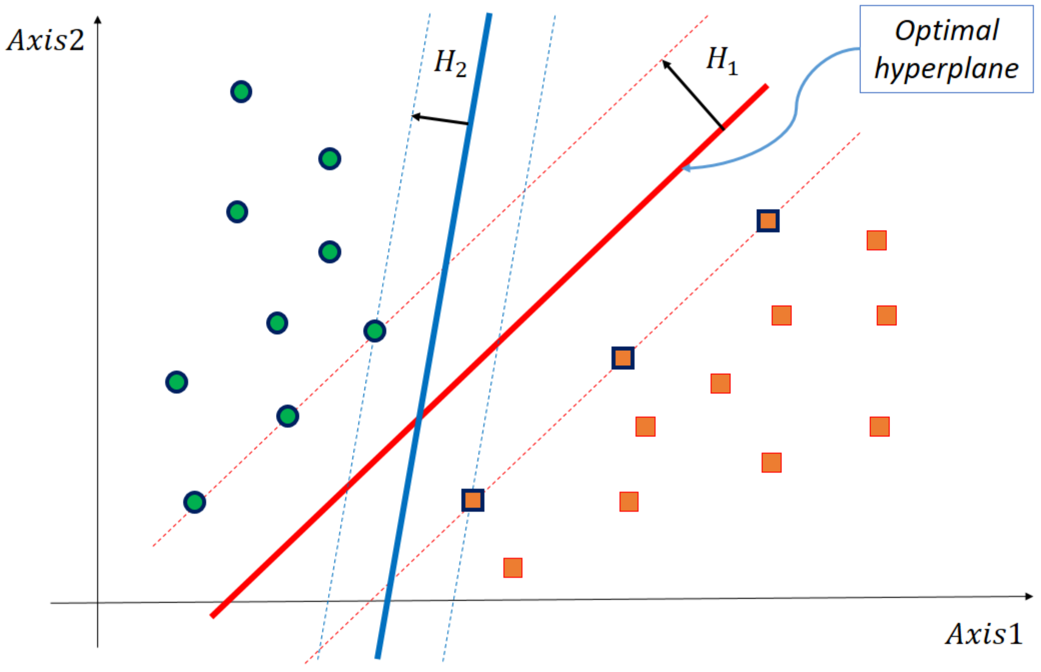


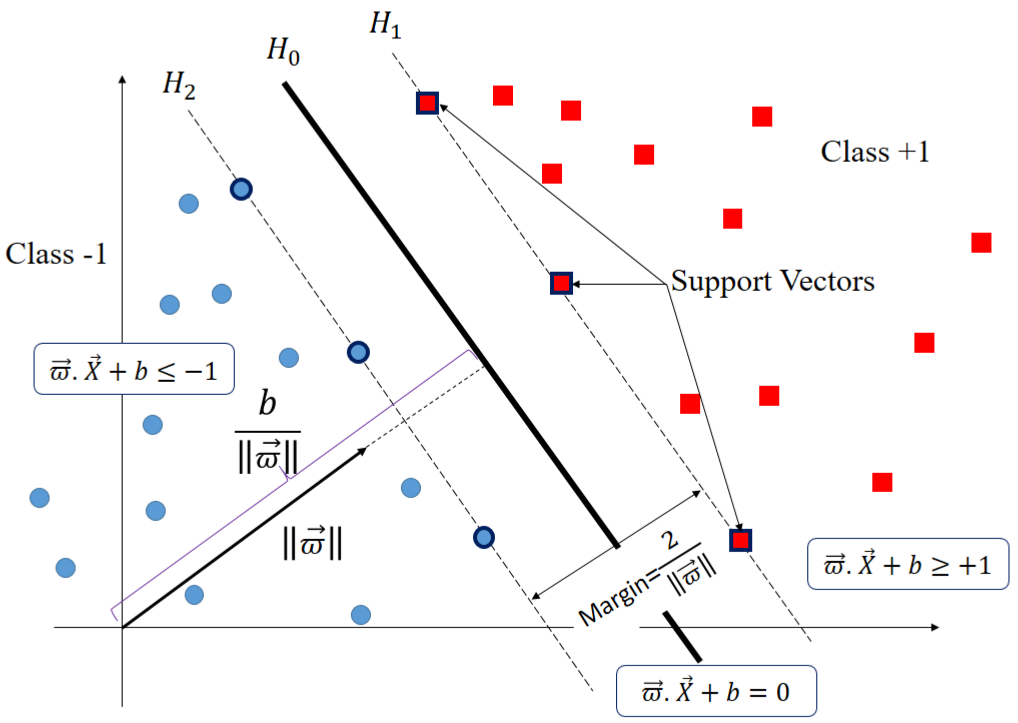
Figure 2.9: Optimal margin hyperplane separator

For linearly separable data, hard margin is applied, where the entire training set needs to be correctly classified (all points with the same label must be on the same side of the hyperplane).

As depicted in Figure 2.10, the equation of the separating hyperplane is $\vec{\varpi} \cdot \vec{X} + b = 0$ where $\vec{\varpi}$ is the normal vector to the hyperplane, b a constant(bias) and $\vec{X}$ the set of points that belong to the hyperplane. Let $\vec{X_i}$ be a sample of training data with a label $y_i = +1$ or -1. $\vec{X_i}$ satisfies one of the following two inequalities:

$$\vec{\varpi} \cdot \vec{X_i} + b \geq +1, \ y_i = +1 \tag{2.16}$$

$$\vec{\varpi} \cdot \vec{X_i} + b \leq -1, \ y_i = -1 \tag{2.17}$$



H_1
H_0
H_2
Class +1
Class -1
Support Vectors
$\vec{\varpi}.\vec{X} + b \leq -1$
$\frac{b}{\|\vec{\varpi}\|}$
$\|\vec{\varpi}\|$
Margin=$\frac{2}{\|\vec{\varpi}\|}$
$\vec{\varpi}.\vec{X} + b \geq +1$
$\vec{\varpi}.\vec{X} + b = 0$

Figure 2.10: Optimal margin hyperplane for linearly separable data

Equations 2.16 and 2.17 can be more conveniently expressed as:

$$y_i\left(\vec{\varpi}\cdot\vec{X}_i + b\right) - 1 \geq 0 \quad \forall\, i \tag{2.18}$$

The margin being equal to $\frac{2}{\|\vec{\varpi}\|}$, maximizing it leads to minimizing $\frac{1}{2}\|\vec{\varpi}\|^2$, the problem is then transformed to finding $\vec{\varpi}$ and b such that the expression $\frac{1}{2}\|\vec{\varpi}\|^2$ is minimized and the constraint of the inequality 2.18 is verified.

After some optimization process, the decision function is obtained and given by:

$$f(X) = sign(\sum_{X_i \in SV} \alpha_i y_i \vec{X}_i \cdot \vec{X} + b) \tag{2.19}$$

Where:
$\vec{X}$: *The data to be classified*
$\vec{X}_i$: *The vectors of the training data*
y_i : *The class labels of the training data.*
b : *The bias.*
α_i : *The lagrangian multiplier*

In most real-word problems, data are not linearly separable. Therefore, SVM has been generalized through two concepts: the soft margin and kernel functions to provide a linear separation.

- In the case of soft margin, SVM chooses a hyperplane while tolerating some data to be misclassified. In this case, the inequality 2.18 becomes:

$$y_i \left(\vec{\varpi} \cdot \vec{X} + b \right) - 1 \geq \zeta_i \ , \quad \zeta_i \geq 0 \ \forall i \tag{2.20}$$

  Where ζ is a slack variable. An error is counted if $\zeta \geq 1$.
  The term $\sum_i \zeta_i$ constitutes a higher limit on the number of misclassifications on the training set. In this case, the calculated hyperplane does not necessarily correspond to the optimal solution. To control the cost over errors, the term $\frac{1}{2} \left\| \vec{\varpi} \right\|^2$ is replaced by $\frac{1}{2} \left\| \vec{\varpi} \right\|^2 + C(\sum_i \zeta_i)$. C is a user-defined parameter ($C \geqslant 0$) called the regularization parameter that controls the trade-off between errors on training data and the width of the margin. The bigger C is, the less misclassifications are tolerated. Thus, the narrower the margin.
  Using soft margin, the model tries to find the hyperplane that balances between the two opposing constraints of maximizing the margin and minimizing the misclassification.

- Admitting the misclassification of some samples can not always give a good generalization for a hyperplane, even if it is optimized. Hence, the need of mapping the feautres into a potentially much higher dimensional feature space via a non linear mapping function K as shown in Figure 2.11. The decision function when employing kernels becomes:

$$f(X) = sign(\sum_{X_i \in SV} \alpha_i y_i K(\vec{X_i}, \vec{X}) + b) \tag{2.21}$$

  Where $K(\vec{X_i}, \vec{X})$ is the kernel function.

  It is worth mentioning that several kernel functions exist. The most convenient and efficient kernel in most pattern recognition tasks is the Radial Basis Function (RBF). Its flexibility and general purpose characteristics make it the optimal choice in complex classification problems, such as writer identification from a single handwritten word.

  In the case of the RBF kernel, $K(\vec{X_i}, \vec{X})$ is defined as:

$$K(\vec{X_i}, \vec{X}) = exp \left(- \frac{\left\| \vec{X} - \vec{X_i} \right\|^2}{2\sigma^2} \right) \tag{2.22}$$

  Where σ is the standard deviation of the gaussian function. It defines how much influence a single training example has on the boundary decision. The higher the σ, the more influence each training example has on the boundary decision.

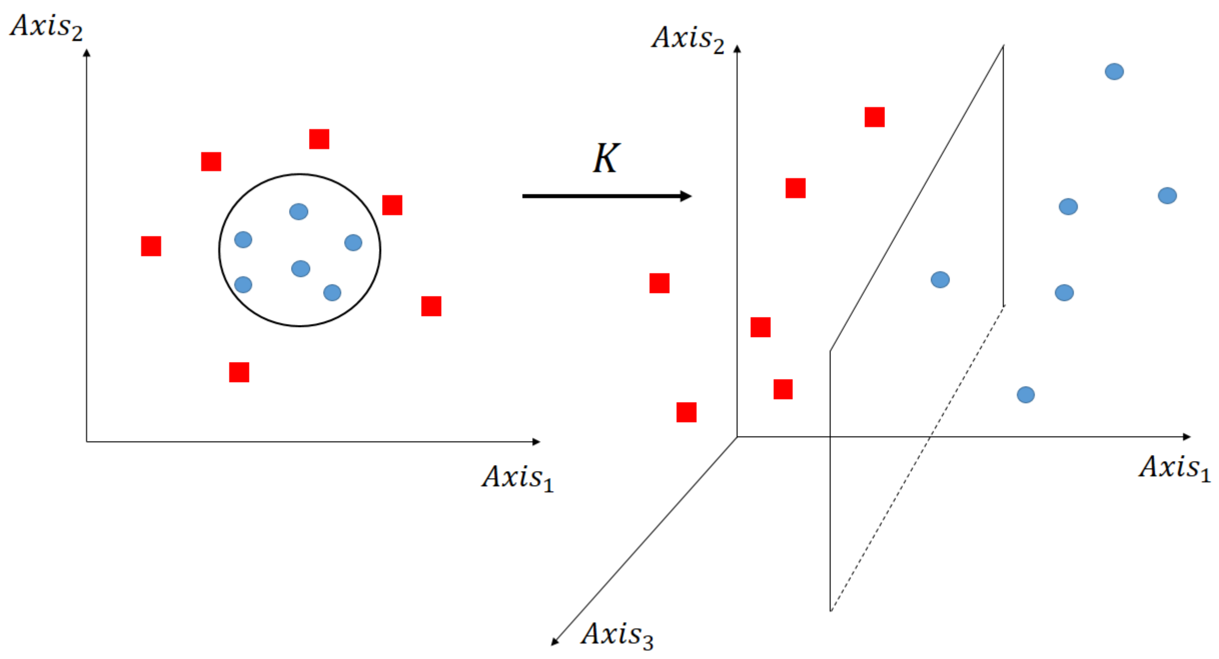


Figure 2.11: Mapping features into a higher dimensional space

■ *Extension to multiclass classification*

Traditionally, multiclass classification-based SVM is addressed through two different strategies. The first strategy is the "one-versus-all" also referred to as "one-versus-rest", which involves training a single classifier per class, with the samples of that class as positive samples and samples of all remaining classes as negatives. Hence, N classifiers are needed to separate N classes. Usually, classification of an unknown pattern is done according to the maximum output among all binary classifiers. Although this strategy is widely used, it suffers from 2 main problems. First, the range of the confidence values may vary between the binary classifiers. Second, even if the class distribution is balanced in the training set, the binary classifiers see unbalanced distributions because typically, the set of negatives is much larger than the set of positives.

The second strategy is the "one-versus-one" also referred to as "pairwise coupling", which involves training $N(N-1)/2$ binary classifiers to separate N classes. Each binary classifier receives the samples of a pair of classes from the original training set. Usually, classification of an unknown pattern is done according to the maximum voting, where each SVM votes for one class.

It can seem logical that the total training time with the "one-versus-one" strategy is larger than with the "one-versus-all", because it is imperative to train more binary classifiers; but it is not true when the binary classifiers are SVM. Indeed, the training time of an SVM increases more with the number of training samples. Thus, since each sub-problem involves a small number of training samples and is easier to solve, it is quicker to train the $N(N-1)/2$ SVM of the “one-versus-one” strategy than the N SVM of the “one-versus-all” strategy. Consequently, if the number of training samples is very large, the training time can become problematic, and then the “one-versus-one” strategy appears more suitable for practical use. In addition, for problems with a large number of classes, like writer identification, the unbalanced number of the samples could cause problems with the “one-versus-all” strategy, especially when it has few training samples per class [39].

2.4 Conclusion

In this chapter, we introduced the feature extraction methods adopted in this work, especially the brand new proposed Multiscale Local Oriented Gradient (MLOG) descriptor. Besides, we gave an overview on the selected Support Vector Machine classifier for writer identification. However, the complexity of the target task in our work, which is identifying the author from a single handwritten word, makes that a simple pattern recognition system based on feature extraction and classification steps might not give the expected results. Hence, when in the last years, a new generation of machine learning algorithms called deep leaning had emerged, it is only rational to investigate the outcomes of using such algorithms on such a highly complicated task.
Therefore, the next chapter is dedicated to the most modern deep learning algorithm that is based on artificial neural networks, namely, Convolutional Neural Networks (CNN).

Chapter 3

Convolutional Neural Networks

3.1 Introduction

In the last decade, artificial ntelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines by introducing deep learning. As a part of a broader family of machine learning methods, deep learning is based on artificial neural networks. Indeed, advances in computer vision with deep learning have revolved mainly over the well-known algorithm, called Convolutional Neural Networks (CNN or ConvNet). Nowadays, CNN is considered as the best artificial intelligence algorithm for image classification problems. In the literature of writer identification from handwritten documents, CNN has been widely employed [40, 41]. However, only one work dealing with writer identification from a single handwritten word is reported. Indeed, in 2018, He et al,.[6] implemented a complex architecture of deep adaptive CNN in order to get as much information as possible from handwritten words' images to identify the writer. Findings were promising, nevertheless, an important computational power was needed, and training time was significantly high (7 to 9 hours).

Therefore, in the present work, we propose to investigate the use of CNN as an end-to-end system and as a feature extractor. This chapter first provides a brief overview of Artificial Neural Networks, before moving on to a detailed description of Convolutional Neural Networks. For the implementation of this network, we adopt the famous architecture VGG-16, which is the most used in image processing applications.

3.2 Artificial Neural Networks over time

Artificial Intelligence (AI) has grown considerably in recent years. Being at the crossroads of computer science, mathematics and neurobiology. AI aims to allow machines to imitate or even simulate the behavior of the human brain, in order to solve very complex problems, such as classification, recognition and detection. The history of artificial neural networks goes back to 1943, when Warren McCulloch, a neurophysiologist and Walter Pitts, a mathematician, wrote a paper in which, inspired by biological neurons, modeled the first artificial neuron using electrical circuits.
In 1949, Donald Hebb wrote his book "The Organization of Behavior", in which he provided further insights for the concept of neurons through explaining that neural pathways are strengthened each time they are used. A discovery that provided essential knowledge to the ways the human mind learns.
In the 1950s, as computers became more advanced, Nathanial Rochester from the IBM research laboratories led the first attempt to simulate an artificial neural network.
In 1957, inspired by previous works, Frank Rosenblatt developed the model of the perceptron, representing the oldest neural network based on a single layer of neurons, which has

two main flaws: it can only separate between two classes and the data must be linearly separable.

Furthermore, at that time, computers did not have enough processing power to effectively manage the time required by large neural networks. This stagnated the research field of neural networks for many years, before it was recognized in the mid-1980s, when computers reached higher processing power, that a neural network with two or more layers is much more powerful in terms of processing, than a single layer perceptron. This led to the Multi-Layer Perceptron (MLP). However, the use of MLP model was put on hold due to the lack of conceptual tools necessary to analyze and predict its behavior. It was not until late 1980s, that neural networks really took off with the appearance of back propagation learning algorithm [42]. Several other neural network models have been developed to solve different problems, of prediction, classification, detection, and others. However, these models require a prior feature extraction operation, mandatory to represent the data. More recently, the increase in the power of computing infrastructures has enabled certain computers to process large amounts of data with deep learning, which refers to artificial neural networks that contain in addition to the classical layers, several layers dedicated to feature generation. The first model called LeNet was proposed by LeCun et al., in 1998 [43], for the recognition of handwritten digits. Subsequently, several models more suitable for other applications were proposed. In the following sections, we will present some necessary definitions for understanding the CNN used in our work.

3.2.1 Perceptron

The most basic existing neural network is the perceptron, which is composed of a single formal neuron. It is considered as the earliest learning process to be made, for the classification of two linearly separable classes. A perceptron is composed of one or many inputs, a processing unit and one output. As illustrated in Figure 3.1, given an input vector $A = (a_1, a_2, ..., a_K)$, the neuron proceeds by a summation of the elements of A, weighted by the elements of the weight vector W, where $W = (w_1, w_2, ..., w_K) \in \mathbb{R}^K$. Then, the resulting sum passes through an activation function $Z(.)$, thus producing the output O.

$$S = \sum_{i=0}^{K} a_i w_i \tag{3.1}$$

The perceptron attempts to find a linear separation between two classes, i.e, it tries to find a separating line defined by its slope $w_{i,i\neq 0}$ and its bias w_0 which is considered as just the weight of another input $a_0 = 1$. Therefore, we can define the summation function as given in equation 3.1.

The activation function that follows, uses an activation threshold β to translate the input signal into an output O. If the resulting input sum is greater than the threshold β, then the neuron is activated and thus produces a positive output. Otherwise, the output will be a zero.

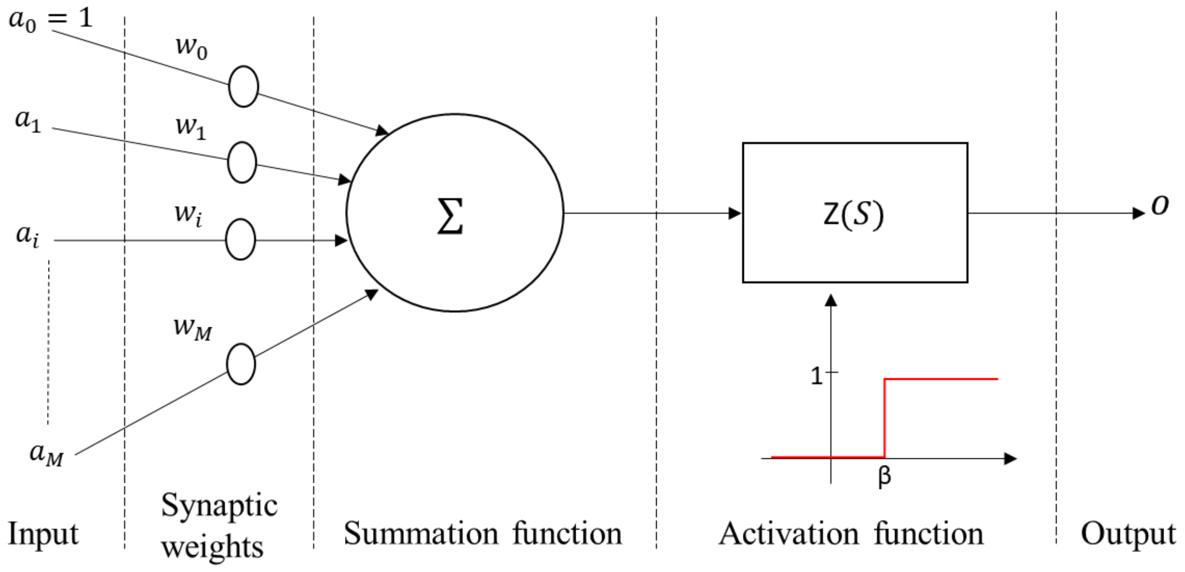


Figure 3.1: Perceptron

3.2.2 Multi Layer Perceptron (MLP)

Multi Layer Perceptron is an extended version of the single layer perceptron. It is composed of an input layer, one or more hidden layers and an output layer, where each neuron of a layer is connected to all neurons of the adjacent layer, constructing a fully connected network of at least three layers (see Figure 3.2). The number of neurons in the input layer defines the dimension of the input feature, while the number of neurons in the output layer defines the number of classes. Whereas the number of neurons in hidden layers is considered a design issue. The fewer hidden neurons are, the harder it is for the model to learn complex decision limits. On the other hand, the more hidden layers there are, the less generalized the model becomes [36].

In order to determine the synaptic weights w_i, MLP applies backpropagation when learning, which consists of adjusting the weights of the network by minimizing the mean squared error e between the actual output and the predicted output $(\hat{\mu}, \mu)$ respectively.

$$e_{\hat{\mu},\mu} = \frac{1}{2} \sum_{i=1}^{m} (\hat{\mu}_i - \mu_i)^2 \tag{3.2}$$

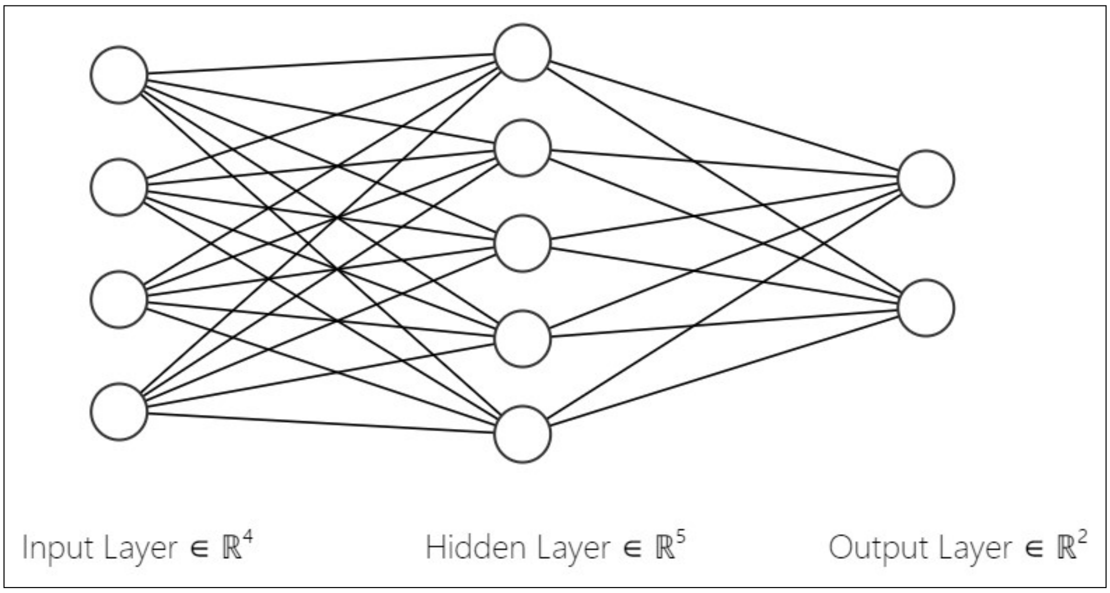

Input Layer ∈ $\mathbb{R}^4$
Hidden Layer ∈ $\mathbb{R}^5$
Output Layer ∈ $\mathbb{R}^2$

Figure 3.2: Multi Layer Perceptron with one hidden layer

The synaptic weights are then modified as follows :

$$w(l+1) = w(l) + \Delta w(l+1) \tag{3.3}$$

where $\Delta w(l+1)$ is the weight change over iteration $(l+1)$, calculated by:

$$\Delta w(l+1) = -\tau \frac{\partial e}{\partial w} + \rho \Delta w(l) \tag{3.4}$$

The two parameters τ and ρ, respectively learning rate and momentum, are used to ensure stability of the convergence process by lowering the variations in weights changes over two consecutive iterations.

3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN or ConvNet) are an extension of MLP that allow to effectively overcome the flaws of the latter. They are designed to automatically extract features of the input data. They were initially inspired by the discovery made by Hubel and Wiesel, in 1962, where they noted that particular patterns stimulated activity in specific parts of the cats brain [44]. The first use of these networks was carried out by Fukushima with the self-organizing map (SOM) for features extraction using unsupervised learning [45]. In pattern recognition, an important development was attained by LeCun et al., in [43], who proposed LeNet architecture for the handwriting digit recognition.
CNN eliminates the need for manual feature extraction. It extracts features and assigns learnable weights and biases to various aspects of the image in order to differentiate one from the other. Technically, each input image will pass through two blocks of layers. A

first block containing the convolutional layers which are responsible for generating the characteristics, and a second classification block which contains the Fully Connected layers, similarly to the Multi Layer Perceptron (MLP). Figure 3.3 depicts the arrangement of the layers in a CNN.

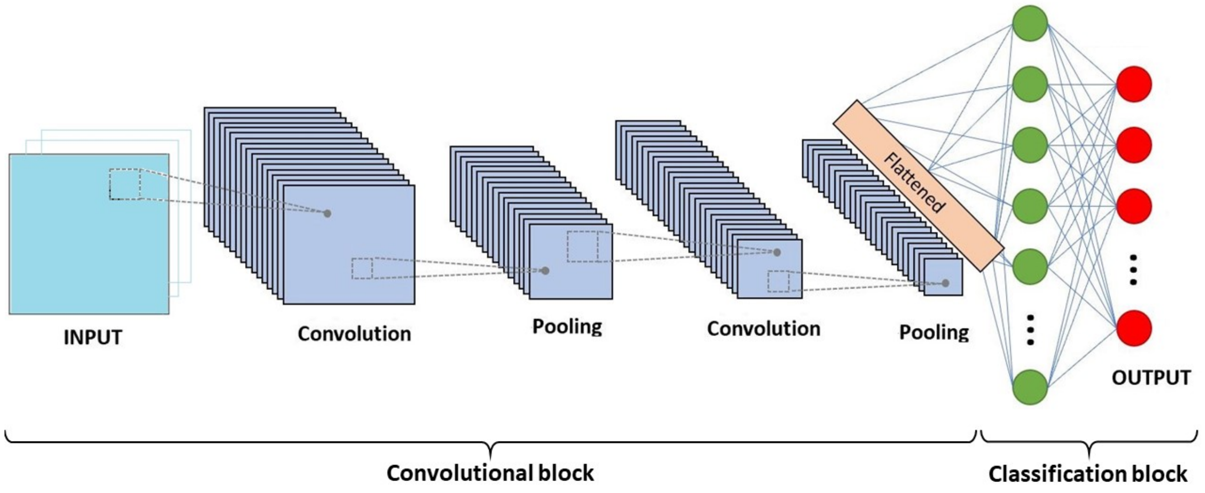


Figure 3.3: CNN layers

3.3.1 Feature extraction layers block

This first block makes the particularity of this network, since it functions as a feature extractor. The first layer of this block filters the image with several convolution kernels, and returns features which are normalized or resized via an activation function, yielding a feature map. This process can be repeated several times. Obtained feature maps are filtered using new kernels each time, which gives new features to normalize, filter again, and so on. Finally, the values of the last features are concatenated and flattened into a column vector [46]. This vector defines the output of the first block, and the input of the second.
More precisely, there are essentially two types of layers in this part of the network. The convolutional layers and the pooling layers.

- **Convolutional layers**
  A 2D convolutional layer extracts features from an input image and preserves the relationship between pixels by learning these image features using squared filters. It takes two inputs: image matrix of dimension h x w x d (where h is the height, w is the width and d is the number of channels) and a filter or kernel of dimension f_hxf_wx d. Typically, the input of the convolutional layer is an RGB color image, so it is represented by 3 channels. In the case of a binary or grayscale image, the input image of the convolutional layer is represented by a single channel. The output

matrix is calculated with sequential dot product between the filter and a part of the input matrix. The filter is dragged across the image from left to right, top to bottom, with a number of steps called strides, to perform the same calculation as depicted in Figure 3.4.

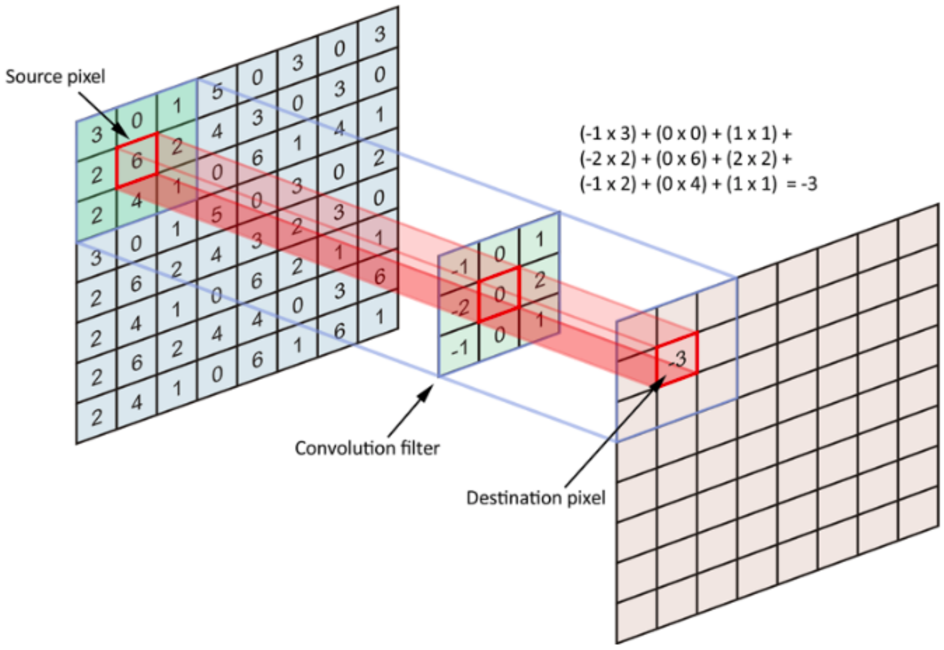

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Figure 3.4: 2D convolutional layer

- **Pooling layers**

  The pooling layer performs an image sub-sampling operation according to the spatial dimensions width and height only. The input image is divided into a series of non overlapping squares. This type of layer is frequently inserted between two successive convolutional layers of a CNN architecture to reduce the number of parameters and required computational power, while preserving their most important characteristics. In practice, the image is divided into adjacent small square cells, spaced from each other by a stride, in order to not lose too much information. The same number of output images as input images is obtained but each image has a lower number of pixels [46]. There are two main types of pooling: max-pooling and average-pooling. Max-pooling returns the maximum value from the portion of the image covered by the kernel. On the other hand, average-pooling returns the average of all the values from the portion of the image covered by the kernel. In practice, max-pooling is the most used and performs a lot better than average-pooling. Figure 3.5 represents an example of max-pooling and average-pooling operations.

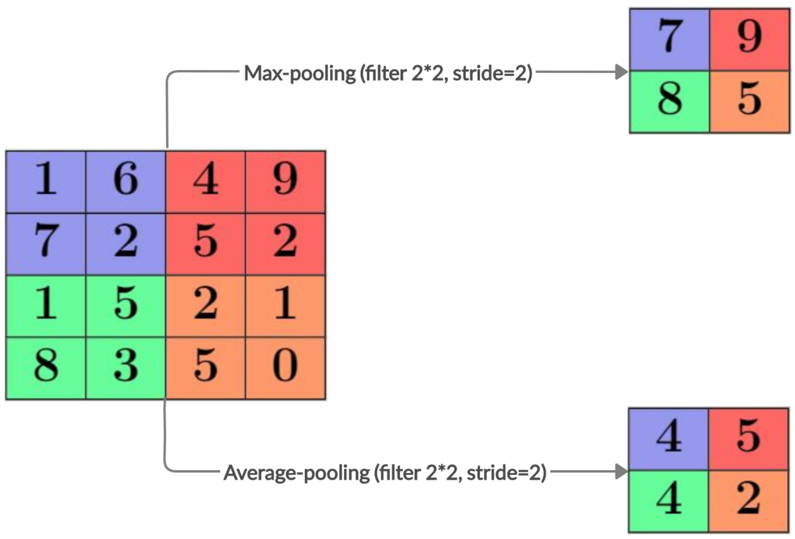

Max-pooling (filter 2*2, stride=2)
Average-pooling (filter 2*2, stride=2)
1 6 4 9
7 2 5 2
1 5 2 1
8 3 5 0
7 9
8 5
4 5
4 2

Figure 3.5: Example of pooling operation

■ **Flatten layer**

This is the last step in the feature extraction block. Flattening consists of concatenating the rows of matrices in a one-dimensional vector that we can connect to the prediction block (see Figure 3.6).

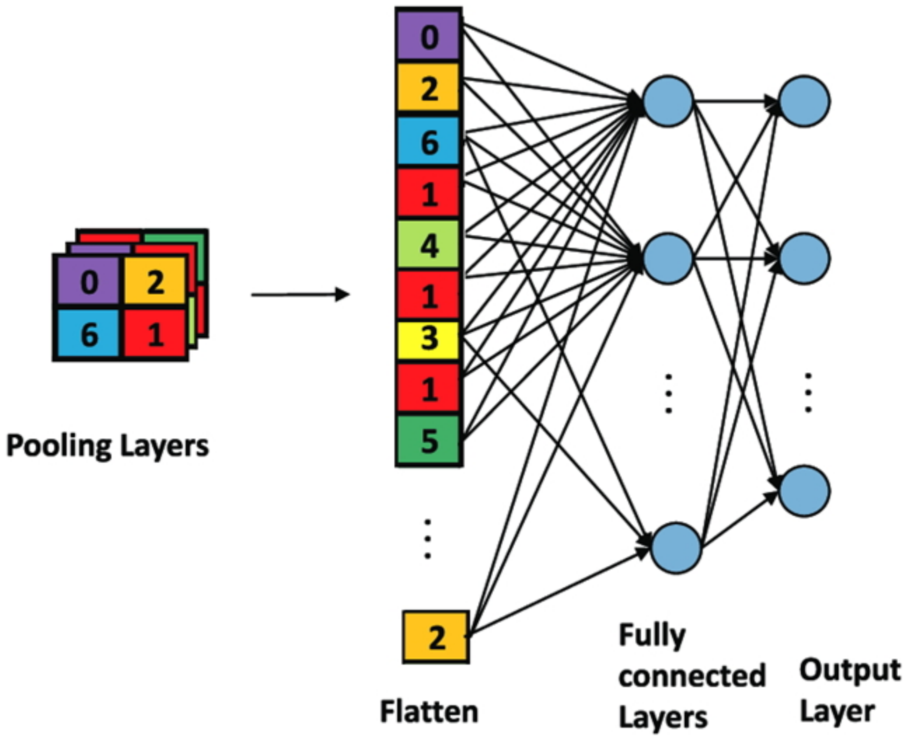


Figure 3.6: Flatten layer illustration connecting the pooling layers to the fully connected layers

3.3.2 Classification layers block

Classification is done through Fully Connected (FC) Layers, similar to the Multi-Layer Perceptron. The objective of the FC layers is to use the input flatten vector to learn the best parameters to classify the image into a class. In practice, several layers are placed one after the other to reinforce features learning and improve decision-making. ”Dropout” can be partially applied on these layers in order to reduce overfitting and learning time by reducing the number of parameters to define. The last FC layer returns the CNN output, which corresponds to a vector of size equal to the number of classes. The neurons in this layer use the softmax activation function to provide the probabilities with which an input image belongs to different classes [47].

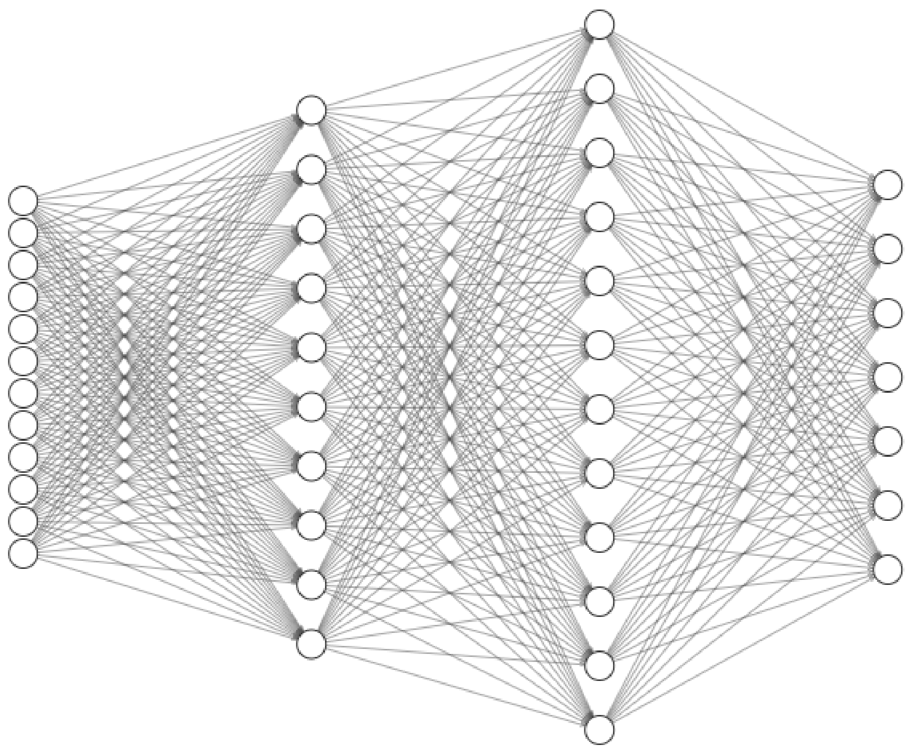


Figure 3.7: A fully connected neural network

3.3.3 CNN parameters

In addition to the parameters of the MLP (learning rate, momentum, number of epochs, number of hidden layers and units), which are used in the block of the FC layers, several parameters are used to manage the convolution block. This section briefly reviews all of these parameters.

- **Filter**
  The convolution consists of sliding a series of filters or convolution kernels over the input image. The choice of filter values, their size and their number (also called

depth) is an experimental task. “Depth of the convolutional layer” refers to the number of kernels used, which is the origin of the name ”Deep learning”. If we use a number of q filters, the image will be convoluted q times. This process creates q filtered matrices called “feature maps”. An example of filtering is illustrated in Figure 3.8. In most of the predefined CNN architectures, there is always a default choice specific to the application for which the model was built [48].

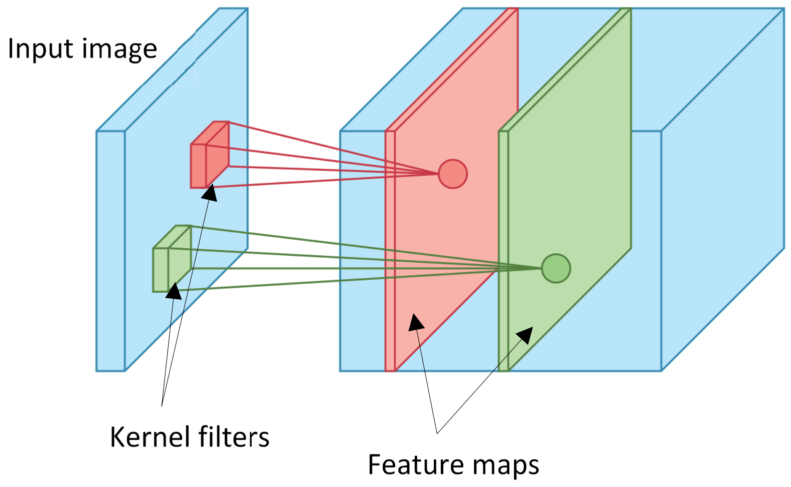


Figure 3.8: Deep leaning feature maps

- **Activation functions**

  They are mathematical functions that discern the output of a neural network. They determine whether a neuron should be activated or not, based on its relevance to the model’s prediction. There are many types of activation functions, we cite three of them:

  – **Rectified Linear Unit (ReLU) function** : Mostly applied in Convolutional Neural Networks, after convolutional layers. It sets all negative values to 0 and keeps the positive values unchanged. It is used in order to increase the non-linearity of the network. It is described by the following equation:

$$G(E) = \max(0, E) = \begin{cases} E & \text{if } E \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

  – **Exponential Linear Unit (ELU) function** : ELUs result negative and positive values, allowing to push mean unit activation towards zero, in a procedure similar to normalization but with lower computational complexity [49], hence, speeding up the learning of the model. It is described by the following equation:

$$G(E) = \begin{cases} E & \text{if } E \geq 0 \\ \alpha\left(e^{E}-1\right) & \text{otherwise} \end{cases} \tag{3.6}$$

Where α is a parameter to define.

– **Softmax function**: Commonly used in the last layer of the network, it is a normalization function that gives an approximation of the probability for a class being correct. The probability value is calculated as follows:

$$G\left(e_j\right) = \frac{\exp\left(e_j\right)}{\sum_i \exp\left(e_i\right)} \tag{3.7}$$

Where e_j is the considered element j of the input vector.

- **Stride**

  Stride is the number of pixels shifted over the input matrix. When the stride is 1, then we move the filters with 1 pixel at a time. When the stride is 2, then we move the filters with 2 pixels at a time and so on.

- **Zero-padding**

  After each convolutional layer, the size of the resulting image is smaller than that of the input image, because the filter cannot process the border pixels. If multiple convolutional layers are used, the output image will be very small compared to the input image. To keep the same size as the input image, the zero-padding adds zeros to the edges of the image, taking into account the size of the filter. In this way, whatever the number of convolutional layers, the output image has the same size as the input image.

- **Dropout**

  It is a technique used during the training phase, designed to reduce the test error and avoid the problem of overfitting linked to fully connected layers. The principle of using the dropout is to randomly switch off neurons so that the network will be more reactive and therefore be able to learn faster. Figure 3.9 presents an example of the technique.

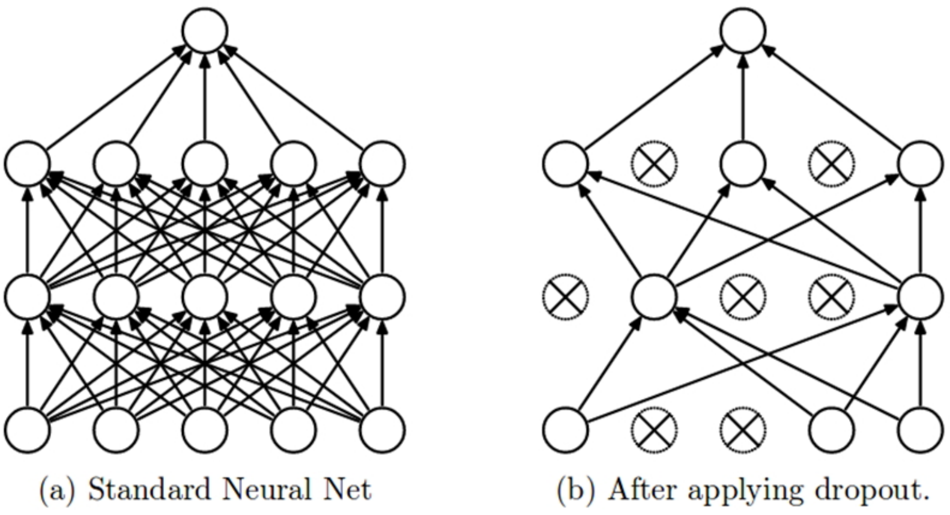


Figure 3.9: Dropout neurons during training phase

3.4 Adopted CNN model: VGG-16

Training CNN to solve a problem is a very complicated task. On one hand, several parameters require adjustment by the user such as number of convolutional layers, number of filters, size of filters, number of FC layers, number of neurons, etc. On the other hand, the optimization of hundreds of thousands of interconnection weights between the different layers, requires a large amount of training data, which is not always available in real life applications. In addition, the training of a CNN containing several convolution blocks, can only be done on powerful machines with a Graphic Processor Unit (GPU). To overcome these limitations, several researchers have developed specific CNN architectures and techniques such as "Transfer learning". As we trained CNN on personal computers or on Google Colaboratory with limited resources, we opted for the use of the VGG-16 architecture, that has been proposed for processing natural images. VGG-16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [48]. The model achieved 92.7% TOP-5 test accuracy on ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. This model improves AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3x3 kernel-sized filters one after another. The architecture of VGG-16 is depicted in Figure 3.10.

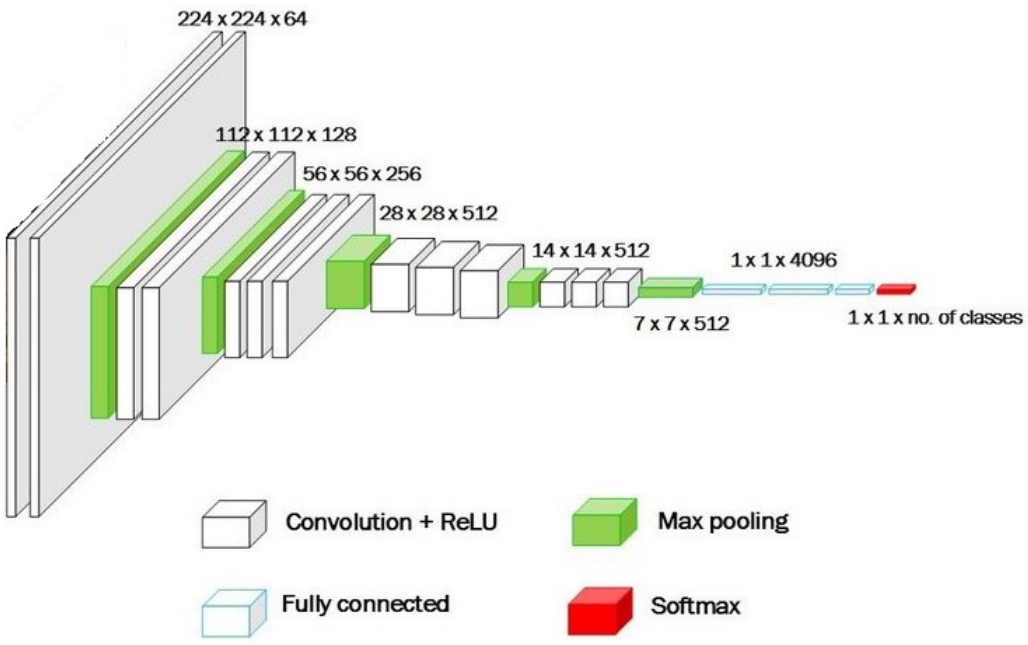


Figure 3.10: VGG-16 model configuration for ImageNet dasaset

The input of the first convolutional layer is an RGB image of size 224 x 224 x 3. The image is passed through a stack of convolutional layers, where the filters were of size 3x3 (which is the smallest size to capture the notion of left/right, up/down, center). The convolution was carried out by 13 convolutional layers. The convolution stride is fixed to 1 pixel; the spatial padding of the convolutional layer's input is such that, the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3x3 convolutional layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the convolution layers (not all of them are followed by max-pooling). Max-pooling is performed over a 2x2 pixel window, with a stride of 2.
After the stack of convolution and max-pooling layers, we obtain (7, 7, 512) feature maps, which are flattened into a feature vector of (1, 25088). Following this, are 3 Fully-Connected layers (FC), with the first two, having 4096 channels each and the third, performing 1000 classes classification, and thus containing 1000 channels (one for each class). The final layer is the softmax layer. Note that, the name VGG-16 came from the 13 convolutional layers plus the 3 FC layers. Figure 3.11 summarizes VGG-16 configuration.

VGG-16 consists of 138 million parameters, which can be challenging to handle as it would require huge computational power resources and a lot of training time. Therefore, a more convenient approach is used which is "Transfer Learning". This last is a machine learning technique, where a model trained for a task is reused as the starting point for training a model on a second task. In our case, we used the parameters of VGG-16 model trained on ImageNet, as a starting point for training our model to perform writer identification from single handwritten words and extract the most meaningful features.

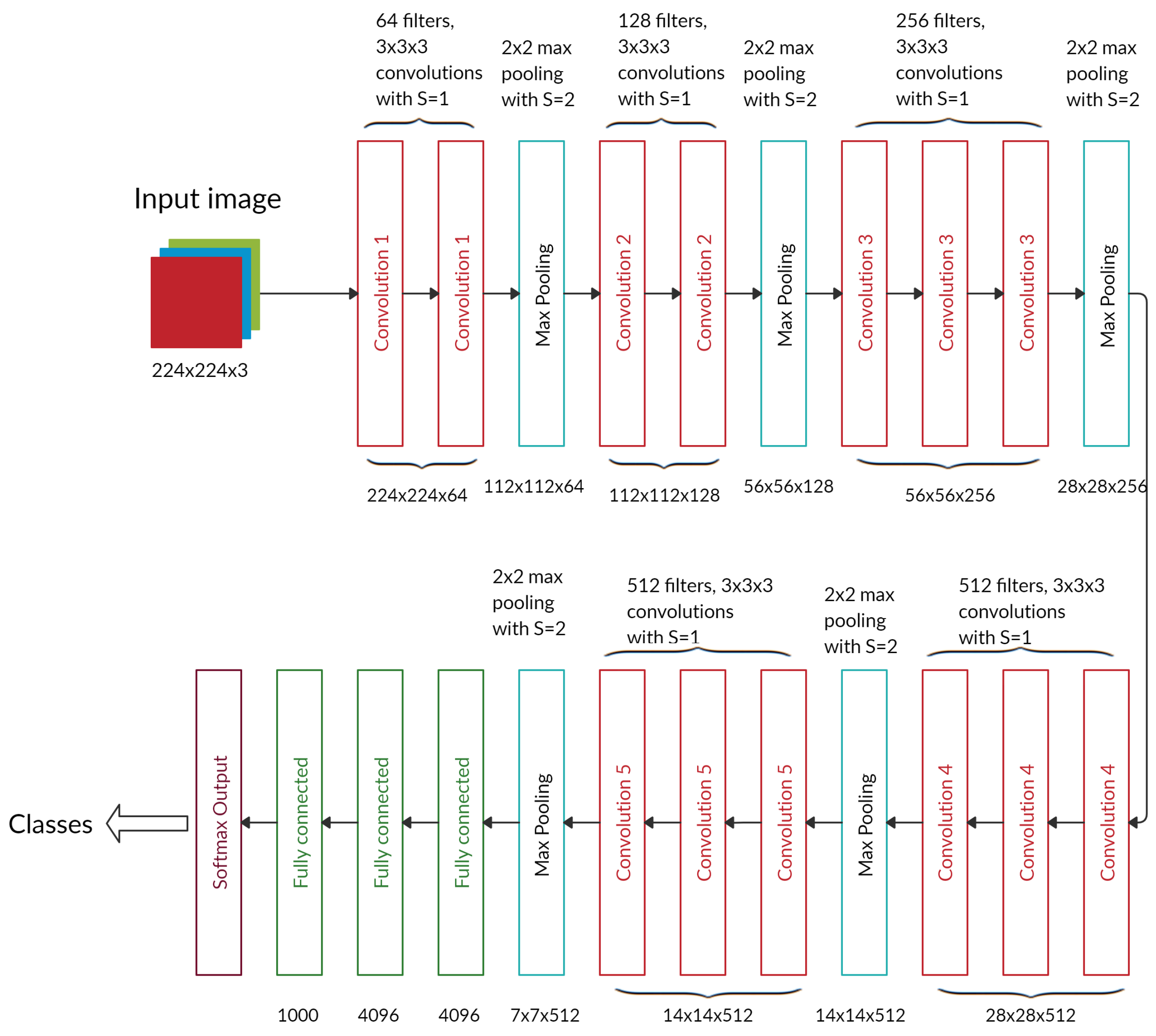


Figure 3.11: VGG-16 configuration

3.5 Conclusion

In this chapter, we started by discussing the history of Artificial Neural Networks, then we detailed Convolutional Neural Networks and their main parameters. We also presented VGG-16, which is the adopted CNN model for writer identification from a single handwritten word. In the next chapter, we will present the results of the VGG-16 model, the descriptor-based methods presented in chapter 2 and eventually the results of the fusion between the extracted features from VGG-16 and MLOG to improve writer identification from a single handwritten word.

Chapter 4

Experimental results

4.1 Introduction

This chapter is dedicated to analyse the performance of different implemented systems for writer identification using single handwritten words. Precisely, the first part of experiments discusses and compares descriptor-based systems, namely pixel distribution, Gradient Local Binary Patterns (GLBP) and the proposed Multiscale Local Oriented Gradient (MLOG), during which, the best order derivative of MLOG is identified. Then after, MLOG performances are investigated for both writer-dependent and writer-independent approaches. In a second part, the performance of the adopted CNN-based system, namely, VGG-16 is evaluated for writer-dependent approach. Finally, features generated from MLOG descriptor and the configured VGG-16 are fused to form hybrid features, which are fed to SVM classifier for more robust writer identification from single handwritten words.

Experiments are conducted on two benchmark datasets using Python environment to which, several libraries of neural networks and image processing have been associated, namely, Keras, TensorFlow and OpenCV. Tests are carried out on an Intel Core I7 CPU (2 x 2.5GHz) with 8GB RAM and a GPU provided by Google Colaboratory for the CNN implementation.

4.2 Datasets

In this work, writer identification using single handwritten words is addressed in a supervised classification manner, hence, labeled data are required. Therefore, we adopted two benchmark datasets, namely, ICDAR2011 and IAM presented below. These two datasets are collected in a multi-script and multi-writer environment.

4.2.1 ICDAR2011 dataset

ICDAR2011[1] handwriting dataset was offered as a part of the writer identification contest organized by the International Conference on Document Analysis and Recognition " ICDAR " in 2011. This dataset involves 26 writers, each contributed with eight text documents in four different languages : English, French, German and Greek. Another version of this dataset exists, which is the cropped ICDAR2011, that contains the same documents, but only the first two text lines from each document are taken. In this work, we used the cropped ICDAR2011, which needed to be segmented to words as explained previously, yielding into 25 words for each writer in each of the four languages. The handwritten words are saved as TIF binary images. Figure 4.1 shows examples of ICDAR2011

[1]**users.iit.demokritos.gr/~louloud/Writer˙Identification˙Contest/index.html**.

cropped images.
In our experiments we perform language-independent and language-dependent writer identification. Language-dependent considers each language as an individual set, namely, German (DE), English (EN), French (FR) and Greek (GR). As to language-independent experiments, two corpora were collected, a Latin alphabet-based "ICDAR-3", composed of German, English and French, giving 75 words per writer. The second set "ICDAR-4", considers all four languages mixed, giving 100 words per writer. As for most of classification and data-mining tasks as well as writer identification, 2/3 of the samples are used for the training step, while the remaining 1/3 are used for testing the system.

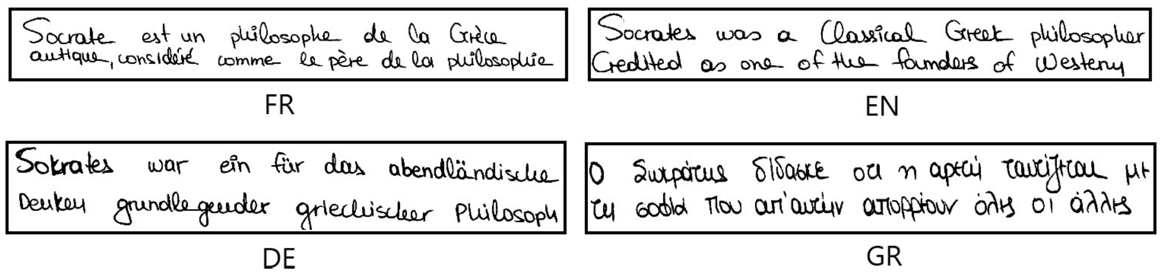


Figure 4.1: Examples of ICDAR2011 cropped images

4.2.2 IAM dataset

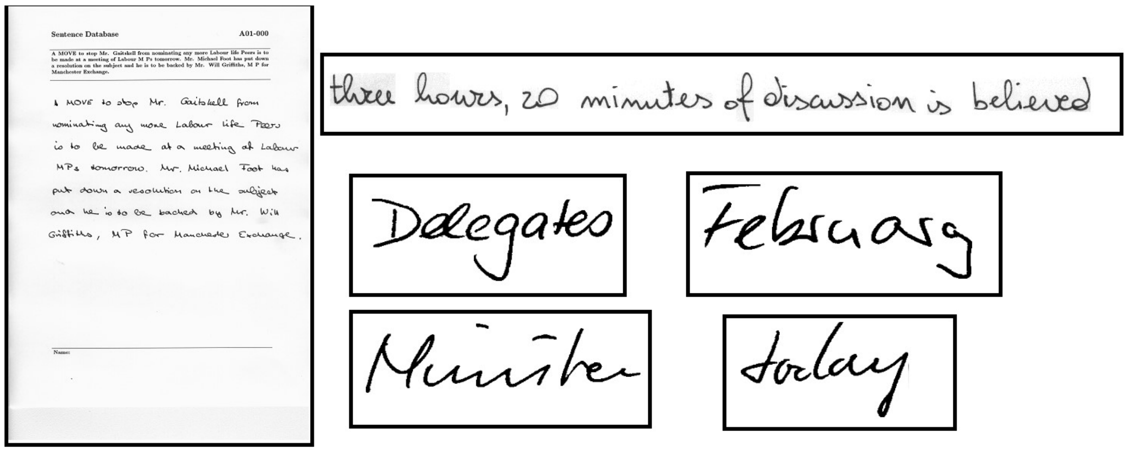
IAM[2] is a handwriting dataset that contains forms of handwritten English text (see Figure 4.2), which can be used for several tasks including writer identification. It was first published at the ICDAR 1999 contest. The database contains forms of unconstrained handwritten text, which are scanned at a resolution of 300dpi and saved as PNG images with 256 gray levels. Therefore, we needed to perform binarization of the given words.

IAM 3.0 is the latest available version. It is structured as follows:

- 657 writers
- 1539 pages of scanned text
- 5685 isolated and labeled sentences
- 13353 isolated and labeled text lines
- 115320 isolated and labeled words

[2]**fki.inf.unibe.ch/databases/iam-handwriting-database**.

In our work, two corpora are collected to perform writer identifiation from single handwritten words. The first corpus ”IAM-1”, contains 100 writers that contribute each with 310 words. The second corpus ”IAM-2”, contains also 100 writers but by collecting 60 words for each writer. Samples in each corpus are divided in 2/3 for training and 1/3 for testing.



Sentence Database A01-000

A MOVE to stop Mr. Gaitskell from nominating any more Labour life Peers is to be made at a meeting of Labour M Ps tomorrow. Mr. Michael Foot has put down a resolution on the subject and he is to be backed by Mr. Will Griffiths, M P for Manchester Exchange.

A MOVE to stop Mr. Gaitskell from nominating any more Labour life Peers is to be made at a meeting of Labour MPs tomorrow. Mr. Michael Foot has put down a resolution on the subject and he is to be backed by Mr. Will Griffiths, MP for Manchester Exchange.

Name:

three hours, 20 minutes of discussion is believed

Figure 4.2: Some examples of IAM handwritten images (text, line, words)

Table 4.1 summarizes the data distribution for all corpora of ICDAR2011 and IAM datasets for the writer-dependent approach.

	IAM-1	IAM-2	ICDAR (DE, EN, FR, GR)	ICDAR-3	ICDAR-4
Number of training samples	210	40	17	51	68
Number of testing samples	100	20	8	24	32

Table 4.1: Data distribution for writer-dependent approach

4.3 Software, libraries and hardware

In this section, we provide a brief overview of the programming language and software tools employed in our work, namely, Python[3], Keras[4], TensorFlow[5], OpenCV[6], along with the hardware used.

4.3.1 Python

Python is an interpreted, object-oriented, high-level open source programming language created by Guido van Rossum and released in 1991. It emphasizes readability through its simple, easy to use syntax, and thus reduces the cost of program maintenance. Python supports numerous very stable modules and packages, making it an attractive and versatile language that can be used in many areas, such as, web development, data analysis and the increasingly popular fields of Machine Learning and Artificial Intelligence. Python is also a very powerful programming language, which opens the way widely to complex project development. In our work, we used Python version 3.6.

4.3.2 Keras

Keras is an open source high level python library implemented to support Deep Neural Networks (DNN). It provides a convenient interface for users to manipulate DNN by leveraging other libraries such as Theano, CNTK, and TensorFlow. This allows it to handle multidimensional data tables. For our application, we have associated Keras to TensorFlow. In our work, we used Keras version 2.3.1.

[3]**python.org/**.
[4]**https://keras.io/**.
[5]**tensorflow.org/**.
[6]**opencv.org/**.

4.3.3 TensorFlow

TensorFlow is an open source library originally developed by researchers from the Google Brain team for conducting research on Machine Learning and Deep Neural Networks. It provides workflows and differentiable programming tools, and allows a manipulation on different levels of abstraction. It can be built on Python, Javascript and Swift, and is easily deployed on CPUs (Central Processing Unit), GPUs (Graphics Processing Unit) and browsers. The version used in this work is TensorFlow 2.2.0.

4.3.4 OpenCV

OpenCV (Open Source Computer Vision) is an open source library that proposes more than 2500 optimized algorithms dedicated to computer vision, machine learning and image processing applications. It is adapted for use in many languages including C++ and Python. The version used in this work is OpenCV 4.2.0.

4.3.5 Hardware

Conducted tests in this work were mostly on an Intel Core I7 CPU (2x2.5GHz) with a RAM of 8Go. However, an Intel Xeon four processors (2x2.2GHz) 25 Go RAM virtual machine, backed with 16GB NVIDIA Tesla P100 GPU provided by Google Colaboratory was used for the CNN part of our work. The virtual machine is exploitable up to 12 hours continuously and its GPU sped up the processing by more than 35 times compared to the normal CPU usage.

4.4 Evaluation criteria

In order to evaluate the performance of our writer identification systems, two evaluation measures have been used, so called classification accuracy and TOP-k. The former was used in the evaluation of the writer-dependent approach, whereas the latter was employed to evaluate the writer-independent approach.

4.4.1 Classification accuracy

When the term ”accuracy” is used, it usually refers to classification accuracy, which is defined as the ratio of the number of correct predictions made by the model in question over the the total number of predictions. Let $\hat{L_i}$, L_i and N_s be the predicted class of the test sample i, the correct class of the test sample i and the number of samples respectively. We can express the accuracy mathematically with the following equation :

$$Accuracy = \frac{1}{N_s} \sum_{i=1}^{N_s} f\left(\hat{L}_i, L_i\right) \tag{4.1}$$

Where $f(a, b)$ is defined as :

$$f(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases} \tag{4.2}$$

4.4.2 TOP-k

This evaluation metric considers that the model has successfully predicted the writer of the query word, if at least the correct writer appears among the TOP-k predicted writers. Let Q_i be the query word labeled L_i, and let R_j $(j = 1, 2, ...N_R)$ be the reference word with N_R the number of references. We define $P_{i,j}$ as the probability of Q_i and R_j belonging to the same writer. We thus obtain a vector of probabilities $P_i = [P_{i,1}, P_{i,2}, P_{i,3}, P_{i,N_R}]$. These probabilities are then ordered decreasingly from the highest to the lowest probability. We consider that the writer of Q_i is correctly predicted if L_i exists among the labels of the TOP-k ordered probabilities.

4.5 Experimental protocol

Writer identification from a single handwritten word is addressed through different approaches; writer-dependent and writer-independent, but also through different systems, which are descriptor-based system and CNN-based system. Hence, experimental protocols are different and are explained in what follows.

4.5.1 Descriptor-based system

The descriptor-based system extracts features and feeds them to SVM classifier. These features in question are calculated using pixel distribution, GLBP and MLOG descriptors. Each of these descriptors needs some tuning. Indeed, pixel distribution and GLBP are locally computed by applying a uniform grid over word images. The uniform grid divides the input image into a number of predefined cells, computes the features for each cell, and then proceeds to the concatenation of all cell features to form the image feature vector. In our experiments, the grid size was varied from 1x1 to 40x40 cells for computing pixel distribution, and from 1x1 to 4x8 cells for GLBP features. Furthermore, experiments were conducted to rise MLOG performance by choosing the suitable order directional derivatives, along with the weights given to sets B and C (refer to in chapter 2). The weights were varied in the range [0.05 : 0.05 : 1]. To evaluate SVM classifier for writer identification, RBF kernel was used, and tests were executed by varying the regularization parameter C in the range $[0.01, 10000]$, while the RBF kernel sigma was varied in the range $[10^{-9}, 100]$. In each test, the couple giving the best training accuracy was selected. As to the writer-independent approach, in the training phase, the extracted features are used to create the dissimilarity space according to the transformation presented in chapter 2. Indeed, within-class dissimilarities (+) are computed between all possible pairs of references for the same writer, while between-class dissimilarities (-) are calculated between all possible pairs of references for different writers. This yields, obviously, to a negative class that is much more represented than the positive class. Hence, to overcome this unbalanced problem, a number of negative dissimilarities are chosen randomly according to the number of positive dissimilarities, since the representation of the negative class is less important. In the testing phase, given a query image Q_i belonging to writer W_i, we compute the distance between its feature vector, and each of the reference samples of each writer, including writers that did not take part in the training phase. The resulting distances are fed to the previously trained SVM model, giving as output, a probability vector. The performance is evaluated based on TOP-k metric. Note that, in order to compute the dissimilarities, a comparative study is conducted using Euclidean, Manhattan, Canberra and Chi-square distances.

4.5.2 CNN-Based system

The adopted CNN-based system is a modified version of the VGG-16 configuration built for ImageNet dataset. Instead of input images of size $224 * 224 * 3$, we use normalized $64 * 64 * 3$ images. As for the Fully Connected layers, we adopt the configuration depicted in Figure 4.3. The size of the first layer H is set to 512 neurons for IAM-1, IAM-2, ICDAR-3 and ICDAR-4. As to ICDAR individual language corpora, H is set to 1024 neurons.

Furthermore, the second FC layer is set to 128 neurons. The last FC layer represents the number of classes M, either 100 for IAM or 26 for ICDAR corpora. The activation functions employed are ReLU, ELU and Softmax for the first, second and third layer respectively.

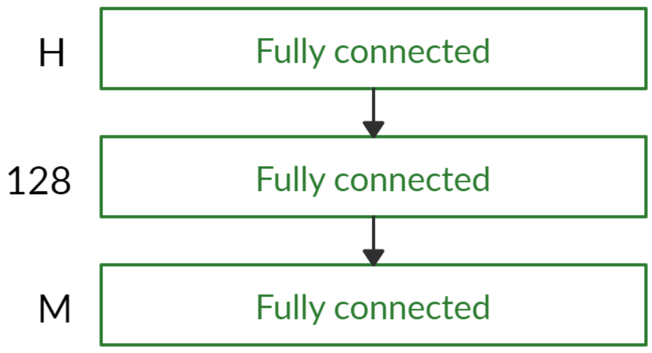


Figure 4.3: The adopted Fully Connected layers

Using "Transfer Learning" technique, our model starts the training process using the weights provided from the pre-trained ImageNet model. The last 4 convolutional layers remain trainable, while dropout of 20% for IAM and 50 % for ICDAR is employed between the first two FC layers in order to reduce overfitting. Along with the activation functions of the FC layers, we also determine other parameters experimentally, which are:

- ⋆ Learning rate (Lr); set to 5e-6 for both IAM corpora and 1e-6 for all ICDAR corpora.
- ⋆ Optimizers (opt): RMS (Root Mean Square) optimizer for both IAM corpora and Adam optimizer for all ICDAR corpora.
- ⋆ Batch size (bt): set to 10 for all corpora.
- ⋆ Number of epochs: varied in the range [30,500].

4.6 Results of the descriptor-based system

This section presents descriptor-based system results for writer identification from a single handwritten word for both approaches; writer-dependent and writer-independent. Indeed, it investigates several influent parameters such as MLOG order directional derivative, the distance used for the dissimilarities computation, the number of training writers, and the number of references per writer.

4.6.1 Writer-dependent results

First, MLOG order directional derivative needs to be tuned along with the weights given to each set B and C, then MLOG performance is compared to pixel distribution and GLBP.

■ MLOG order experiment

In order to define the most suitable n^{th} order directional derivative of MLOG for writer identification from single handwritten words, different orders are tested from 0 to 10, along with the weights assigned to sets B and C, which are varied in the range [0.05:0.05:1]. Experiments are conducted for all corpora, since similar behavior is observed, only results obtained using IAM-2 corpus are presented in Figure 4.4. As we can see, accuracy inceases from 45,25% for the 1^{st} order, reaching a maximum accuracy of 63.70% for the 5^{th} order before decreasing to 54,10% for the 10^{th} order derivative. Hence, the 5^{th} order directional derivative seems to be the most appropriate to characterize the handwriting style of the writer. Note that, for this order value, weights of sets B and C are set to 0.7 and 0.3, respectively.

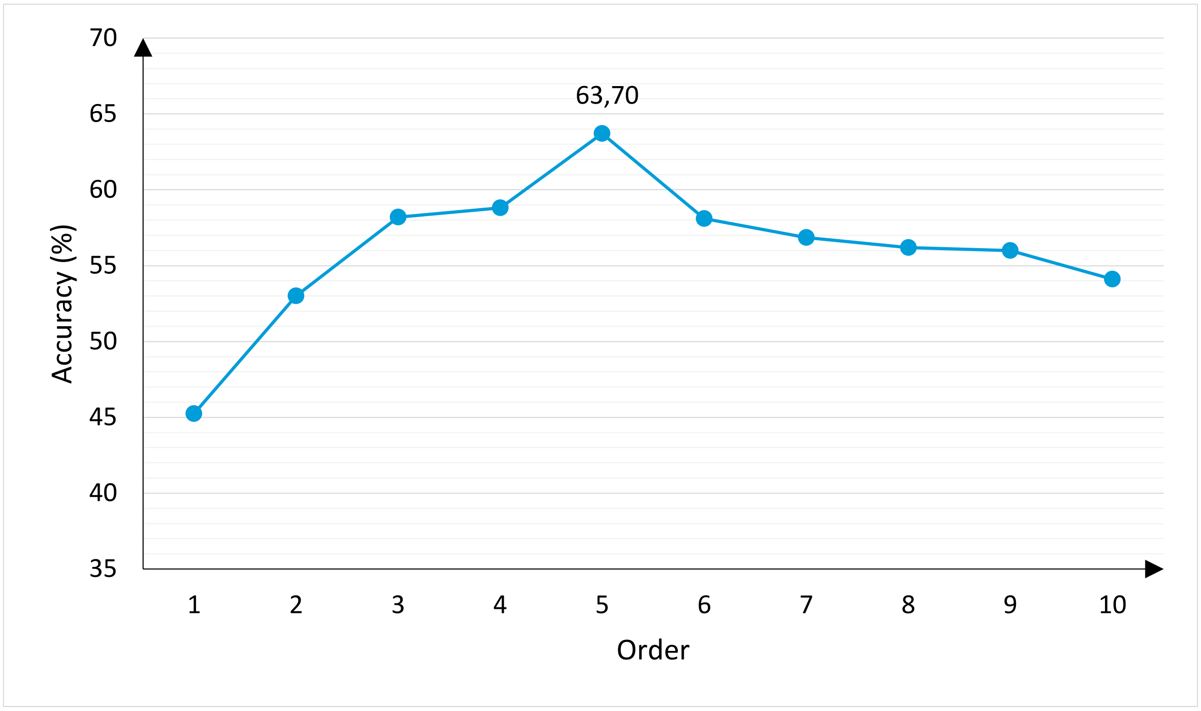


Figure 4.4: MLOG order directional derivative variation

■ Descriptors comparison results

Once MLOG order is defined, its performance is compared to two descriptors that have proven their efficiency in many applications related to handwriting. Note that the results of pixel distribution and GLBP given in Table 4.2, are those obtained for the best grid size configuration. Recall that writer identification from single handwritten words, is a very challenging task, and the obtained results testify. Indeed, pixel distribution accuracies are very low for all corpora. GLBP outperformes pixel distribution with accuracies that vary between 37.72% for Greek language and 47.97% for IAM-1 corpus, but remains insufficient. Nevertheless, MLOG outperformes both pixel distribution and GLBP, where its accuracies vary from 61.53% for Greek language to 76.94 % for IAM-1, allowing a gain that varies between 17.89% to 29.71% compared to GLBP. The reason for which Greek language performances are always the lowest, seems to be that, Greek alphabets are more complicated to characterize than Latin alphabets. On the other hand, IAM-1 provides the best accuracies because it contains much more data than the other corpora. Indeed, each writer is represented with 210 words during training phase, so with that many information, the handwriting style of the writer can be captured.

	IAM-1	IAM-2	DE	EN	FR	GR	ICDAR-3	ICDAR-4
Pixel-D	30.77	20.25	28.25	23.21	16.83	19.23	27.94	22.50
GLBP	47.97	39.90	42.30	44.61	38.46	37.72	42.56	40.96
MLOG	**76.94**	**63.70**	**70.19**	**62.50**	**62.05**	**61.53**	**72.27**	**70.31**

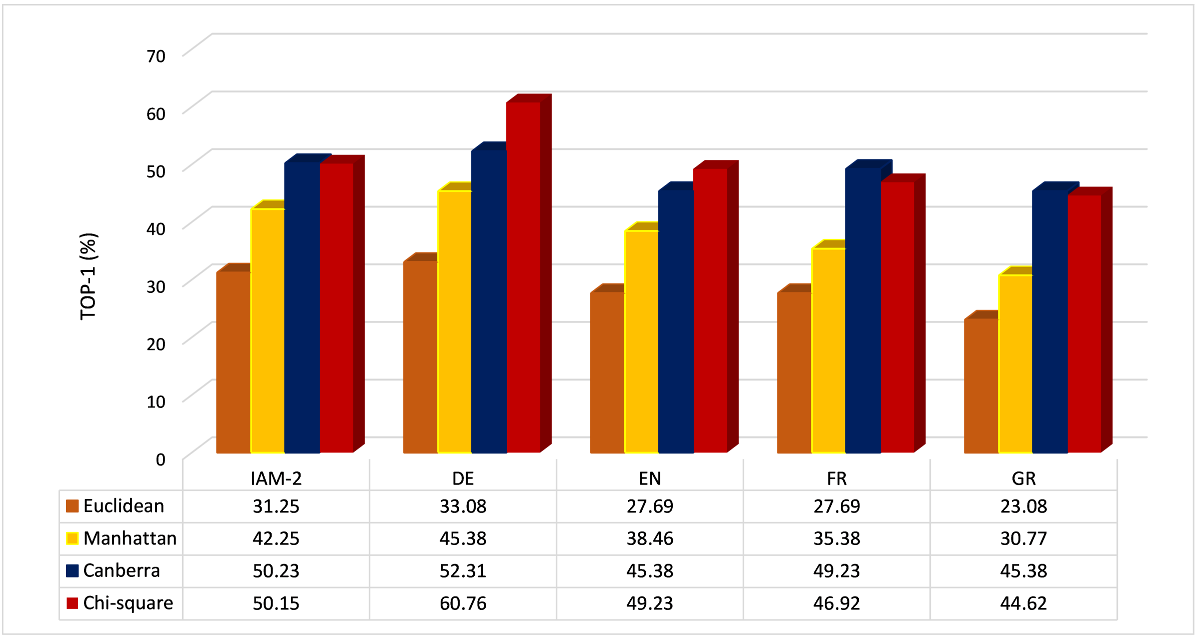
Table 4.2: Descriptor-based system results for writer-dependent approach (%)

4.6.2 Writer-independent results

Once the performance of MLOG in writer identification from single handwritten words in writer-dependent approach is established, it is only fair to investigate writer-independent approach using this particular descriptor. In addition, in this approach, three important parameters need to be discussed, which are the influence of the distance in dissimilarities computation, the influence of the number of writers and the number of references per writer in the training stage.

- **Distance influence in dissimilarities computation**

Four distances that are Euclidean, Manhattan, Canberra and Chi-square are investigated in the dissimilarities calculation for writer-independent approach to determine the most suitable one. Experiments are carried out for all corpora. The results in terms of TOP-1 are given in Figure 4.5 for IAM-2 and ICDAR individual languages. Results show that Euclidean and Manhattan distances are far behind Canberra and Chi-square distances. These last two distances, behave similarly. Indeed, for TOP-1, Canberra distance is slightly better for IAM-2, with 50.23 %, against 50.15 % provided by Chi-square distance. As for ICDAR individual languages, Chi-square offers an improvement that varies between 3.85 % for English language and 8.45 % for German language. Whereas, Canberra distance offers a gain that varies between 0.76 % and 2.31 % for Greek and French languages, respectively. Roughly, we can conclude that Canberra distance is more adapted to IAM dataset, while Chi-square distance is more suitable for ICDAR dataset.



	IAM-2	DE	EN	FR	GR
🟧 Euclidean	31.25	33.08	27.69	27.69	23.08
🟨 Manhattan	42.25	45.38	38.46	35.38	30.77
🟦 Canberra	50.23	52.31	45.38	49.23	45.38
🟥 Chi-square	50.15	60.76	49.23	46.92	44.62

Figure 4.5: Distance influence on the writer-independent approach

■ **Influence of the number of training writers**

Recall that, the independent approach carries the advantage of identifying writers that were not involved in the training process of the model. Therefore, we investigate the number of writers necessary in the training phase to learn at best the boundaries of within-class dissimilarities (+) and between-class dissimilarities (-). Experiments are conducted on all corpora, but as for IAM corpora no influence is observed whatsoever, we present only the impact of the number of writers for ICDAR corpora using Chi-square distance. Results are reported in Figure 4.6. Tests were carried out using 15 references per writer for each individual language, 45 (15 x 3) references per writer for ICDAR-3 and 60 (15 x 4) references for ICDAR-4. Obtained results demonstrate that no significant influence is noticed regardless to the corpus. In light of these outcomes, it is clear that, using MLOG descriptor, the number of training writer has no impact. These outcomes testify for the power of MLOG descriptor associated with the right distance to characterize handwriting style, such that the boundary between positive class (+) and negative class (-) in the dissimilarity space, does not rely on the number of training writers.

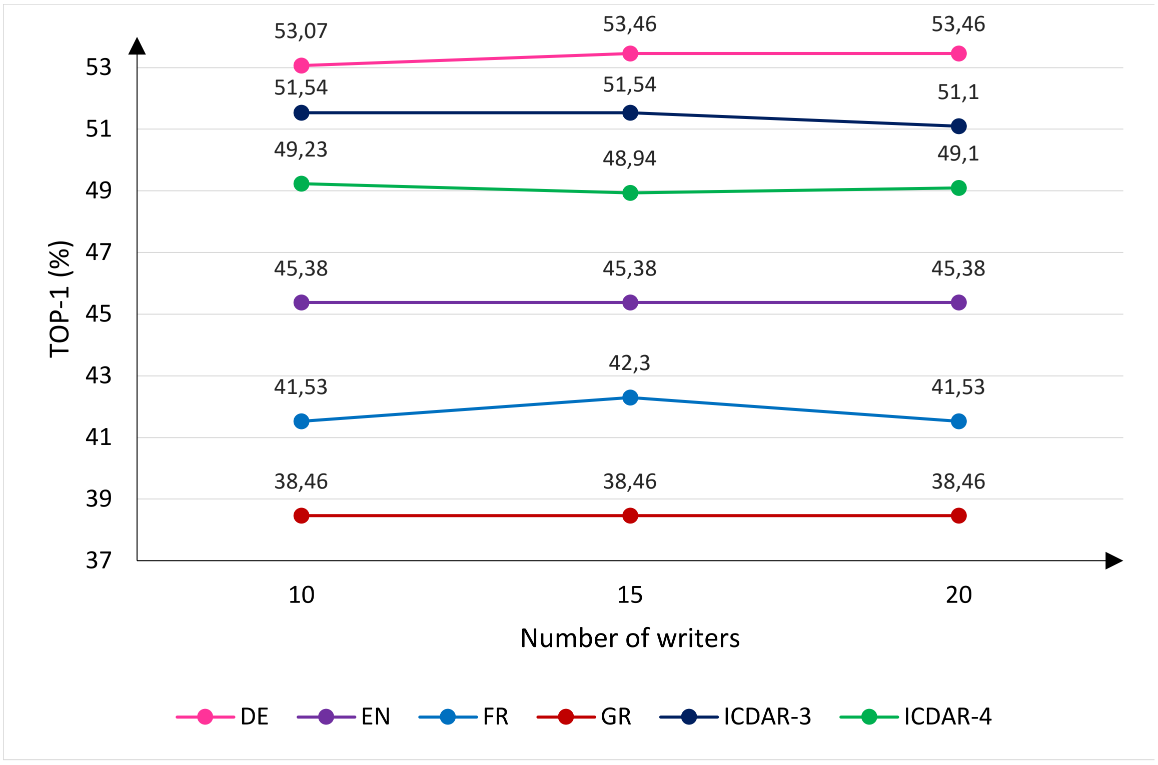
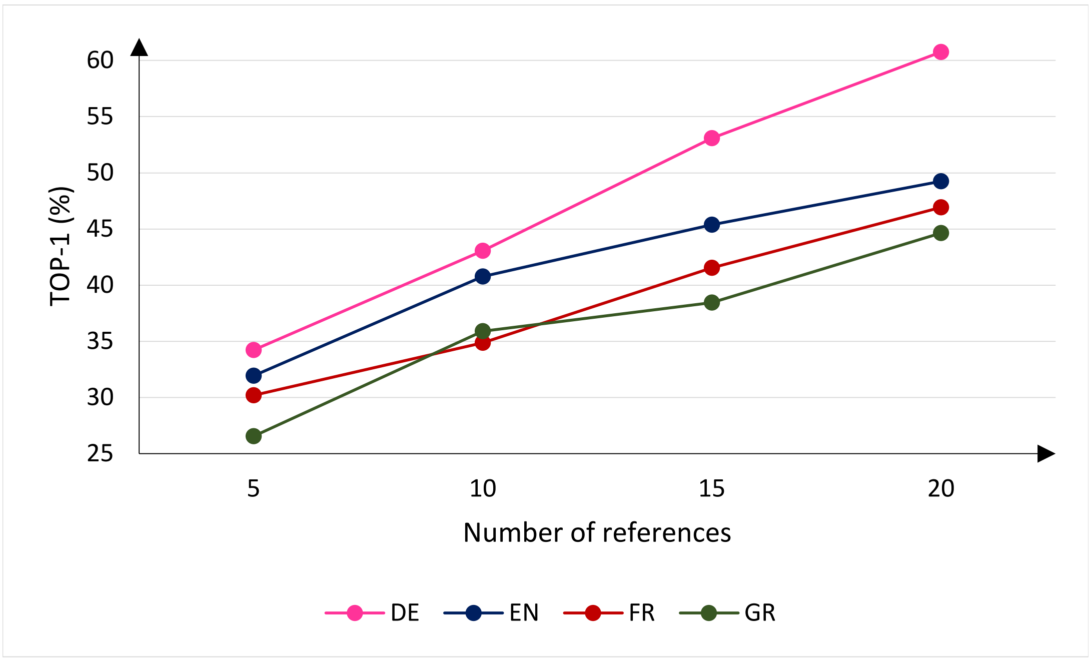


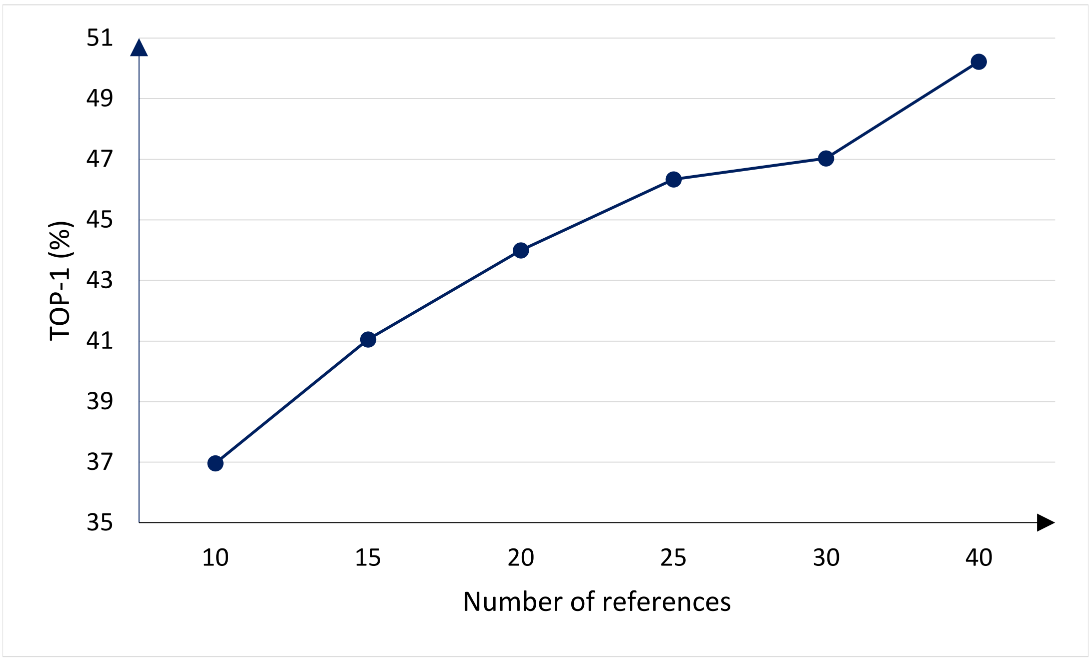
Figure 4.6: Influence of the number of writers using Chi-square distance

■ Influence of the number of references per writer

After the choice of the distance used and the investigation of the impact of the number of writers, which is ultimately not one, comes the investigation of the influence of the number of references per writer. For that, experiments are conducted by fixing the number of training writers to 10 for all corpora and varying the number of references according to data availability in each corpus. Similar behavior is observed, but once again to lighten the reading, only results for IAM-2 and ICDAR individual languages are given in terms of TOP-1 in Figure 4.7. Findings show that, the higher the number of references, the higher TOP-1 is. For example, for ICDAR DE, TOP-1 is at 34.23% using only 5 references and reaches 60.76% using 20 references. Another example, for IAM-2, TOP-1 is at 36.96% using only 10 references and reaches 50.23% using 40 references per writer. In light of these outcomes, it is clear that the number of references has an important impact on the prediction performance. This is explained by the fact that increasing the number of references, minimizes the variability within a writer class. Since more information on the handwriting style of the writer is collected, it is more likely that the query image will be correctly attributed to its writer. Note that, for the rest of the experiments, the highest reference number is retained since it gives the best performance.



(a) ICDAR



(b) IAM-2

Figure 4.7: Influence of the number of references per writer

After a long investigation on different factors that can impact the performance of writer identification from single handwritten words, Tables 4.3 and 4.4 summarize the best performances reached from TOP-1 to TOP-10 for each corpus, using Canberra and Chi-square distances respectively, in the case of writer-independent approach. From the examination of these tables, three outcomes need to be highlighted: first, Chi-square distance performs better for corpora IAM-1, ICDAR DE, ICDAR EN, ICDAR-3 and ICDAR-4, while Canberra distance shows a better performance for corpora IAM-2, ICDAR FR and ICDAR GR. Secondly, the number of writers used in training phase does not have any influence. Last but not least, TOP-1 performances vary from 44.62% for ICDAR GR to 60.76% for ICDAR DE, and at best, a performance of 58.10% for IAM-1. Note that, from TOP-5, performances are rather satisfying. One can say that, these performances are far from what we are used to see in several handwriting applications, especially writer identification from document. However, keep in mind that the tackled task of writer identification from single handwritten words is very complex and challenging.

TOP-k	1	2	3	4	5	6	7	8	9	10
IAM-1	53.5	65.9	73.5	76.7	79.5	81.4	82.9	83.8	84.7	85.4
IAM-2	**50.23**	63.6	70.05	74.75	77.85	80.2	82.05	83.4	84.9	86.2
DE	52.31	64.61	74.62	80.77	83.84	84.61	86.15	89.23	90.77	90.77
EN	45.38	59.23	66.92	71.54	77.69	82.31	86.15	87.69	90.00	91.54
FR	**49.23**	59.23	65.38	67.69	71.53	77.69	79.23	82.3	83.84	85.38
GR	**45.38**	53.84	62.3	66.92	69.23	73.07	76.92	77.69	80.76	83.07
ICDAR-3	52.82	65.12	73.84	79.48	82.05	83.59	86.15	87.43	88.46	89.74
ICDAR-4	50.38	63.65	71.15	76.92	80.00	81.73	85.00	86.34	87.69	89.04

Table 4.3: Writer-independent results for MLOG using Canberra distance (%)

TOP-k	1	2	3	4	5	6	7	8	9	10
IAM-1	**58.10**	69.00	73.60	76.50	79.70	81.30	82.80	84.20	84.70	85.50
IAM-2	50.15	62.70	68.90	73.70	76.40	78.05	80.90	82.40	83.75	84.55
DE	**60.76**	71.53	78.46	81.53	83.84	85.38	87.69	90.76	91.53	92.30
EN	**48.46**	65.38	72.31	79.23	80.77	83.08	86.15	86.92	89.23	89.23
FR	46.92	60.76	67.76	69.23	72.30	76.92	79.23	80.00	80.00	83.84
GR	44.62	51.54	61.54	66.15	68.46	70.77	74.62	76.15	78.46	81.54
ICDAR-3	**56.15**	67.95	75.38	79.74	82.82	85.38	86.92	87.69	87.69	89.23
ICDAR-4	**53.46**	64.42	73.07	79.23	81.92	84.42	85.96	86.92	87.88	89.42

Table 4.4: Writer-independent results for MLOG using Chi-square distance (%)

4.7 Results of the CNN-based system

Convolutional Neural Network VGG-16 model as presented in section 4.5.2 is used as a feature extractor. However, this architecture was first tested on writer-dependent approach to see how it performs on such a task. Results are reported in Table 4.5. The first major drawback of using CNN, is the need for very large amount of data in the training stage to perform well. Recall that, for each writer, IAM-1 contains 210 training samples, IAM-2 contains 40 training samples, each ICDAR individual language contains 17 training samples, mixed ICDAR-3 contains 51 training samples, and finally mixed ICDAR-4 contains 68 training samples. As can be observed, the amount of data is very small, especially for ICDAR individual languages. Obtained results show very poor accuracies compared to those obtained using MLOG descriptor, except for IAM-1 corpus which gave an accuracy of 72.94% against 76.94% using MLOG. As to other corpora, a loss that varies between 13.62% and 30.77% is observed, compared to MLOG descriptor. This loss is huge for writer-dependent protocol, however, it is explained by the lack of data, and probably due to the last output FC layer. Nevertheless, in real-world applications, it proves to be difficult to have large datasets. In order to overcome these limitations, the architecture of a hybrid CNN–SVM model is proposed, by replacing the last output FC layer of the VGG-16 model with an SVM classifier. Output units of the last FC layer in the CNN are the estimated probabilities for the input data. These values can be treated as features for any other classifiers. Moreover, hoping that CNN will perform better, ICDAR individual languages were tested on the model of mixed ICDAR-3 or ICDAR-4, since they contain more data. Obtained results given in Table 4.5 show an improvement that varies from 0.97% to 21.15%. The gain for ICDAR individual languages is more than 14%, which is not negligible at all. However, obtained performances are still far from those obtained using MLOG descriptor. Consequently, we propose to fuse MLOG features and CNN features to form hybrid features that will be fed as an input vector to SVM classifier. This feature fusion will be discussed in the next section.

	IAM-1	IAM-2	DE	EN	FR	GR	ICDAR-3	ICDAR-4
CNN	72.94	49.00	39.42	40.38	34.62	31.17	**58.65**	52.40
CNN-SVM	**73.60**	**53.35**	**60.57**	**54.32**	**53.85**	**49.52**	55.76	**53.37**

Table 4.5: Writer-dependent results for CNN-based system (%)

4.8 Hybrid features fusion to improve writer identification

Hybrid features system combines features of different nature. Indeed, it fuses MLOG features with CNN features from the second FC layer that provides a feature vector of 128 elements. Since these features are of different nature, they do not stand in the same range, hence a two normalization step of the 128 CNN features is needed. First, a normalization in the range $[0, 1]$ is applied as follows:

$$X_1 = \frac{X - X_{min}}{(X_{max} - X_{min}) * (max - min)} + min \tag{4.3}$$

Where min and max correspond to the new range [0,1], and X corresponds to the input 128 feature vector.
Second, the resulting normalized CNN features are given an experimentally determined weight, varied in the range $[0.0035, 0.0075]$. Consequently, a feature vector of 296 values is obtained, and is referred to as "hybrid feature vector". Figure 4.8 illustrates the process of the fusion of hybrid features. Note that this hybrid feature is used in both writer-dependent and writer-independent approaches.

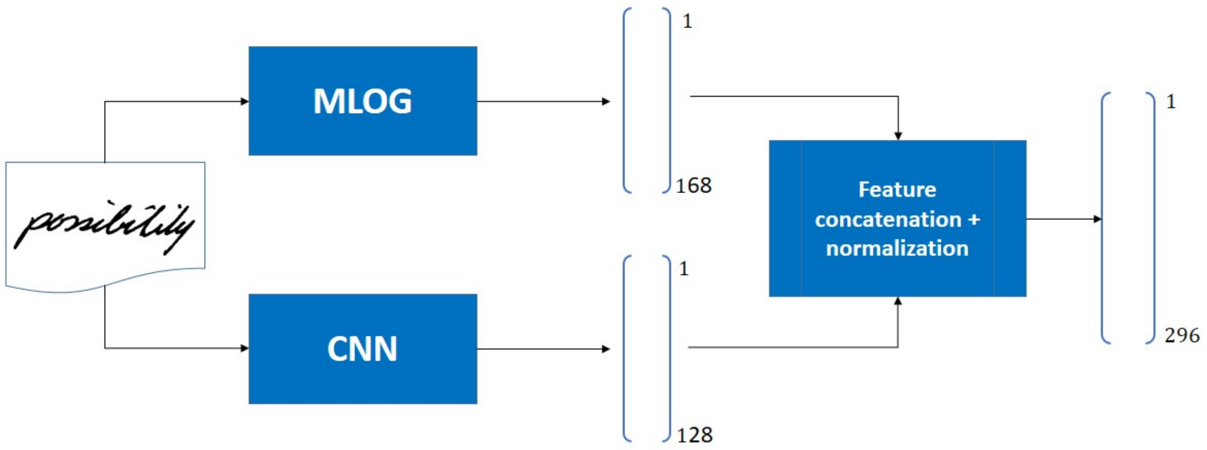


Figure 4.8: Hybrid features fusion

4.8.1 Writer-dependent approach

Hybrid features are fed to SVM classifier, where experiments are conducted on all corpora, Table 4.6 reports the obtained results, while Figure 4.9 highlights MLOG performance (so far the best) against hybrid features performance for writer dependent approach. As can be seen, IAM accuracies have exceeded 80% while ICDAR accuracies exceeded 70% except for Greek language. Furthermore, from Figure 4.9, it can be seen that hybrid features allow an improvement of more than 17.5% for IAM corpora, when for ICDAR individual languages, an improvement of more than 7.5% is noticed except for ICDAR Greek language, where a gain of only 1.41% is observed. As to blended datasets, improvements are respectively 13.46% and 5.06% for ICDAR-3 and ICDAR-4.

IAM-1	IAM-2	DE	EN	FR	GR	ICDAR-3	ICDAR-4
94.72	82.70	77.88	75.48	71.15	63.46	85.73	75.36

Table 4.6: Writer-dependent accuracies for hybrid features (%)

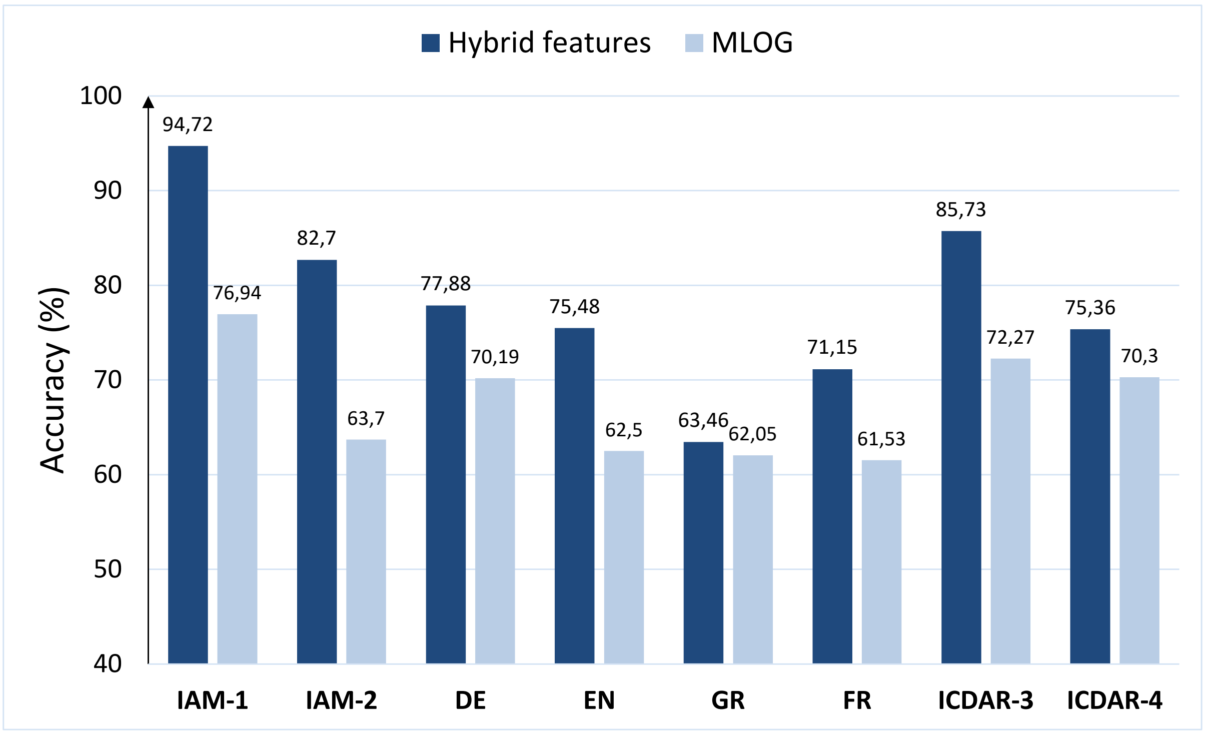


Figure 4.9: Hybrid features vs MLOG performances for writer-dependent approach

4.8.2 Writer-independent approach

Tables 4.7 and 4.8 report TOP-1 to TOP-10 accuracies for all corpora with both Canberra and Chi-square distances. Again, we can observe that the fusion greatly enhanced the performance on the writer-independent approach. For most corpora, Chi-square distance outperforms Canberra distance, except for ICDAR English and French. From TOP-5, almost all corpora reached 90%, which is very promising for such a complicated task. Figure 4.10 highlights a comparison between MLOG TOP-1 and hybrid features TOP-1 for writer-independent approach. Hybrid features allow a gain that varies between 4% and 8.5% for ICDAR Greek and French, respectively, while an improvement of about 10% is noticed for ICDAR German and ICDAR-4. However, the most impressive improvement, which is more than 20% is noted for ICDAR English and IAM-2, reaching a maximum gain of 29.2% for IAM-1 corpus.

TOP-k	1	2	3	4	5	6	7	8	9	10
IAM-1	86.80	92.60	95.10	95.70	96.30	96.80	97.00	97.30	97.70	98.30
IAM-2	72.90	82.70	86.60	89.15	90.95	92.10	93.05	93.60	94.05	94.85
DE	68.46	81.53	87.69	87.69	89.23	89.23	82.30	93.07	93.09	94.61
EN	**71.53**	80.76	84.61	86.92	87.69	92.30	92.30	93.84	94.61	95.38
FR	**57.69**	68.46	78.46	82.30	84.61	86.15	90.00	91.53	92.30	93.07
GR	49.23	62.30	73.84	77.69	80.00	83.84	86.92	90.00	91.53	93.07
ICDAR-3	73.33	81.28	85.38	87.17	89.48	90.51	92.05	93.07	94.10	94.61
ICDAR-4	60.38	72.59	78.12	81.73	84.49	86.89	88.46	89.42	90.63	90.98

Table 4.7: Writer-independent results for hybrid features using Canberra distance(%)

TOP-k	1	2	3	4	5	6	7	8	9	10
IAM-1	**87.30**	92.70	94.90	95.90	96.60	97.10	97.60	97.70	97.90	98.00
IAM-2	**73.45**	82.90	86.80	89.05	90.90	92.10	92.70	93.30	93.93	94.00
DE	**70.76**	83.07	85.38	88.46	90.00	91.53	92.30	93.07	96.15	96.15
EN	66.92	76.15	81.53	86.15	90.00	90.00	91.53	92.30	95.38	96.92
FR	55.38	69.23	74.61	83.07	85.38	87.69	92.30	93.07	93.84	93.84
GR	**50.00**	64.61	72.30	77.69	82.30	85.38	86.92	89.23	92.30	92.30
ICDAR-3	**75.12**	85.12	90.76	92.05	93.07	94.87	95.38	95.64	95.89	96.67
ICDAR-4	**63.07**	73.07	78.26	83.27	84.80	88.07	89.03	90.00	90.96	91.53

Table 4.8: Writer-independent results for hybrid features using Chi-square distance(%)

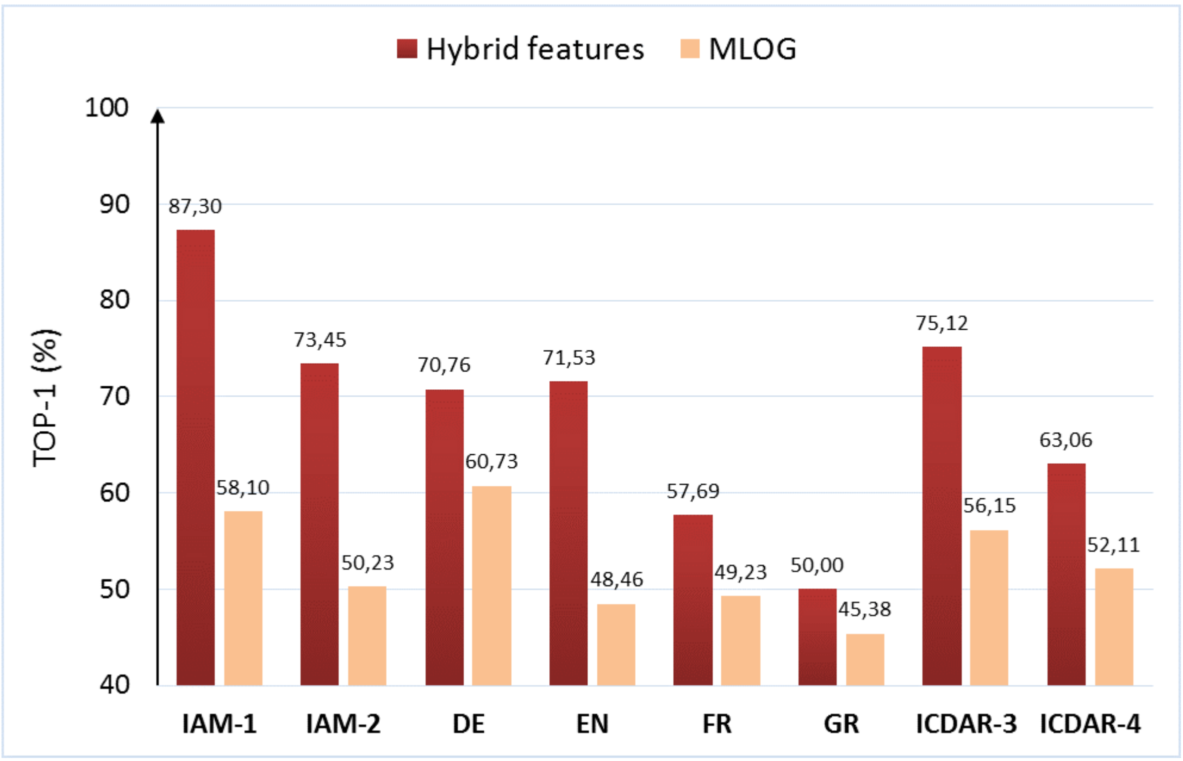


Figure 4.10: Hybrid features vs MLOG performances for writer-independent approach

Performances achieved thanks to hybrid features fusion are very satisfying for both writer-dependent and writer-independent approaches. The best achieved accuracy for writer-dependent approach is **94.72%** for IAM and **85.73%** for ICDAR, whereas the best TOP-1 result for writer-independent approach is **87.3%** for IAM and **75.12%** for ICDAR. The proposed framework outperfoms the results obtained by He et al., [6], for IAM dataset, where authors used 657 writers, reaching a TOP-1 of 69.5% and a TOP-5 of 86.1% using word recognition as an auxiliary task. Unfortunately, no state of the art results are available for ICDAR dataset.

4.9 Conclusion

Throughout this chapter, the obtained results demonstrated the strength of the hybrid features fusion proposed between CNN and MLOG features. The new MLOG descriptor has proven its efficiency and presented promising results, as it stays open to improvements. On the other hand, CNN have demonstrated its efficiency when provided with a large dataset such as IAM-1. It is now clear that the more data is provided, the better CNN performs. Furthermore, results attest that the fusion of CNN features and MLOG features expresses better than any other feature extraction method, the implicit information of authorship contained in a single handwritten word.

Conclusion

Writer identification represents an important biometric recognition application employed in the process of identification of individuals. This task remains an active research area, where researchers have approached the problem by considering a text document or text line. However, identifying the writer from a single word, has received very little interest, due to the lack of information contained in single handwritten words. In this work, we ventured to propose a robust system that identify the writer from single handwritten words. Two approaches of writer identification are addressed: writer-dependent and writer-independent. In the writer-dependent approach, the system can only identify samples of writers involved in the training phase, which leads to a classification problem. While the writer-independent approach, provided the existence of reference images for each writer, can identify even writers who were not involved in the training of the system. Consequently, in a first step, we addressed the problematic through a conventional pattern recognition system, composed of two main blocks, which are, feature extraction and classification. Feature extraction is perform by proposing a new descriptor, namely, Multiscale Local Oriented Gradient (MLOG), which was able to characterize the handwriting style contained in a word comparatively to pixel distribution and Gradient Local Binary Pattern descriptors. The classification step was carried out using the well-known Support Vector Machine (SVM). In a second step, the contribution of Convolutional Neural Network (CNN), especially, pre-trained VGG-16 architecture, was investigated as an end-to-end system and as a feature extractor, associated with SVM classifier, to identify the writer from single handwritten words. In the last step, features from MLOG and VGG-16 were fused, yielding a robust hybrid feature. Experiments were conducted on IAM and ICDAR2011 datasets. From the obtained results, we can draw the following conclusions:

– MLOG of order 5 seems to be the most suitable to perform writer identification. In addition, outcomes testify of MLOG efficiency and superiority compared to pixel distribution and GLBP descriptors, as it achieved very promising results and reached an accuracy of 76.94% and 72.27% for IAM and ICDAR datasets, respectively, on writer-dependent approach, while it achieved a TOP-1 of 58.1% and 60.76% for IAM and ICDAR datasets, respectively, on writer-independent approach.

– The distance used in the dissimilarities calculation impact the performance of writer-independent system. Canberra distance seems more adequate for IAM dataset, while Chi-square distance is more appropriate for ICDAR2011 dataset, however, roughly, both distances behave similarly.

– The number of writers used in training stage does not influence the performance accuracy using MLOG descriptor.

– The number of writers used in training stage does not influence the performance accuracy using MLOG descriptor.

– The performance of writer-independent approach depends strongly on the number of references per writer. Indeed, the more references, the more the performance accuracy increases.

– CNN-based system showed that, VGG-16 model encountered difficulties to identify the writer, mostly, when very few data are available. The results proved to be good on relatively large datasets, but rather poor on those with little amount of data. However, when substituting the last FC layer by SVM classifier, performances increased allowing a gain that reached 21.15%, yielding a best performance of 73.60% for IAM and 60.57% for ICDAR2011 on writer-dependent approach.

– Findings showed that Greek language presents the lowest performances, this can be explained by the fact that Greek alphabet is different and unique. In fact, language independent experiments carried out by ICDAR-3 and ICDAR-4 showed that, the proposed system operates independently of the language. ICDAR-3 contains English, French and German, which share the same Latin alphabet with some specific characters of each, provide a better accuracy than ICDAR-4 that contains Greek alphabet.

– The proposed hybrid features, gave very satisfying results as the system reached a maximum accuracy of 94.7% for IAM and 85.73% for ICDAR, on the writer-dependent approach, and a maximum TOP-1 accuracy of 87.3% for IAM and 75.12% for ICDAR on the writer-independent approach.

It is also important to note that training a CNN is far from a trivial task. Indeed, several difficulties arose throughout this work, namely:

⋆ Tuning parameters to adapt the used architecture to our datasets requires a large number of tests, while the training time could take several hours. For this, we were forced to limit the number of data and work with a limited number of writers and words per writer. Besides, it also forced us to use Transfer Learning to achieve better parameters optimization, instead of training a CNN from scratch.

⋆ The unavailability of powerful machines equipped with GPU, to carry out experiments was one of the major difficulties, luckily we had access to Google Colaboratory.

Finally, the results obtained in this work are very promising and encouraging. As possible research perspectives for this work, we suggest to train CNN on the 5^{th} order derivative, to capture deeper features that characterize handwriting style and therefore improving performances. Also, we recommend to investigate the proposed system on other datasets, especially, an Arabic dataset such as KHATT.

Bibliography

[1] Ayixiamu Litifu et al. “Writer Identification Based on Combination of Bag of Words Model and Multiple Classifiers”. In: *Pattern Recognition.* Springer, Mar. 2020, pp. 47–57. ISBN: 978-981-15-3650-2. DOI: 10.1007/978-981-15-3651-9_6.

[2] D. Arabadjis et al. “New mathematical and algorithmic schemes for pattern classification with application to the identification of writers of important ancient documents”. In: *Pattern Recognition* 46.8 (Aug. 2013), pp. 2278–2296. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2013.01.019. URL: http://dx.doi.org/10.1016/j.patcog.2013.01.019.

[3] Sreeraj.M and Sumam Mary Idicula. “Article: A Survey on Writer Identification Schemes”. In: *International Journal of Computer Applications* 26.2 (July 2011). Full text available, pp. 23–33.

[4] Abderrazak Chahi et al. “Local gradient full-scale transform patterns based off-line text-independent writer identification”. In: *Applied Soft Computing* 92 (Apr. 2020), p. 106277. DOI: 10.1016/j.asoc.2020.106277.

[5] Lorette G. Plamondon R. “Automatic Signature Verification and Writer Identification The State of the Art”. In: *Pattern Recognition* 22.2 (1989), pp. 107–131.

[6] Sheng He and Lambert Schomaker. “Deep adaptive learning for writer identification based on single handwritten word images”. In: *Pattern Recognition* 88 (Apr. 2019), pp. 64–74. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2018.11.003. URL: http://dx.doi.org/10.1016/j.patcog.2018.11.003.

[7] F. Monrose L. Ballard D. Lopresti. “Evaluating the security of handwriting biometrics”. In: *10th International Workshop on Frontiers in Handwriting Recognition* (2006).

[8] P.J. Flynn A.K. Jain A. Ross. *Handbook of Biometrics.* Science Business Media, 2007.

[9] L. chomaker. “Advances in Writer Identification and Verification”. In: *9th International Conference on Document Analysis and Recognition (ICDAR‘07)* 2 (2007), pp. 1268–1273.

[10] H. Arora S.N. Srihari S.-H. Cha and S. Lee. “Individuality of handwriting”. In: *Journal of Forensic Sciences. 47: (4)856-872* (2002).

[11] Puneet and Naresh Garg. “Binarization Techniques used for Grey Scale Images”. In: *International Journal of Computer Applications* 71 (June 2013), pp. 8–11. DOI: 10.5120/12320-8533.

[12] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66.

[13] S.Feffer. “It’s all about the features”. In: *Reality AI Blog* (2017). URL: https://reality.ai/it-is-all-about-the-features/ (visited on 06/04/2020).

[14] E.Justino D.Bertolini L.S.Oliveira and R.Sabourin. “Texture-based descriptors for writer identification and verification”. In: *Expert Systems with Applications* (2012).

[15] Anguelos Nicolaou et al. “Sparse radial sampling LBP for writer identification”. In: *13th International Conference on Document Analysis and Recognition (ICDAR).* Aug. 2015, pp. 716–720. DOI: 10.1109/ICDAR.2015.7333855.

[16] He Sheng and Lambert Schomaker. “Writer identification using curvature-free features”. In: *Pattern Recognition* 63 (Sept. 2016). DOI: 10.1016/j.patcog.2016.09.044.

[17] F. Shahabi and M. Rahmati. “Comparison of Gabor-Based Features for Writer Identification of Farsi/Arabic Handwriting”. In: *Tenth International Workshop on Frontiers in Handwriting Recognition.* Ed. by Guy Lorette. http://www.suvisoft.com. Université de Rennes 1. La Baule (France): Suvisoft, Oct. 2006. URL: https://hal.inria.fr/inria-00104466.

[18] A. Gattal et al. “Writer Identification on Historical Documents using Oriented Basic Image Features”. In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR).* Aug. 2018, pp. 369–373. DOI: 10.1109/ICFHR-2018.2018.00071.

[19] He Sheng and Lambert Schomaker. “Delta-n Hinge: Rotation-Invariant Features for Writer Identification”. In: *22nd International Conference on Pattern Recognition.* Sept. 2014. DOI: 10.1109/ICPR.2014.353.

[20] He Sheng and Lambert Schomaker. “Co-occurrence Features for Writer Identification”. In: *15th International Conference on Frontiers in Handwriting Recognition (ICFHR).* Oct. 2016, pp. 78–83. DOI: 10.1109/ICFHR.2016.0027.

[21] Theodore Bluche, Hermann Ney, and Christopher Kermorvant. “Feature Extraction with Convolutional Neural Networks for Handwritten Word Recognition”. In: *12th International Conference on Document Analysis and Recognition.* Aug. 2013, pp. 285–289. DOI: 10.1109/ICDAR.2013.64.

[22] Golnaz Ghiasi and Reza Safabakhsh. “Offline Text-Independent Writer Identification Using Codebook and Efficient Code Extraction Methods”. In: *Image Vision Comput.* 31.5 (May 2013), pp. 379–391. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2013.03.002. URL: https://doi.org/10.1016/j.imavis.2013.03.002.

[23] Ameur Bensefia, Thierry Paquet, and L. HEUTTE. “Grapheme Based Writer Verification”. In: *11th Conference of the International Graphonomics Society, IGS* (Nov. 2003).

[24] Alia Karim Abdul Hassan. “Writer Identification Based on Arabic Handwriting Recognition by using Speed Up Robust Feature and K- Nearest Neighbor Classification”. In: *JOURNAL OF UNIVERSITY OF BABYLON for Pure and Applied Sciences* 27.1 (2019), pp. 1–10. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2013.03.002. URL: https://doi.org/10.1016/j.imavis.2013.03.002.

[25] Andreas Schlapbach and Horst Bunke. “Off-line Writer Identification and Verification Using Gaussian Mixture Models”. In: *Machine Learning in Document Analysis and Recognition.* Ed. by Simone Marinai and Hiromichi Fujisawa. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 409–428. ISBN: 978-3-540-76280-5. DOI: 10.1007/978-3-540-76280-5_16. URL: https://doi.org/10.1007/978-3-540-76280-5_16.

[26] Zhenyu He, Xinge You, and Yuan Yan Tang. “Writer identification of Chinese handwriting documents using hidden Markov tree model”. In: *Pattern Recognition* 41.4 (2008), pp. 1295–1307. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2007.08.017. URL: http://www.sciencedirect.com/science/article/pii/S0031320307004037.

[27] Yousri Kessentini, Sana BenAbderrahim, and Chawki Djeddi. “Evidential combination of SVM classifiers for writer recognition”. In: *Neurocomputing* 313 (2018), pp. 1–13. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.05.096. URL: http://www.sciencedirect.com/science/article/pii/S0925231218306817.

[28] Sung-Hyuk Cha and Sargur Srihari. “On measuring the distance between histograms”. In: *Pattern Recognition* 35 (June 2002), pp. 1355–1370. DOI: 10.1016/S0031-3203(01)00118-2.

[29] C. I. Tomai, Bin Zhang, and S. N. Srihari. “Discriminatory power of handwritten words for writer recognition”. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 2. 2004, 638–641 Vol.2.

[30] Paquet T Bensefia A. “Writer verification based on a single handwriting word samples”. In: *EURASIP Journal on Image and Video Processing* (Dec. 2016). DOI: 10.1186/s13640-016-0139-0.

[31] D. Bertolini et al. “Reducing Forgeries in Writer-Independent off-Line Signature Verification through Ensemble of Classifiers”. In: *Pattern Recogn.* 43.1 (Jan. 2010), pp. 387–396. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2009.05.009. URL: https://doi.org/10.1016/j.patcog.2009.05.009.

[32] Nesrine Bouadjenek, Hassiba Nemmour, and Youcef Chibani. “Robust Soft-Biometrics Prediction from off-Line Handwriting Analysis”. In: *Appl. Soft Comput.* 46.C (Sept. 2016), pp. 980–990. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2015.10.021. URL: https://doi.org/10.1016/j.asoc.2015.10.021.

[33] Yu.W Jiang.N Xu.J and Goto.S. “Gradient local binary patterns for human detection”. In: *International Symposium on Circuits and Systems (ISCAS), Beijing* (2013), pp. 978–981.

[34] Bouibed Mohamed Lamine, Hassiba Nemmour, and Youcef Chibani. “Multiple writer retrieval systems based on language independent dissimilarity learning”. In: *Expert Systems with Applications* 143 (Oct. 2019), p. 113023. DOI: 10.1016/j.eswa.2019.113023.

[35] M. L. Bouined, H. Nemmour, and Y. Chibani. “New gradient descriptor for keyword spotting in handwritten documents”. In: *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. 2017, pp. 1–5.

[36] N.Bouadjenek. “Contribution to writer’s soft-biometrics prediction from handwriting analysis.” PhD thesis. Electronics: University of Science and Technology Houari Boumediene, 2017, p. 153.

[37] Soumendu Chakraborty and Pavan Chakraborty. “Local directional gradient pattern: a local descriptor for face recognition”. In: *Multimedia Tools and Applications* 76 (Jan. 2017), pp. 1201–1216. DOI: 10.1007/s11042-015-3111-6.

[38] Christopher J. C. Burges. “A Tutorial on Support Vector Machines for Pattern Recognition”. In: *Data Min. Knowl. Discov.* 2.2 (June 1998), pp. 121–167. ISSN: 1384-5810. DOI: 10.1023/A:1009715923555. URL: https://doi.org/10.1023/A:1009715923555.

[39] Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. ““One Against One” or “One Against All”: Which One is Better for Handwriting Recognition with SVMs?” In: *Tenth International Workshop on Frontiers in Handwriting Recognition*. Ed. by Guy Lorette. http://www.suvisoft.com. Université de Rennes 1. La Baule (France): Suvisoft, Oct. 2006. URL: https://hal.inria.fr/inria-00103955.

[40] A. U. Islam et al. “Hyperspectral Image Analysis for Writer Identification using Deep Learning”. In: *2019 Digital Image Computing: Techniques and Applications (DICTA)*. 2019, pp. 1–7.

[41] L. G. Helal et al. “Representation Learning and Dissimilarity for Writer Identification”. In: *2019 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2019, pp. 63–68.

[42] David E. Rumelhart et al. “Backpropagation: The Basic Theory. In Y. Chauvin D. E. Rumelhart (Eds.), Developments in connectionist theory”. In: *Backpropagation: Theory, Architectures, and Applications*. USA: L. Erlbaum Associates Inc., 1995, pp. 1–34.

[43] Y. Bengio Y. Lecun L. Bottou and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[44] T. Wiesel D. H. Hubel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *Physiol* (1962), pp. 160, 106–154.

[45] K. Fukushima. “Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics, 36* (1980), pp. 193–202.

[46] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Dec 15,2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (visited on 06/14/2020).

[47] R.Prabhu. *Understanding of convolutional neural network CNN deep learning*. March 4,2018. URL: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 (visited on 06/14/2020).

[48] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv 1409.1556* (Sept. 2014).

[49] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *Under Review of ICLR2016 (1997)* (Nov. 2015).