THE UNIVERSITY OF SOUTH ALABAMA

School of Computing

# Multi-Script Handwriting Identification by Fragmenting Strokes

BY

Joshua Jude Thomas

A Thesis Proposal

Submitted to the Graduate Faculty of the

University of South Alabama

in partial fulfillment of the

requirements for the degree of

Master of Science in Computer Science

in

Electrical and Computer Engineering

May 2024

Approved: Date:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Chair of Thesis Committee: Dr. Ryan G Benton

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Committee Member: Dr. Tom Johnsten

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Committee Member: Dr. Mohamed Shaban

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Committee Member: Dr. James Van Haneghan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Director of Graduate Studies: Dr. Debra Chapman

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dean of the Graduate School: Dr. J. Harold Pardue

# Multi-Script Handwriting Identification by Fragmenting Strokes

A Thesis

Submitted to the Graduate Faculty of the

University of South Alabama

in partial fulfillment of the

requirements for the degree of

Master of Science

in

Computer Science

by

Joshua Jude Thomas

B.S., University of South Alabama, 2021

A.S., Coastal Alabama Community College, 2017

May 2024

# TABLE OF CONTENTS

Page

[LIST OF TABLES iv](#_Toc161382784)

[LIST OF FIGURES v](#_Toc161382785)

[ABSTRACT vii](#_Toc161382786)

[CHAPTER I 1](#_Toc161382787)

[1.1 Verification and Identification Model 1](#_Toc161382788)

[1.2 Multi-Script Writer Identification 3](#_Toc161382789)

[1.3 Convolutional Neural Networks 4](#_Toc161382790)

[1.4 Goal of Study 5](#_Toc161382791)

[CHAPTER II 6](#_Toc161382792)

[2.1 Handwriting Identification 6](#_Toc161382793)

[2.1.1 Single-Script Handwriting Identification 6](#_Toc161382794)

[2.1.2 Multi-Script Handwriting Identification 7](#_Toc161382795)

[2.2 Stroke Decomposition 9](#_Toc161382796)

[CHAPTER III 11](#_Toc161382797)

[3.1 Stroke Fragmentation 12](#_Toc161382798)

[3.1.1 Stroke Fragmentation Process 14](#_Toc161382799)

[3.2 Training and Evaluation 17](#_Toc161382800)

[3.2.1 Model and Method of Analysis 18](#_Toc161382801)

[CHAPTER IV 20](#_Toc161382802)

[4.1 Visual Examination of Stroke Fragments 22](#_Toc161382803)

[4.2 CERUG Evaluation Results 25](#_Toc161382804)

[4.2.1 Per Class Metrics 27](#_Toc161382805)

[CHAPTER VI 34](#_Toc161382806)

[Future Work 34](#_Toc161382807)

[APPENDIX A 36](#_Toc161382808)

[REFERENCES 41](#_Toc161382809)

# LIST OF TABLES

Table Page

1. Experimental results from experiments 1 (train CN test EN), 2 (train EN, test CN), and 3 (train EN+CN, test MIXED) 26
2. Worst and best performers in experiments 1 and 2 with respect to F1 score 29
3. Number of sub-image samples for the worst and best performers in experiments 1, and 2 30
4. Per-Class precisions and recalls from the evaluation phase of all three experiments

36

# LIST OF FIGURES

Table Page

1. The Handwriting Identification Model vs the Handwriting Verification Model 2
2. Writing Scripts of three different origins that look very different on the visual level

4

1. Visualization of the stroke fragmentation process 14
2. Visualization of class distributions in the CERUG-CN and CERUG-EN datasets vs, the total number of stroke fragments extracted from each 21
3. Random Sample of 400 sub-images extracted from CERUG-CN 23
4. Random Sample of 400 sub-images extracted from CERUG-EN 24
5. Box Plot of F1 Scores for Experiments 1, 2, and 3 28
6. Random Sample of 400 sub-images extracted from Writer6464 32
7. Random Sample of 400 sub-images extracted from Writer1616 32
8. Random Sample of 400 sub-images extracted from Writer9101 33

**LIST OF ABBREVIATIONS**

KD Known Document

QD Questioned Document

CNN Convolutional Neural Network

HIM Handwriting Identification Model

HVM Handwriting Verification Model

CERUG Chinese-English database of the University of Groningen

OCR Optical Character Recognition

MSWI Multi-Script Writer Identification

ICFHR International Conference on Frontiers in Handwriting Recognition

# ABSTRACT

Joshua Jude Thomas, M. S., University of South Alabama, May 2024. Multi-Script Handwriting Identification by Fragmenting Strokes. Chair of Committee: Ryan G. Benton, Ph.D.

This study tests the effectiveness of Multi-Script Handwriting Identification after simplifying character strokes. Character simplification is performed through fragmentation of the character by branching-points and end-points, called stroke fragmentation in this study. This process shares similarities with the concept of stroke decomposition in Optical Character Recognition which attempts to recognize characters through the writing strokes that make them up. The main idea of this study is that the characters of different writing scripts (English, Arabic, Mandarin, etc.) may have common shapes which can be extracted and used in the handwriting identification process. The effectiveness of the processes described in this study is tested on the Chinese-English Database from the University of Groningen. While not achieving state of the art performance, the results of this study imply that simplifying characters shows promise in use for handwriting identification.

# CHAPTER I

**Introduction**

**Handwriting Identification** is the process of classifying the writer of a Handwritten Questioned Document based on the handwriting habits contained in that document. According Harralson and Miller in *Huber and Headrick’s Handwriting Identification: Facts and Fundamentals*, forensics experts commonly compare twenty one “discriminating elements of handwriting” that deal with properties such as word size, word placement, margin sizes, abbreviation choices, etc. [1].

Computationally assisted Handwriting Identification is performed by extracting visual features from a set of Known Documents (KD) and Questioned Documents (QD). In general, the features of a QD are compared to a database of writers having a set of KD to which the QD is compared against. The goal is to attribute the QD to one of the known writers by detecting similar features; the accuracy of this attribution, generally, increases as the set of KD increases [2].

## 1.1 Verification and Identification Model

Srihari et al. describe two main frameworks for performing handwriting identification, the **Handwriting Verification Model** (HVM) and the **Handwriting Identification Model** (HIM), both of which are shown in figure 2 [2]. Much of the research literature falls under one of the two categories. The goal of **HVM** is to determine whether the document was written by the same person or not (2-class classification). The extracted feature vectors of two documents are compared by a model to produce either a direct classification or a similarity score representing the likelihood that the two documents were written by the same person. The **HIM**, compares the features of a QD to a model of known writers to determine the writer directly (assuming the writer of the QD is in the set of known writers) [2]. Like the HVM, the HIM can either produce a direct classification or a vector of probabilities that the document belongs of any of the writers. The HIM is typically the more popular of the two frameworks and is the framework used in this study.



Figure 1. The Handwriting Identification Model vs the Handwriting Verification Model. As presented in [2]

## 1.2 Multi-Script Writer Identification

In **Multi-Script Handwriting Identification** (MSWI), writers are not limited to one language or writing script. A writer can produce handwritten documents in multiple languages (English, Chinese, etc.), write in different languages that share a writing script, or even write in the same language but use different writing-scripts across the set of documents. The goal of MSWI, according to the 2018 ICFHR competition paper on the subject, is to find “… writing patterns that are common across different scripts [and] may be exploited to identify the writer” [3]. This problem is based on the assumption that there are ingrained patterns in a person’s handwriting that are stable across different writing scripts, and that these patterns could be extracted as feature vectors for writing identification [3]. The competition paper specifically tries to do MSWI such that the model used to identify the writer is trained on one writing script, and then evaluated on the other. A model that performs well on this specific task would heavily be implied to have detected writing patterns unique to the writer that are present in both the training and evaluation (testing) dataset.

MSWI requires common features that can be extracted from the multiple different handwriting scripts being compared. However, the visual features of the different writing scripts can vary drastically from each other. Figure 2 shows the differences between four different sample scripts. On one side, you have Tamil and Arabic which typically have a flowy, nearly cursive style. On the opposite end you have Hanzi (Chinese), which has a more printed style. The English writing script can vary between having a cursive and print style, and even the visual difference between cursive and printed English could influence the Handwriting Identification process.

A picture containing text, document

Description automatically generated

Figure 2. Writing Scripts of three different origins that look very different on the visual level. (From left to right) Chinese Hanzi, Bengali, Tamil, and English.

## 1.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are deep neural networks designed to perform spatial information such as images [4]. They are very good architectures for classification tasks on images and have been used in handwriting identification research as feature extractors [3], [5]. However, one problem of CNN, often shared by all deep learning models, is the sheer amount of data required to train them properly. A CNN trained to extract features for handwriting identification would have to have a large amount of, potentially labeled, data to train on. Even then, the CNN model may be trained on many data on a writer class in one writing script but lack data for that same writer in another writing script. In addition, the visual features of the different writing scripts may, themselves, have an impact on model performance.

## 1.4 Goal of Study

This study attempts to perform Multi-Script Handwriting Identification (MSWI) by breaking down the characters of different writing scripts into simpler shapes, namely strokes. The concept is that similar looing shapes in the sub-parts of a handwritten character can be used to compare handwritten documents across different writing scripts. This study uses a CNN model as its HIM to directly classify the writer of a given multi-script dataset. The novelty presented in this study is the method in which the characters are passed to the CNN, which by performing Stroke Fragmentation on the characters of a document yields multiple, simpler shapes from each.

There are several important assumptions in this study. Each document is assumed to have only one writer per document. A document may contain multiple writing scripts, but there is a many-to-one relationship between the documents in a dataset and the writer classes of the dataset. Another assumption is that the writing medium and utensil is the same, or similar enough, between each of the datasets. This study also assumes that the habits of a writer do not vary across a document due to influences such as time [1].

# CHAPTER II

**Literature Review**

## 2.1 Handwriting Identification

In general, the methodologies presented in the Handwriting Identification literature differ mainly by the features they extract from a handwritten document. Each of the studies presented in this review either define new features to extract from a handwritten document or utilize known feature extraction techniques in a novel way to perform writer identification. “Individuality of Handwriting” by Srihari et al. is an old and well-regarded paper that tests the hypothesis that “handwriting is individual” [2]. The hypothesis is tested by implementing features based on the twenty-one features by Huber and Headrick, plus a set of “computational features” consisting of a set of eleven “macro-features”, and “micro-features”. The features are tested in both the HVM and HIM, described in the study.

### 2.1.1 Single-Script Handwriting Identification

Foroozandeh et al. used a deep transfer-learning approach to perform signature verification [5]. Several popular CNN architectures were used as feature extractors which were then used to classify a genuine signature versus a forgery. Nguyen et al. uses a CNN to extract the local features of randomly sampled sub-images of a document [6]. The local features were aggregated via a pooling operation and then used to classify the writer. Shaikh et al. use a “Hybrid Deep Learning” approach to perform writer verification [7]. They pair one of two “Auto-Learned” features, a CNN and an Auto Encoder, and one of two “Human Engineered” features, SIFT descriptors and GSC descriptors. The four resulting combinations were trained to classify the writers of pairs of “AND” images. Wu et al. extracted SIFT key points from segmented word regions to generate a codebook based classifier [8]. Jain, Rajiv and Doermann, David approximate the contours of handwritten characters into “k-adjacent” segments [9]. The contours are approximated into lines by a line-fitting algorithm and then sets of 2-4 line segments are taken and described through a feature vector. Tan et al. extract features from a bounding box and a bounding quadrilateral for writer identification [10]. Pervouchine et al. extract handwriting strokes via modeling with cubic splines [11]. Strokes are recreated via curves by vectorizing the input image, merging choice skeletal branches, and recreating the loops of a handwriting stroke, caused by self-overlapping strokes. The recreated strokes are not directly passed to the HIM but are summarized via a feature vector.

### 2.1.2 Multi-Script Handwriting Identification

The International Conference on Frontiers in Handwriting Recognition (ICFHR) 2018 Multi-Script Handwriting Identification competition reports on the successes of four different systems submitted to the competition [3]. These systems are the LIMPAF-I, LIMPAF-II, Tokyo System, and the Nuremberg System. The LIMPAF-I and LIMPAF-II were submitted by the same group; LIMPAF-I uses Uniform Complete Local Binary Patterns (U-LBP) [12] for its feature extraction while the LIMPAF-II uses Oriented Basic Image Features (oBIF) [13]. The Tokyo system used two CNNs to extract features from randomly selected sub-images of a writing sample. Features extracted from writing samples were passed into a “Transfer Neural Net” to transform the extracted features, of writing samples of possibly different writing scripts into a more uniform representation, which were then used in classification. The Nuremberg system was actually based on another paper ([14]) which extracts features by a pre-trained CNN. The extracted features were then “PCA-Whitened” and encoded in a visual bag of words algorithm called VLAD [15]. Abbas *et al* combines LBP and oBIF to create a histogram of both over the whole range of the document [16]. He et al. tested the power of handwriting junctions on the writer identification task [17]. Junctions are grouped into L-junctions representing points where writing strokes are sufficiently curved, and T, Y and X-junctions where two handwriting strokes intersect. A junction feature is defined containing the center-point, scale (defined as minimum branch-length, where a branch is one part of an intersecting stroke), two to four angles representing the directions the branches of a junction point in, and the “strength of a branch” in a set number of directions. This junction feature is used to form a “junctlet” codebook of common junctions. Finally, the codebook is used to create a histogram containing the number of times a type of junction was detected in a document, which is then used for classification. This study also introduces the Chinese-English database of the university of Groningen (CERUG) consisting of a collection of Documents written in Chinese, English, or a combination of both[[1]](#footnote-2). Semma et al. use CNN features encoded into modified vectors ([15]) for handwriting identification [18]. Sub-images of a handwritten document are taken around key-points found via the Harris corner detector. And then CNN features are extracted, processed into fixed length VLAD vectors, and classified. Ahmed et al. focus on the “ending strokes”, parts of a character appearing at the tail-end, for writer identification by assembling their contours into a code-book [19].

## 2.2 Stroke Decomposition

**Stroke Decomposition** is a method of reducing a handwritten character into approximate individual strokes. While similar concepts have been applied in Handwriting Identification [11], This technique typically appears in research on Optical Character Recognition (OCR) which tries to convert a handwritten text into a typed, digital format rather than identify the writer of the text. The stroke fragmentation process defined in this study shares many similarities to stroke decomposition, mainly through the skeletonization (thinning used in the stroke decomposition literature) of a handwritten character, finding branching points in that character, and region growing the pixels of the original character into the segmented strokes.

Kim et al. decompose Chinese characters into individual strokes by first performing a morphological thinning to reduce each character to a single pixel width [20]. They then segment the characters based on branching points (areas where strokes overlap) and excessively curved segments, similar to the critical points in [17]. The segments are grown morphologically using two modifications on a morphological dilation which use vectors both parallel and perpendicular (elongation and fattening) to the direction of each segment. A more standard dilation, named isotropic expansion, is performed on segments that are not long enough for the elongation step (less than five pixels in the paper). Both Fattening and Isotropic expansion are constrained by an approximate convexity measure. Finally, Grown stroke segments that have intersecting parts are then potentially merged using the same convexity measure as a conditional [20]. Chen et al. convert handwritten Chinese characters into stroke sequences (strings of numbers that indicate the type of stroke by number and the order the stroke was written by the position of that number) by using an encoder-decoder architecture to convert the character images to stroke sequences[21]. Liu et al. use a model based approach to stroke decomposition [22]. They represent the model of a character through an attributed-relationship graph and generate said graphs through thinning, forming control points, then approximating lines where possible. Kim et al. do not try to form the strokes of a character directly. Instead, they use a handwritten character model composed of (statistical) random variables on the distribution of pixel positions [23]. To more efficiently compare the pixels of an input character to the character model, the pixels are grouped into approximate stroke regions by applying a special thinning method to segment the strokes of an image into “sub-strokes”, and then using a nearest neighbors’ scheme to group the pixels of the original image on those strokes.

# CHAPTER III

**Methodology of Study**

The stroke fragmentation process is performed as a separate process from Writer Identification with the purpose of generating data (filtered sub-images) for the Handwriting Identification Model (HIM). It takes the characters of a handwritten document and produces one or more fragments of a stroke, called **stroke fragments** in this study, and each are stored in one or more sub-images, filtered to contain only that stroke fragment. A typical Handwriting Dataset consists of a set of digitized handwritten document images (scanned or photographed). The stroke fragmentation process converts these document images into multiple, filtered sub-images, containing the stroke fragments, and groups them into different datasets based on the writing script contained in the document.

To extract the stroke fragments, a set of filters is constructed for each. These filters are constructed from the skeleton and then region grown through sklearn’s K-Nearest-Neighbors (KNN) algorithm. Sub-images containing the simplified stroke shape are extracted from the document with the bounding box coordinates of the filters, and then unneeded pixel values (not corresponding to the filter pixels) are removed. Finally, the filtered sub-images are saved to disk, and then later used to fit the HIM (the resnet-50 Convolutional Neural Network (CNN) is used as the HIM). The final evaluation metrics will be measured on a different dataset, containing a different writing script, to judge the performance of the model on a writer it has data on, but with no knowledge of the writing script used for evaluation. The evaluation metrics used in this study are: Top-1 Accuracy (Categorical Accuracy), Top-10 Accuracy, Precision, and Recall.

## 3.1 Stroke Fragmentation

The main idea of the stroke fragmentation process is to split handwritten characters into simpler shapes, which are hopefully more common between different writing scripts. The handwritten characters of a document are split up using **critical points** consisting of the branching points, and end points of the character. The end goal of the fragmentation process is a collection of simplified shapes extracted from each document that could then be used with more conventional handwriting identification methods.

A mathematical graph representation is used to facilitate the finding of the critical points in a document. In this setting, all the characters contained in a handwritten document are represented as a disconnected, planar, multi-graph: , where is a set containing all the critical points of the document and is a multi-set of pairs . Each pair in represents that the critical points are connected by a part of a stroke (A.K.A. a stroke fragment). The simplified shapes we want to extract can then be thought of as subgraphs of the connected portions of the larger document graph. Note that the actual pixel coordinates corresponding to the handwritten character in a document are not represented as the edges or nodes but are instead included as attributes in the nodes and edges of the graph. The general process of stroke fragmentation is shown below.

Stroke Fragmentation:

1. Preprocessing: transform the document image into an inverted grayscale image for feature extraction, and an inverted binary image for forming the filters used in the extraction step.
2. Reduce the characters in the binary image to single pixel width through morphological skeletonization.
3. Convert the morphological skeleton into a graph representation to locate the critical points.
4. Apply an integer label to each edge and take the attributed pixel coordinates of the edge and it’s connecting nodes to form a skeleton of the filter
5. Perform region growing between the skeleton of the filter and the binary image to form different filters.
6. Remove any filter whose bounding box is too small or two large in either two dimensions (filtering criterion)
7. Use the filter(s) and its bounding box to extract the corresponding sub-image from the grayscale image, then remove any pixel values for pixels not coinciding with the pixels in the filter

The result of the Stroke Decomposition process is filtered sub-images, where is the number of fragmented strokes for a document and is the number of stroke fragments rejected by the filter criterion. The filtered sub-images are grouped into a dataset representing a particular writing-script, and then further into their writer class. Figure 3 shows a visualization of the process without the filtering criterion.

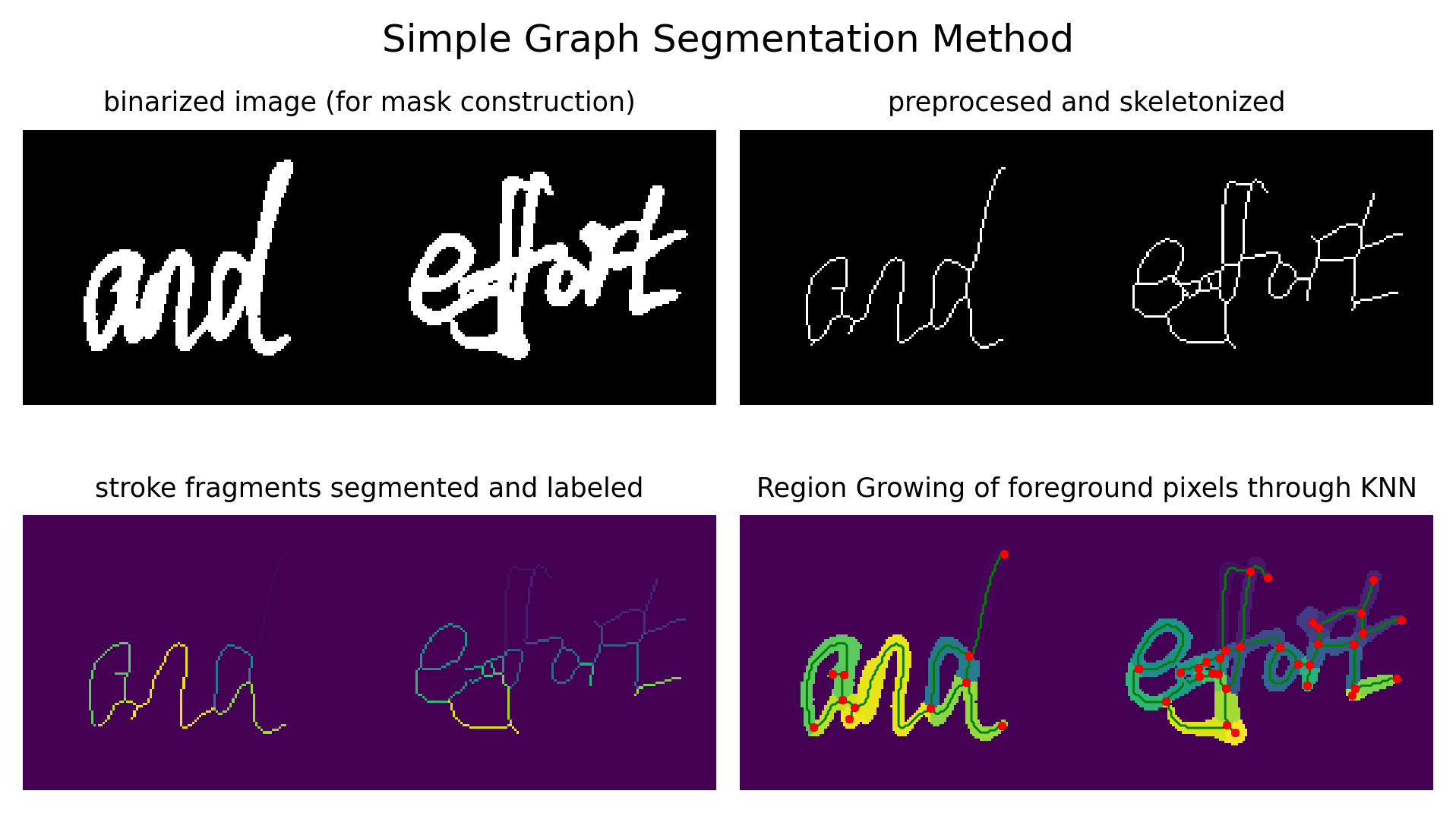


Figure 3. Visualization of the stroke fragmentation process. Starting with a raw binary image to construct the filters. It is skeletonized, segmented on the branching points, and the regions of the stroke fragments are grown. Note that the vertex and edge points are plotted over the region grown image for visualization[[2]](#footnote-3).

### 3.1.1 Stroke Fragmentation Process

**Step 1**

When performing stroke fragmentation on a document, it is converted into a grayscale representation, and a corresponding binary representation. The grayscale representation of the document is kept for the feature extraction stage at the end of the stroke fragmentation process. The binary image is obtained via Otsu’s method [24], which is then used to form the filters. When processing the binary image(s), an inherent amount of noise will exist in the images of the handwriting dataset, depending on both the method and quality of digitization, and may influence the graph representation. Possible examples are smudges on the paper, and visible paper edges from poor digitization, etc. So, a gaussian blur which is applied to the image document before binarization (though the blur is not applied to the grayscale image when extracting the stroke fragments).

**Step 2**

The documents, now blurred and binarized, go through morphological skeletonization. The skeletonization of the binary shapes in the image reduce them to a single pixel width and try to preserve the general shape of the character as much as possible.

**Step 3**

The skeletonized handwritten document is converted into a graph theory representation[[3]](#footnote-4) to facilitate finding the critical points of each character. The graph representing a handwritten document is defined as such that the vertices of the graph correspond to pixel considered critical points in the document. A critical point is either an endpoint of a handwriting stroke in the document (pixels connected to only one other pixel) or a branching point between two strokes in a document (pixels who are connected to three or more other pixels). Every pixel that does not fit the above criteria is considered an edge pixel; The edges of a graph represent critical points connected by a line of edge pixels. When the graph representation is made, the critical point pixels are stored in the corresponding vertices and the edge pixels are stored in the corresponding edge.

**Step 4 and 5**

During the construction of the filters, each the attributed edge-pixels of each edge is assigned an integer label . These labeled edge pixels represent the skeleton of a simplified stroke fragment. Afterwards, region growing is performed, via KNN, between the filter skeletons and the binarized image to form different filters. That is, the filter skeletons are used as the training data to fit the KNN model, which is then used to segment the pixels of the binary image into the final set of filters.

**Step 6**

Some artefacts may remain in the image even after the blur is applied. Large artefacts such as visible page edges will be viewed as handwriting strokes by the stroke decomposition algorithm will result in a noisier dataset. In addition to large artefacts, some filters may end up being too small to give any meaningful information and may result in a noisy dataset as well. A **filtering criterion** is defined to remove large artefacts and small filters from the final output. The bounding box of each filter is found and used for both filtering and the feature extraction step (step 7). The filtering criterion is defined such that if both width and height is smaller than 10 pixels, or if either the width or height is smaller than 3 pixels or larger than 75% the width/height of the image document, the filter is rejected and not used in feature extraction. This has the consequence that some foreground pixels of the original image are discarded and not used during the training process.

**Step 7**

Filters that are not rejected are used to extract the corresponding pixels in the inverted grayscale image. The bounding box coordinates are used to define the location of the sub-image containing the desired stroke pixels. After the sub-image boundary is obtained any pixel not coinciding with the pixels in the filter is zeroed out.

In summary, the stroke fragmentation process takes a scanned handwritten document of a given Dataset and transforms it from a single image into multiple, filtered sub-images containing stroke fragments that, if put back together, make up the original (except for the pixels corresponding to any rejected filter). The sub-images are saved to a directory and split into the different writing scripts of the document (e.g. Arabic, Chinese, English) such that each writing script can be treated as its own dataset. These datasets are then used as the training and evaluation (testing) sets on the CNN model.

## 3.2 Training and Evaluation

Besides the prior stroke fragmentation process, the training and evaluation of the Handwriting Identification Model (HIM) is a standard process. Writing script directories are loaded as datasets via Kera’s image\_dataset\_from\_directory() method, which loads a directory of sub-directories representing the writer classes. The class labels are inferred from the names of the subdirectories and are then one-hot encoded and used to predict the probabilities that the sub-image comes from a document written by any of the writers (through SoftMax activation). What writing scripts are used in the Training and Test splits will depend on the experiment being performed. Training sets will be further split into training and validation datasets, using an 80-20 split.

### 3.2.1 Model and Method of Analysis

A resnet-50 model is used as the CNN model of this study [25]. The SoftMax layer of resnet-50 is modified to predict the 105 writer classes of CERUG and the weights are randomly initialized. The entire model is fit to the dataset over 40 epochs, using the NADAM optimizer with the default parameters as set in Keras [26]. The Keras ModelCheckpoint callback is used to select the best performing model trained over the epochs. Note that this serves as a form of regularization for the model [4]. Validation loss is used to determine the best model.

For evaluation metrics: Top-1 (Categorical Accuracy), Top-10, Precision, and Recall are used. Top-10 is a Top-N metric and, as defined in the ICFHR 2018 Competition on Multi-Script Writer Identification, defined as “… the scenario where the genuine writer of a query document is present within the list of N most probable writers received by the system” [3]. The Top-N metric is a popular evaluation metric in the literature, and Top-10 is specifically chosen to compare with the results of this study with the junctlets feature presented by He et al. in ‘Junction Detection in Handwritten Documents and its Application to Writer Identification’ [17]. The precision and recall are metrics measuring the model’s ability to correctly predict a label and to capture all the labels of a given class, respectively.

Where:

1. Correct Predictions and Total Predictions are over the entire model.
2. TP (True Positive) is the sum of correct predictions of a class.
3. FP (False Positive) is the sum of instances incorrectly labeled as a class.
4. FN (False Negative) is the sum of instances incorrectly labeled as not being part of a class.

The metrics used are fine-grained, meaning that the sum of the true positive, false positive, and false negative instances of all the classes are summed before calculating the precision and recall.

# CHAPTER IV

**Experiments**

The Chinese-English Database of the University of Groningen (CERUG), defined in [17] contains different writing script per writer, for 105 writers. The writing scripts are partitioned into: CERUG-CN for Chinese writing in page 1 and 2, CERUG-EN in page 3 (split over two images), and CERUG-MIXED in page 4 which consists of a mix of Chinese and English. Three experiments are performed in this study (two of which follow the tasks described in [3]). Experiment 1 uses the CERUG-CN as the training-set, and CERUG-EN as a test set. Experiment 2 uses CERUG-EN as the training set and CERUG-CN as the test set. Experiment 3 merges CERUG-CN and CERUG-EN together for use as the training-set and uses CERUG-MIXED as the test-set. Note that experiment 3 does not fit the goal of training on one writing set and evaluating on another and is used as more of a baseline model.

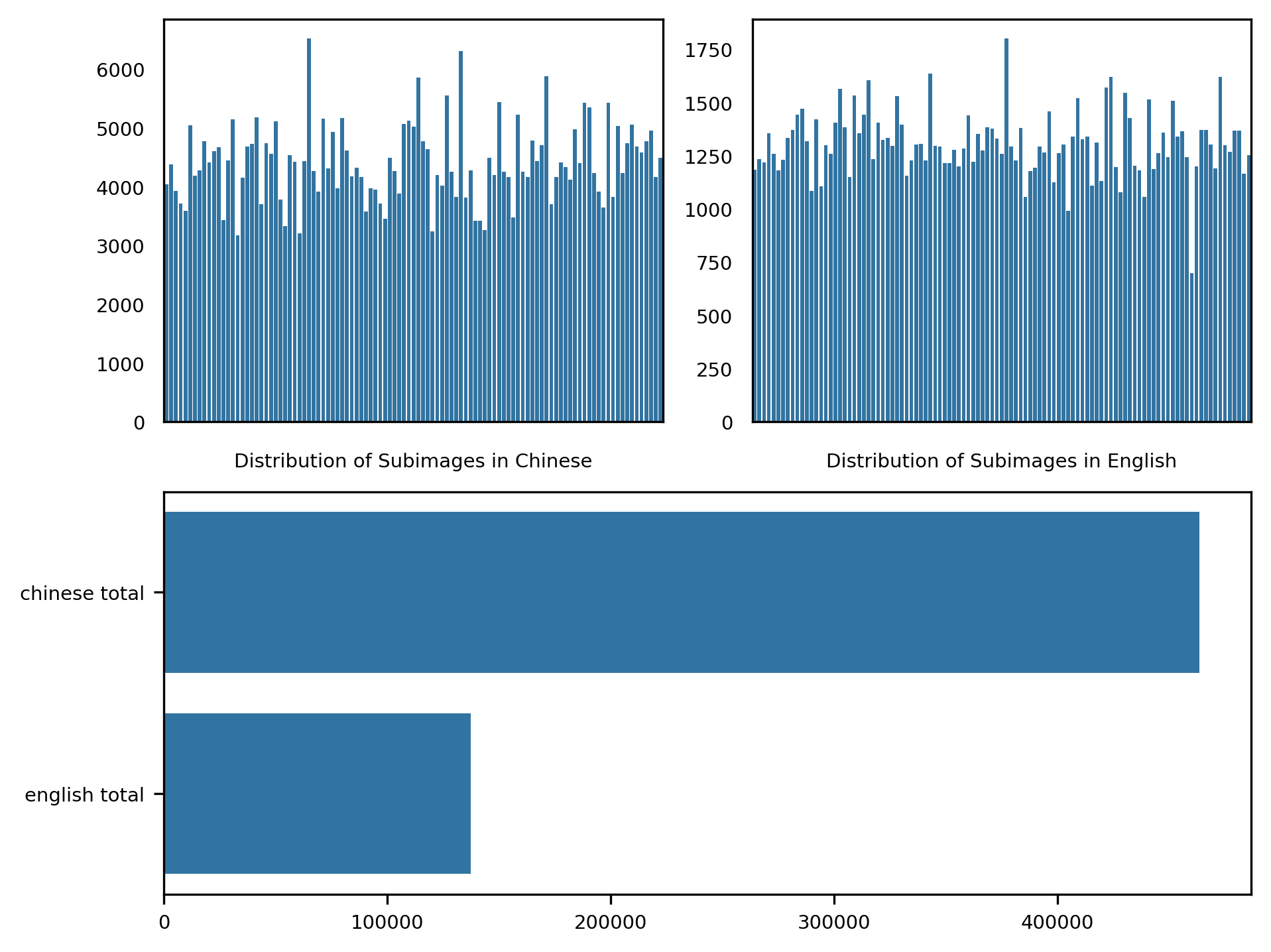


Figure 4. Visualization of class distributions in the CERUG-CN and CERUG-EN datasets vs, the total number of stroke fragments extracted from each. The number of sub-image samples in CERUG-CN greatly outnumbers the number of sub-image samples in CERUG-EN.

Figure 4. shows the class distribution CERUG-CN and CERUG-EN vs. the distribution of stroke fragments extracted from each. It is of interest to note that, while the individual classes seem relatively balanced, the different writing scripts themselves constitute a data imbalance. In Total: there were 463,507 sub-images extracted from CERUG-CN, 137,258 sub-images extracted from CERUG-EN, resulting in different drastically differing performances in all three experiments.

## 4.1 Visual Examination of Stroke Fragments

It may be useful to examine the images extracted during the stroke fragmentation process. The stroke fragments of the sub-images are essentially parts of a handwriting stroke in between the critical points of a character: the branching points between strokes and also where a stroke begins or ends. The pixels corresponding to the critical points aren’t necessarily in the stroke fragment, since only the edge pixels in the graph are used in the KNN region growing step. Thus, the stroke fragments generated will be line curves with no branching off parts, and little, if any at all, will be complete loops. Figures 5 and 6 show two different samples of CERUG-CN and CERUG EN, respectively. Each shows a sample of 400 different stroke fragments of varying scales.

A group of black and white symbols

Description automatically generated

Figure 5. Random Sample of 400 sub-images extracted from CERUG-CN.

A close up of a number

Description automatically generated

Figure 6. Random Sample of 400 sub-images extracted from CERUG-EN.

With the assumption that both stroke fragment samples are representative of the total dataset, we can draw some visual observations from the two figures. Three classes of stroke fragments are observed: blobs, curves, and loops. Blobs are squarish, small-scale parts of a stroke that were sandwiched between two critical points in close proximity. It is assumed that little useful information can be drawn from the blobs, and the filtering criterion (step 6 of stroke fragmentations) removes the especially small blobs. Curves are any sufficiently long stroke fragment that does not contain a loop. Curves seem to make up most of the stroke fragments extracted from CERUG. Finally, loops are any curves that completely wrap around and connect to themselves. For curves to form, the part of the handwriting character making up the loop would have to be connected to the character by only one branching point, or the branching points would have to be positioned such that the region growing step partitions all of them into the loops filter.

The following is based on casual examination of the two samples. CERUG-CN appears to have many more blobs than CERUG-EN while CERUG-EN seems to have more loops. The curves of CERUG-EN also seem to have more curvature than the curves of CERUG-CN. The increased number of blobs, at least in CERUG-CN could be attributed to tightly packed, crisscrossing handwriting strokes making up the characters. The loops and higher curvature of the curves in CERUG-EN seems to be due to the more cursive nature of English in CERUG-EN, with the handwriting strokes overall being more flowing that than of the handwriting strokes of Hanzi (Chinese) in CERUG-CN.

## 4.2 CERUG Evaluation Results

Table 1 presents the experimental results from all three experiments. The performance results of the junctlets feature, from, are also added for reference. Note that the two metrics are not exactly comparable because the junctlets feature compute a global feature (the occurrence histogram) over the entire document while this study tests CNN accuracy across the fragmented strokes of all documents of a particular script (with an additional split along the training dataset for verification). However the results from [17] are still a good baseline metric.

While not performing nearly as well as the junctlets feature proposed in [17], the evaluation performance does show that the method of simplifying stroke shapes has promise in use for writing identification. A completely random classifier would have an average top-1 (categorical accuracy) score of about 1% for the 105 writer classes of CERUG. The range of 21%-47% implies that there is some information of the writer carried in the stroke fragments produced in this experiment. Furthermore, the top-10 accuracy, having a range of 71-92%, implies that the writer producing a stroke is classified as being in the ten most likely writers, out of 105 writers, for at least 71% of all the strokes in the dataset.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment Number | Top-1 Accuracy | | Top-10 Accuracy | | Precision | Recall |
| Proposed | Junctlets | Proposed | Junctlets | Proposed | Proposed |
| Experiment 1 | 0.365 | 0.907 | 0.865 | 0.967 | 0.524 | 0.266 |
| Experiment  2 | 0.214 | n/a | 0.715 | n/a | 0.318 | 0.138 |
| Experiment  3 | 0.477 | n/a | 0.920 | n/a | 0.662 | 0.355 |

Table 1. Experimental results from experiments 1 (train CN test EN), 2 (train EN, test CN), and 3 (train EN+CN, test MIXED).

While not directly important to the study, it is of interest to note the overall precision and recall of the trained CNN model for each experiment. The recall performance is noticeably less than the precision performance, and the trend between precision and recall matches the training size between all three experiments (with experiment 3 having the highest). These scores seem to imply that while the model is not able to correctly classify many of the stroke fragments of a writer, the strokes it does classify as belonging to a particular writer have a, relatively, higher chance of actually being produced by that writer.

As expected with the imbalance of the writing scripts, Experiment 2 is the worst performer of all three experiments due to CERUG-EN being much smaller, at least in the number of extracted stroke fragments, than CERUG-CN. Experiment 3, naturally, has the highest performance since it includes both CERUG-CN and CERUG-EN as the training set.

### 4.2.1 Per Class Metrics

The precision and recall metrics presented in table 1 are the fine-grained metrics over all 105 writer classes. The per-class precision and recalls may also be used to draw some useful insights of the model.

A graph with multiple colored squares

Description automatically generated with medium confidence

Figure 7. Box Plot of F1 Scores for Experiments 1, 2, and 3.

Figure 7 shows the boxplots of the F1 score for experiments 1, 2, and 3. Experiment 3 uses both CERUG-CN and CERUG-CN as it’s training set so it is not surprising that it outperforms the models in both experiments 1 and 2. Experiment 3 is ignored for the remainder of this analysis as it does not experiment with the specific case of MSWI used in this study, to train on one writing script and evaluate on the other, and is more of a baseline.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Writer Class** | **Experiment 1** | | | **Experiment 2** | | | **Experiment 3** | | |
| precision | recall | F1 | precision | recall | F1 | precision | recall | F1 |
| Writer1616 | 0.023 | 0.008 | 0.006 | 0.06 | 0.019 | 0.014 | 0.186 | 0.218 | 0.1 |
| Writer9101 | 0.107 | 0.015 | 0.013 | 0.023 | 0.021 | 0.011 | 0.504 | 0.404 | 0.224 |
| Writer6464 | 0.833 | 0.95 | 0.444 | 0.651 | 0.673 | 0.331 | 0.886 | 0.82 | 0.426 |

Table 2. Worst and best performers in experiments 1 and 2 with respect to F1 score. Writer1616 performed the worst in experiment 1, Writer9191 performed the worst in experiment 2, and Writer6464 performed the best in all three experiments.

Table 2 presents the worst performers in experiments 1 and 2, and then the best performer in all three experiments. The full table of writers and their performances in all three experiments can be found in Appendix A, table A1. Writer1616 performed the worst in experiment 1, Writer9101 performed the worst in experiment 2, and Writer6464 performed the best in all three experiments. There does not appear to be a correlation between the number of samples (filtered sub-images) of the writer classes, per script, and the performance of that class in a particular experiment. Table 3 shows the number of samples in the writer classes of CERUG-CN and CERUG-EN, for all three writers. If the number of samples were to be the deciding factor, Writer9191 would outperform Writer6464 in experiment 1, and Writer1616 would outperform Writer6464 in experiment 2, and neither is the case.

|  |  |  |
| --- | --- | --- |
| **Writer** | **Samples in CERUG-CN** | **Samples in CERUG-EN** |
| Writer1616 | 3174 | 1299 |
| Writer9191 | 3916 | 1244 |
| Writer6464 | 3816 | 1126 |

Table 3. Number of sub-image samples for the worst and best performers in experiments 1, and 2.

The following is a visual analysis of the three writers (as performed in 4.1). Figures 8, 9, and 10 (below) show a random sample of stroke fragments from Writers 6464, 1616, and 9101, respectively. One immediate observation to make is that the stroke fragments of Writer6464 are much lighter than both the sample strokes of Writer1616 and Writer9101, as well as the sample strokes from CERUG-CN and CERUG-EN overall. It is highly likely that Writer6464 performed so well in all three experiments due to the lighter shade used to create the strokes. It is not as clear why it may be that Writer1616 and Writer9101 perform worse in experiments 1 and 2, respectively. One thing that stands out is that Writer1616 has consistently darker values representing their stroke fragments. The Stroke fragments presented in Figure 9 all have very dark gray level values with little variation in light. These two observations for Writer6464 and Writer1616 may indicate a grayscale level bias in the trained model for the three experiments. It is less clear why Writer9101 performs badly. Possible reasons may be that the particular writing style of Writer9101 may have a big impact on the stroke fragment generation process. The types of stroke fragments generated for Writer9101 may throw off the model for experiment 2.

A group of black and white letters

Description automatically generated with medium confidence

Figure 8. Random Sample of 400 sub-images extracted from Writer6464.

A group of black letters

Description automatically generated with medium confidence

Figure 9. Random Sample of 400 sub-images extracted from Writer1616

A group of black lines

Description automatically generated

Figure 10. Random Sample of 400 sub-images extracted from Writer9101

# CHAPTER VI

**Conclusion**

Multi-Script Handwriting Identification attempts to classify the writer of a handwritten document in a setting where there can be multiple writing scripts or languages in use, and with the possibility that a writer can create documents in more than one writing script. As such, Multi-Script Handwriting Analysis seeks common features between the different writing scripts that are both effective and consistent. This study tested the effectiveness of breaking down the characters of a document into simpler shapes and performed this by breaking down a character along its critical points. While not state-of-the-art, the results of the three experiments performed on the CERUG dataset show that the simplifying character shapes have a promising potential for being used in Handwriting Identification[[4]](#footnote-5).

## Future Work

In this study, the characters are broken down into simpler shapes by segmenting the fragments of a stroke along the “critical points” found in the document. Performance might be increased by adding an additional step that re-merges these stroke fragments into larger, but still simpler, shapes than the whole character, shapes. It may also be interesting to try and train a model directly on the curvature of the skeletons produced during the stroke fragmentation process. The contours of a skeleton, produced during the stroke fragmentation process, may be used to classify the writer by converting those skeleton edges into a chain-code, or similar, representation.

# 

# APPENDIX A

**Per-Class Precision and Recall Metrics**

Table A1. Per-Class precisions and recalls from the evaluation phase of all three experiments.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Writer Class** | **Experiment 1** | | | **Experiment 2** | | | **Experiment 3** | | |
| precision | recall | F1 | precision | recall | F1 | precision | recall | F1 |
| Writer0101 | 0.626 | 0.739 | 0.339 | 0.537 | 0.493 | 0.257 | 0.765 | 0.811 | 0.394 |
| Writer0202 | 0.454 | 0.36 | 0.201 | 0.216 | 0.114 | 0.075 | 0.38 | 0.543 | 0.224 |
| Writer0303 | 0.254 | 0.286 | 0.135 | 0.15 | 0.073 | 0.049 | 0.407 | 0.464 | 0.217 |
| Writer0404 | 0.336 | 0.237 | 0.139 | 0.079 | 0.303 | 0.063 | 0.378 | 0.402 | 0.195 |
| Writer0505 | 0.303 | 0.13 | 0.091 | 0.123 | 0.079 | 0.048 | 0.433 | 0.358 | 0.196 |
| Writer0606 | 0.222 | 0.09 | 0.064 | 0.09 | 0.053 | 0.033 | 0.366 | 0.178 | 0.12 |
| Writer0707 | 0.188 | 0.147 | 0.083 | 0.07 | 0.065 | 0.034 | 0.374 | 0.228 | 0.142 |
| Writer0808 | 0.321 | 0.166 | 0.109 | 0.128 | 0.148 | 0.068 | 0.554 | 0.361 | 0.219 |
| Writer0909 | 0.26 | 0.293 | 0.138 | 0.109 | 0.173 | 0.067 | 0.396 | 0.344 | 0.184 |
| Writer1010 | 0.302 | 0.159 | 0.104 | 0.122 | 0.13 | 0.063 | 0.398 | 0.263 | 0.159 |
| Writer1111 | 0.705 | 0.793 | 0.373 | 0.293 | 0.498 | 0.185 | 0.765 | 0.751 | 0.379 |
| Writer1212 | 0.185 | 0.171 | 0.089 | 0.193 | 0.026 | 0.023 | 0.372 | 0.216 | 0.137 |
| Writer1313 | 0.159 | 0.082 | 0.054 | 0.066 | 0.022 | 0.016 | 0.334 | 0.27 | 0.149 |
| Writer1414 | 0.322 | 0.237 | 0.136 | 0.111 | 0.32 | 0.082 | 0.345 | 0.467 | 0.198 |
| Writer1515 | 0.528 | 0.288 | 0.186 | 0.262 | 0.311 | 0.142 | 0.606 | 0.431 | 0.252 |
| Writer1616 | 0.023 | 0.008 | 0.006 | 0.06 | 0.019 | 0.014 | 0.186 | 0.218 | 0.1 |
| Writer1717 | 0.327 | 0.205 | 0.126 | 0.279 | 0.071 | 0.057 | 0.374 | 0.494 | 0.213 |
| Writer1818 | 0.309 | 0.33 | 0.159 | 0.206 | 0.132 | 0.08 | 0.429 | 0.303 | 0.178 |
| Writer1919 | 0.093 | 0.324 | 0.073 | 0.076 | 0.116 | 0.046 | 0.328 | 0.446 | 0.189 |
| Writer2020 | 0.197 | 0.177 | 0.093 | 0.12 | 0.085 | 0.05 | 0.375 | 0.437 | 0.202 |
| Writer2121 | 0.333 | 0.181 | 0.117 | 0.156 | 0.175 | 0.083 | 0.344 | 0.187 | 0.121 |
| Writer2222 | 0.259 | 0.413 | 0.159 | 0.104 | 0.164 | 0.064 | 0.596 | 0.396 | 0.238 |
| Writer2323 | 0.328 | 0.151 | 0.103 | 0.142 | 0.139 | 0.07 | 0.366 | 0.318 | 0.17 |
| Writer2424 | 0.261 | 0.231 | 0.122 | 0.163 | 0.192 | 0.088 | 0.394 | 0.364 | 0.189 |
| Writer2525 | 0.42 | 0.277 | 0.167 | 0.13 | 0.185 | 0.076 | 0.555 | 0.51 | 0.266 |
| Writer2626 | 0.295 | 0.17 | 0.108 | 0.085 | 0.32 | 0.067 | 0.306 | 0.203 | 0.122 |
| Writer2727 | 0.311 | 0.236 | 0.134 | 0.117 | 0.177 | 0.07 | 0.425 | 0.486 | 0.227 |
| Writer2828 | 0.591 | 0.471 | 0.262 | 0.207 | 0.316 | 0.125 | 0.626 | 0.726 | 0.336 |
| Writer2929 | 0.388 | 0.698 | 0.249 | 0.125 | 0.41 | 0.096 | 0.594 | 0.53 | 0.28 |
| Writer3030 | 0.439 | 0.348 | 0.194 | 0.184 | 0.295 | 0.114 | 0.715 | 0.529 | 0.304 |
| Writer3131 | 0.578 | 0.358 | 0.221 | 0.217 | 0.504 | 0.152 | 0.795 | 0.547 | 0.324 |
| Writer3232 | 0.311 | 0.115 | 0.084 | 0.099 | 0.058 | 0.036 | 0.219 | 0.348 | 0.134 |
| Writer3333 | 0.506 | 0.432 | 0.233 | 0.331 | 0.209 | 0.128 | 0.626 | 0.635 | 0.315 |
| Writer3434 | 0.382 | 0.372 | 0.188 | 0.356 | 0.083 | 0.067 | 0.373 | 0.727 | 0.247 |
| Writer3535 | 0.347 | 0.104 | 0.08 | 0.19 | 0.04 | 0.033 | 0.373 | 0.334 | 0.176 |
| Writer3636 | 0.066 | 0.067 | 0.033 | 0.046 | 0.026 | 0.016 | 0.273 | 0.246 | 0.129 |
| Writer3737 | 0.119 | 0.093 | 0.052 | 0.16 | 0.054 | 0.04 | 0.219 | 0.309 | 0.128 |
| Writer3838 | 0.449 | 0.448 | 0.224 | 0.275 | 0.217 | 0.121 | 0.489 | 0.552 | 0.259 |
| Writer3939 | 0.152 | 0.243 | 0.094 | 0.142 | 0.049 | 0.036 | 0.237 | 0.332 | 0.138 |
| Writer4040 | 0.196 | 0.21 | 0.101 | 0.117 | 0.06 | 0.04 | 0.325 | 0.3 | 0.156 |
| Writer4141 | 0.167 | 0.203 | 0.092 | 0.124 | 0.023 | 0.02 | 0.335 | 0.256 | 0.145 |
| Writer4242 | 0.7 | 0.6 | 0.323 | 0.338 | 0.321 | 0.165 | 0.488 | 0.675 | 0.283 |
| Writer4343 | 0.252 | 0.37 | 0.15 | 0.213 | 0.111 | 0.073 | 0.319 | 0.428 | 0.183 |
| Writer4444 | 0.473 | 0.245 | 0.161 | 0.207 | 0.235 | 0.11 | 0.639 | 0.52 | 0.287 |
| Writer4545 | 0.542 | 0.438 | 0.242 | 0.236 | 0.208 | 0.111 | 0.672 | 0.486 | 0.282 |
| Writer4646 | 0.248 | 0.29 | 0.134 | 0.149 | 0.115 | 0.065 | 0.331 | 0.328 | 0.165 |
| Writer4747 | 0.269 | 0.186 | 0.11 | 0.186 | 0.135 | 0.078 | 0.48 | 0.451 | 0.233 |
| Writer4848 | 0.488 | 0.466 | 0.238 | 0.348 | 0.232 | 0.139 | 0.694 | 0.497 | 0.29 |
| Writer4949 | 0.149 | 0.092 | 0.057 | 0.122 | 0.091 | 0.052 | 0.228 | 0.208 | 0.109 |
| Writer5050 | 0.71 | 0.635 | 0.335 | 0.47 | 0.48 | 0.238 | 0.703 | 0.649 | 0.337 |
| Writer5151 | 0.704 | 0.568 | 0.314 | 0.44 | 0.425 | 0.216 | 0.593 | 0.609 | 0.3 |
| Writer5252 | 0.373 | 0.324 | 0.173 | 0.165 | 0.147 | 0.078 | 0.482 | 0.494 | 0.244 |
| Writer5353 | 0.265 | 0.276 | 0.135 | 0.242 | 0.152 | 0.093 | 0.369 | 0.532 | 0.218 |
| Writer5454 | 0.223 | 0.389 | 0.142 | 0.17 | 0.178 | 0.087 | 0.404 | 0.605 | 0.242 |
| Writer5555 | 0.303 | 0.525 | 0.192 | 0.212 | 0.182 | 0.098 | 0.619 | 0.477 | 0.269 |
| Writer5656 | 0.42 | 0.243 | 0.154 | 0.236 | 0.169 | 0.098 | 0.018 | 0.007 | 0.005 |
| Writer5757 | 0.453 | 0.661 | 0.269 | 0.278 | 0.308 | 0.146 | 0.551 | 0.725 | 0.313 |
| Writer5858 | 0.485 | 0.217 | 0.15 | 0.253 | 0.157 | 0.097 | 0.488 | 0.565 | 0.262 |
| Writer5959 | 0.315 | 0.372 | 0.171 | 0.177 | 0.104 | 0.065 | 0.48 | 0.362 | 0.206 |
| Writer6060 | 0.452 | 0.399 | 0.212 | 0.429 | 0.137 | 0.104 | 0.606 | 0.372 | 0.23 |
| Writer6161 | 0.437 | 0.485 | 0.23 | 0.237 | 0.176 | 0.101 | 0.799 | 0.58 | 0.336 |
| Writer6262 | 0.802 | 0.455 | 0.29 | 0.449 | 0.427 | 0.219 | 0.742 | 0.788 | 0.382 |
| Writer6363 | 0.453 | 0.674 | 0.271 | 0.464 | 0.365 | 0.204 | 0.504 | 0.65 | 0.284 |
| Writer6464 | 0.833 | 0.95 | 0.444 | 0.651 | 0.673 | 0.331 | 0.886 | 0.82 | 0.426 |
| Writer6565 | 0.459 | 0.518 | 0.243 | 0.4 | 0.198 | 0.132 | 0.71 | 0.379 | 0.247 |
| Writer6666 | 0.185 | 0.307 | 0.116 | 0.12 | 0.094 | 0.053 | 0.391 | 0.557 | 0.23 |
| Writer6767 | 0.143 | 0.242 | 0.09 | 0.161 | 0.149 | 0.078 | 0.254 | 0.157 | 0.097 |
| Writer6868 | 0.253 | 0.485 | 0.166 | 0.18 | 0.207 | 0.096 | 0.415 | 0.55 | 0.237 |
| Writer6969 | 0.273 | 0.129 | 0.088 | 0.146 | 0.336 | 0.102 | 0.399 | 0.315 | 0.176 |
| Writer7070 | 0.153 | 0.069 | 0.047 | 0.131 | 0.186 | 0.077 | 0.323 | 0.142 | 0.099 |
| Writer7171 | 0.416 | 0.069 | 0.059 | 0.22 | 0.181 | 0.099 | 0.498 | 0.581 | 0.268 |
| Writer7272 | 0.327 | 0.31 | 0.159 | 0.245 | 0.144 | 0.091 | 0.276 | 0.318 | 0.148 |
| Writer7373 | 0.284 | 0.224 | 0.125 | 0.215 | 0.098 | 0.067 | 0.483 | 0.298 | 0.184 |
| Writer7474 | 0.351 | 0.593 | 0.221 | 0.252 | 0.269 | 0.13 | 0.469 | 0.734 | 0.286 |
| Writer7575 | 0.223 | 0.456 | 0.15 | 0.194 | 0.231 | 0.105 | 0.345 | 0.435 | 0.192 |
| Writer7676 | 0.379 | 0.42 | 0.199 | 0.117 | 0.215 | 0.076 | 0.637 | 0.529 | 0.289 |
| Writer7777 | 0.464 | 0.438 | 0.225 | 0.274 | 0.288 | 0.14 | 0.553 | 0.639 | 0.296 |
| Writer7878 | 0.665 | 0.617 | 0.32 | 0.566 | 0.308 | 0.199 | 0.737 | 0.805 | 0.385 |
| Writer7979 | 0.135 | 0.114 | 0.062 | 0.075 | 0.128 | 0.047 | 0.34 | 0.351 | 0.173 |
| Writer8080 | 0.221 | 0.398 | 0.142 | 0.152 | 0.164 | 0.079 | 0.404 | 0.335 | 0.183 |
| Writer8181 | 0.523 | 0.691 | 0.298 | 0.319 | 0.294 | 0.153 | 0.421 | 0.665 | 0.258 |
| Writer8282 | 0.373 | 0.354 | 0.182 | 0.185 | 0.162 | 0.086 | 0.319 | 0.22 | 0.13 |
| Writer8383 | 0.325 | 0.472 | 0.193 | 0.192 | 0.219 | 0.102 | 0.462 | 0.472 | 0.234 |
| Writer8484 | 0.824 | 0.871 | 0.423 | 0.57 | 0.622 | 0.297 | 0.804 | 0.841 | 0.411 |
| Writer8585 | 0.667 | 0.355 | 0.232 | 0.236 | 0.289 | 0.13 | 0.492 | 0.335 | 0.199 |
| Writer8686 | 0.286 | 0.136 | 0.092 | 0.12 | 0.04 | 0.03 | 0.415 | 0.319 | 0.18 |
| Writer8787 | 0.491 | 0.693 | 0.288 | 0.464 | 0.118 | 0.094 | 0.499 | 0.645 | 0.281 |
| Writer8888 | 0.434 | 0.521 | 0.237 | 0.216 | 0.304 | 0.126 | 0.647 | 0.553 | 0.298 |
| Writer8989 | 0.518 | 0.551 | 0.267 | 0.35 | 0.342 | 0.173 | 0.609 | 0.73 | 0.332 |
| Writer9090 | 0.619 | 0.866 | 0.361 | 0.469 | 0.449 | 0.229 | 0.687 | 0.794 | 0.368 |
| Writer9100 | 0.856 | 0.884 | 0.435 | 0.576 | 0.538 | 0.278 | 0.836 | 0.851 | 0.422 |
| Writer9101 | 0.107 | 0.015 | 0.013 | 0.023 | 0.021 | 0.011 | 0.504 | 0.404 | 0.224 |
| Writer9102 | 0.426 | 0.504 | 0.231 | 0.332 | 0.17 | 0.112 | 0.583 | 0.676 | 0.313 |
| Writer9103 | 0.412 | 0.491 | 0.224 | 0.371 | 0.153 | 0.108 | 0.557 | 0.724 | 0.315 |
| Writer9104 | 0.517 | 0.269 | 0.177 | 0.212 | 0.361 | 0.134 | 0.516 | 0.551 | 0.266 |
| Writer9105 | 0.671 | 0.661 | 0.333 | 0.391 | 0.38 | 0.193 | 0.7 | 0.764 | 0.365 |
| Writer9191 | 0.327 | 0.199 | 0.124 | 0.129 | 0.164 | 0.072 | 0.381 | 0.4 | 0.195 |
| Writer9292 | 0.278 | 0.185 | 0.111 | 0.179 | 0.132 | 0.076 | 0.448 | 0.343 | 0.194 |
| Writer9393 | 0.358 | 0.662 | 0.232 | 0.314 | 0.406 | 0.177 | 0.527 | 0.581 | 0.276 |
| Writer9494 | 0.296 | 0.515 | 0.188 | 0.154 | 0.166 | 0.08 | 0.484 | 0.617 | 0.271 |
| Writer9595 | 0.593 | 0.716 | 0.324 | 0.453 | 0.31 | 0.184 | 0.609 | 0.752 | 0.336 |
| Writer9696 | 0.475 | 0.266 | 0.171 | 0.256 | 0.16 | 0.098 | 0.537 | 0.415 | 0.234 |
| Writer9797 | 0.511 | 0.156 | 0.12 | 0.181 | 0.328 | 0.117 | 0.411 | 0.687 | 0.257 |
| Writer9898 | 0.309 | 0.493 | 0.19 | 0.232 | 0.187 | 0.103 | 0.455 | 0.465 | 0.23 |
| Writer9999 | 0.312 | 0.411 | 0.177 | 0.276 | 0.241 | 0.129 | 0.485 | 0.352 | 0.204 |

# REFERENCES

[1] H. Harralson and L. Miller, *Huber and Headrick’s Handwriting Identification: Facts and Fundamentals*, 2nd ed. CRC PRess, 2017.

[2] S. N. Srihari, Sung-Hyuk Cha, H. Arora, and S. Lee, “Individuality of Handwriting,” *Journal of Forensic Science*, vol. 47, no. 4, p. 17, Jul. 2002.

[3] C. Djeddi, S. Al-Maadeed, I. Siddiqi, G. Abdeljalil, S. He, and Y. Akbari, “ICFHR 2018 Competition on Multi-Script Writer Identification,” in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Aug. 2018, pp. 506–510. doi: 10.1109/ICFHR-2018.2018.00094.

[4] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 3rd edition. Beijing Boston Farnham Sebastopol Tokyo: O’Reilly Media, 2022.

[5] A. Foroozandeh, A. Askari Hemmat, and H. Rabbani, “Offline Handwritten Signature Verification and Recognition Based on Deep Transfer Learning,” in *2020 International Conference on Machine Vision and Image Processing (MVIP)*, Feb. 2020, pp. 1–7. doi: 10.1109/MVIP49855.2020.9187481.

[6] H. T. Nguyen, C. T. Nguyen, T. Ino, B. Indurkhya, and M. Nakagawa, “Text-independent writer identification using convolutional neural network,” *Pattern Recognition Letters*, vol. 121, pp. 104–112, Apr. 2019, doi: 10.1016/j.patrec.2018.07.022.

[7] M. A. Shaikh, M. Chauhan, J. Chu, and S. Srihari, “Hybrid Feature Learning for Handwriting Verification,” Aug. 2018, pp. 187–192. doi: 10.1109/ICFHR-2018.2018.00041.

[8] X. Wu, Y. Tang, and W. Bu, “Offline Text-Independent Writer Identification Based on Scale Invariant Feature Transform,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 3, pp. 526–536, Mar. 2014, doi: 10.1109/TIFS.2014.2301274.

[9] R. Jain and D. Doermann, “Offline Writer Identification Using K-Adjacent Segments,” in *2011 International Conference on Document Analysis and Recognition*, Sep. 2011, pp. 769–773. doi: 10.1109/ICDAR.2011.159.

[10] J. Tan, J. Lai, C. Wang, and M. Feng, “A Stroke Shape and Structure Based Approach for Off-line Chinese Handwriting Identification,” *IJISA*, vol. 3, no. 2, pp. 1–8, Mar. 2011, doi: 10.5815/ijisa.2011.02.01.

[11] V. Pervouchine, G. Leedham, and K. Melikhov, “Three-stage handwriting stroke extraction method with hidden loop recovery,” in *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)*, Aug. 2005, pp. 307-311 Vol. 1. doi: 10.1109/ICDAR.2005.241.

[12] Z. Guo, L. Zhang, and D. Zhang, “A Completed Modeling of Local Binary Pattern Operator for Texture Classification,” *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, Jun. 2010, doi: 10.1109/TIP.2010.2044957.

[13] A. J. Newell and L. D. Griffin, “Writer identification using oriented Basic Image Features and the Delta encoding,” *Pattern Recognition*, vol. 47, no. 6, pp. 2255–2265, Jun. 2014, doi: 10.1016/j.patcog.2013.11.029.

[14] V. Christlein and A. Maier, “Encoding CNN Activations for Writer Recognition,” in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, Apr. 2018, pp. 169–174. doi: 10.1109/DAS.2018.9.

[15] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 3304–3311. doi: 10.1109/CVPR.2010.5540039.

[16] F. Abbas, A. Gattal, C. Djeddi, I. Siddiqi, A. Bensefia, and K. Saoudi, “Texture feature column scheme for single- and multi-script writer identification,” *IET Biometrics*, vol. 10, no. 2, pp. 179–193, 2021, doi: 10.1049/bme2.12010.

[17] S. He, M. Wiering, and L. Schomaker, “Junction detection in handwritten documents and its application to writer identification,” *Pattern Recognition*, vol. 48, no. 12, pp. 4036–4048, Dec. 2015, doi: 10.1016/j.patcog.2015.05.022.

[18] A. Semma, Y. Hannad, I. Siddiqi, S. Lazrak, and M. E. Y. E. Kettani, “Feature learning and encoding for multi-script writer identification,” *IJDAR*, vol. 25, no. 2, pp. 79–93, Jun. 2022, doi: 10.1007/s10032-022-00394-8.

[19] A. A. Ahmed, H. R. Hasan, F. A. Hameed, and O. I. Al-Sanjary, “Writer Identification on Multi-Script Handwritten Using Optimum Features,” *KJAR*, vol. 2, no. 3, pp. 178–185, Aug. 2017, doi: 10.24017/science.2017.3.64.

[20] J. W. Kim, K. I. Kim, B. J. Choi, and H. J. Kim, “Decomposition of Chinese character into strokes using mathematical morphology,” *Pattern Recognition Letters*, vol. 20, no. 3, pp. 285–292, Mar. 1999, doi: 10.1016/S0167-8655(98)00147-0.

[21] J. Chen, B. Li, and X. Xue, “Zero-Shot Chinese Character Recognition with Stroke-Level Decomposition.” arXiv, Jun. 22, 2021. doi: 10.48550/arXiv.2106.11613.

[22] C.-L. Liu, I.-J. Kim, and J. H. Kim, “Model-based stroke extraction and matching for handwritten Chinese character recognition,” *Pattern Recognition*, vol. 34, no. 12, pp. 2339–2352, Dec. 2001, doi: 10.1016/S0031-3203(00)00165-5.

[23] I.-J. Kim, C.-L. Liu, and J.-H. Kim, “Stroke-guided pixel matching for handwritten Chinese character recognition,” in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR ’99 (Cat. No.PR00318)*, Sep. 1999, pp. 665–668. doi: 10.1109/ICDAR.1999.791875.

[24] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, Art. no. 1, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” arXiv, Dec. 10, 2015. doi: 10.48550/arXiv.1512.03385.

[26] T. Dozat, “INCORPORATING NESTEROV MOMENTUM INTO ADAM,” 2016.

# BIOGRAPHICAL SKETCH

Name of Author: Joshua Jude Thomas

Graduate and Undergraduate Schools Attended:

Coastal Alabama Community College, Bay Minette, Alabama

University of South Alabama, Mobile, Alabama

Degrees Awarded:

Bachelor of Science in Computer Science, 2021, Mobile Alabama

Associate in Science, 2018, Bay Minette, Alabama

1. The CERUG dataset is used for this study and the results are compared to the presenting paper. [↑](#footnote-ref-2)
2. Due to the color-map used, the stem of the d in ‘and’ may not appear visible. [↑](#footnote-ref-3)
3. Credit to <https://github.com/Image-Py/sknw/tree/master/sknw> for providing the code to generate a graph theory representation. [↑](#footnote-ref-4)
4. The Experimental Results, Publication, Code, and Figures can be found at <https://github.com/justjude97/MultiScript-Handwriting-Identification-with-Stroke-Decomposition> [↑](#footnote-ref-5)