5-2024

Multi-Script Handwriting Identification by Fragmenting Strokes

Joshua Jude Thomas

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

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CERUG	Chinese-English database of the University of Groningen
CERUG-CN	CERUG Chinese Partition
CERUG-EN	CERUG English Partition
CERUG-MIXED	CERUG Mixed Partition
CNN	Convolutional Neural Network
GSC	Gradient Scale Concavity
HIM	Handwriting Identification Model
HVM	Handwriting Verification Model
ICFHR	International Conference on Frontiers in Handwriting Recognition
KD	Known Document
KNN	K-Nearest Neighbors
MSWI	Multi-Script Writer Identification
OBIF	Oriented Basic Image Features
OCR	Optical Character Recognition
QD	Questioned Document
SIFT	Scale-Invariant Feature Transform
U-LBP	Uniform-Complete Local Binary Patterns
VLAD	Vector of Locally Aggregated Descriptors

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, by segmenting them into sub-parts. Character simplification is performed through splitting the character by branching-points and end-points, a process called stroke fragmentation in this study. The resulting sub-parts of the character are called stroke fragments and are evaluated individually to identify the writer. 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, Chinese, etc.) may have common shapes which can be extracted and used in the handwriting identification process. The effectiveness of the stroke fragmentation 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 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 uses a similar process, but mainly works 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, or a model of the handwriting habits of the writers. 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 1 [2]. Much of the research literature falls under one of the two categories. The goal of the HVM is to determine whether the document was written by the same person or not (2-class classification). The extracted features 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 produce a direct classification but can also produce vector of probabilities that the document belongs to any of the writers (e.g. SoftMax score). The HIM is typically the more popular of the two frameworks and is the framework used in this study.

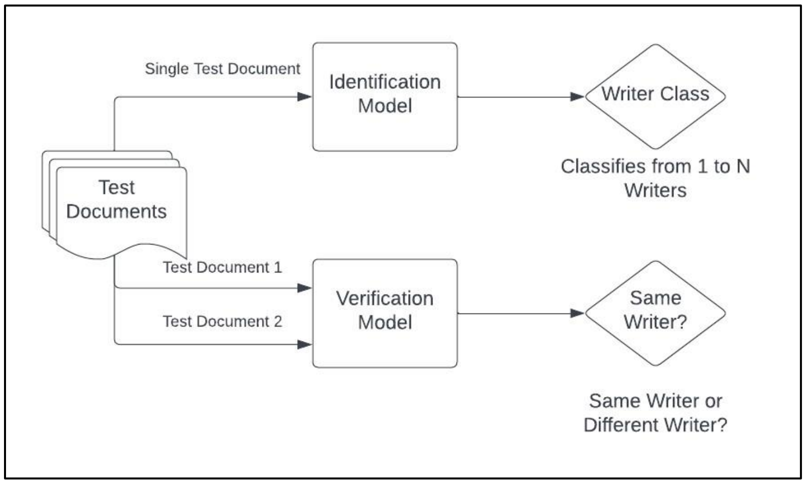

Figure 1. The Handwriting Identification Model vs the Handwriting Verification Model.

1.2 Multi-Script Writer Identification

In **Multi-Script Handwriting Identification** (MSWI), writers are not limited to one language or writing-script. A **writing-script** is a set of characters used to inscribe a written language to a medium, such as paper. A writer can produce handwritten documents in languages (English, Chinese, etc.) that use different writing-scripts, 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 International Conference on Frontiers in Handwriting Recognition (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 features for use in 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 writing-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 Bengali and Tamil which typically have a flowing, 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.

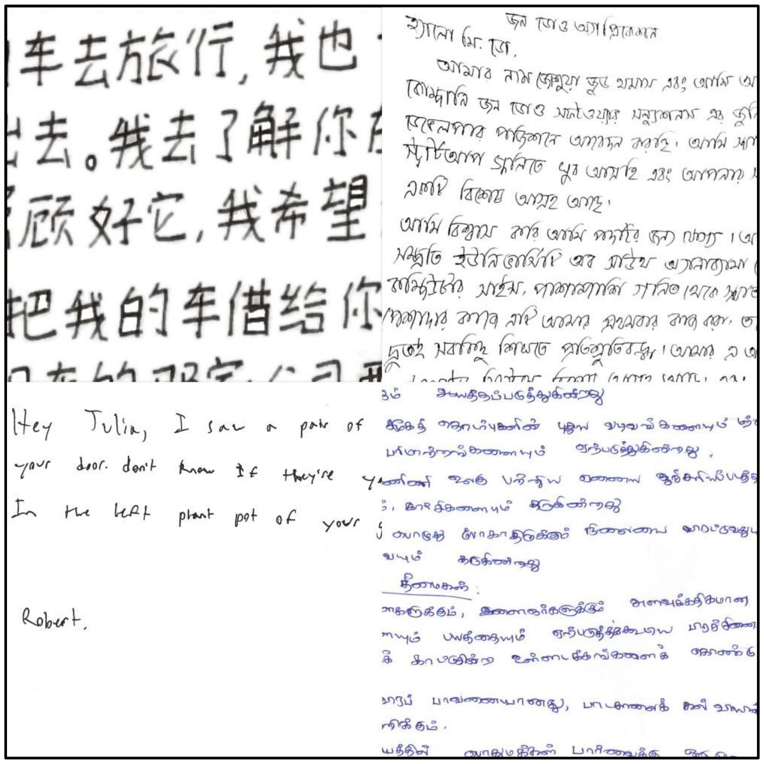

Figure 2. Writing-scripts of three different origins that look very different on the visual level. (From left to right) Chinese, Bengali, English, and Tamil.

1.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are deep neural networks specialized for processing spatial information such as images [4]. They are very good architectures for classification tasks on images and have been used in handwriting identification research to learn and extract the feature of handwriting [3], [5]. However, one problem of CNN, shared by all deep learning models, is the sheer amount of data required to train them properly. A standard CNN trained to extract features for handwriting identification would have to have a large amount of 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 Writer Identification (MSWI) by breaking down the characters of different writing-scripts into simpler shapes, or sub-parts. The concept is that the sub-parts of a handwritten character may be more common across writing-scripts and could be exploited to compare documents across different writing-scripts. This study uses a CNN model as its HIM to produce probabilities of the writer of a given document in a multi-script dataset. The novelty presented in this study is defined in a process called Stroke Fragmentation, which breaks characters of a document into multiple, simpler sub-parts which are called stroke fragments. Stroke Fragmentation is called so because this process typically results in fragments of a writing stroke, which is a portion of a character caused by a writing utensil being pressed down on the paper, then lifted, once. This study also performs a special case of MSWI, described in the competition paper [3], where the HWI is trained on one writing-script and evaluated on another, different writing-script.

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 present new types of 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. "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, Scale-Invariant Feature Transform (SIFT) [8] descriptors and Gradient Scale Concavity (GSC) [2] 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 [9]. Jain, Rajiv and Doermann, David approximate the contours of handwritten characters into "k-adjacent" segments [10]. The contours are approximated into lines by a line-fitting algorithm and then sets of 2-to-4 line segments are taken and described through a feature vector. Tan et al. extract features from a bounding box and a bounding quadrilateral of the handwriting characters for writer identification [11]. Pervouchine et al. extract handwriting strokes via modeling with cubic splines [12]. 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 called 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) [13] for its feature extraction while the LIMPAF-II uses Oriented Basic Image Features (OBIF) [14]. 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 different writing-scripts into a more uniform representation. The Nuremberg system was actually based on another paper [15] 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 Vector of Locally Aggregated Descriptors (VLAD) [16]. Abbas *et al* combines LBP and OBIF to create a histogram of both over the whole range of the document [17]. He et al. tested the power of handwriting junctions on the writer identification task [18]. 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]. Semma et al. use

[1] The CERUG dataset is used for this study and the results are compared to the presenting paper.

CNN features encoded into modified VLAD vectors [16] for handwriting identification [19]. 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 [20].

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 [12], 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 and the identification of branching points.

Kim et al. decompose Chinese characters into individual strokes by first performing a morphological thinning to reduce each character to a single pixel width [21]. They then segment the characters based on branching points (areas where strokes overlap) and excessively curved segments, similar to the critical points in [18]. The segments are grown morphologically using two modifications on a morphological dilation which use vectors both parallel, elongation, and perpendicular, 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. 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 [21]. 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 [22]. Liu et al. use a model based approach to stroke decomposition [23]. 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 [24]. 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 (masked sub-images) for the Handwriting Identification Model (HIM). It takes the characters of a handwritten document and produces one or more sub-parts of a character, called **stroke fragments** in this study, each of which are stored in a sub-image. The sub-images are masked to contain only that stroke fragment. For a handwriting dataset consisting of a set of digitized handwritten document images (scanned or photographed), the stroke fragmentation process takes a document image and extracted multiple masked sub-images, each containing a stroke fragment, and groups them into different datasets based on the writing-script contained in the document.

To extract the stroke fragments, a set of masks is constructed for each document. These masks are constructed by segmenting the foreground pixels (pixels representing the handwritten characters) of a binary representation of the document along detected stroke fragments. Each mask represents a stroke fragment to be extracted from the document. Specifically, they represent the bounding box coordinates of the stroke fragment in the document, as well as the corresponding pixels to be extracted. Sub-images containing the stroke fragment are extracted from the document with the bounding box coordinates of the masks, and then unneeded pixel values (not corresponding to the pixels in the mask) are removed. The extracted stroke fragments of a dataset are grouped together: first by the writing-script used in the document and then by the writer class. This process results in a number of script-specific datasets from the original, depending on the number of writing-scripts contained in that original dataset.

For an experiment, two writing-script datasets are chosen for the training and evaluation of the HIM, evaluating on one and testing on the other. Evaluating the trained model on a different writing-script, not yet encountered by the HIM, will give insight into how well the features extracted by the model are at MSWI. The specific model used as the HIM is a CNN copying the ResNet-50 model architecture. The architecture is modified to classify the writers of a given dataset, and is trained on a single writing script. The evaluation is performed on a separate writing-script and uses Top-1 Accuracy (Categorical Accuracy), Top-10 Accuracy, Precision, and Recall, as the evaluation metrics.

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** which are defined as the branching-points and end-points of the character. The end goal of the fragmentation process is a collection of stroke fragments extracted from each document that could then be used in 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: $G = \{V, E\}$, where V is a set containing vertices representing the critical points of the document and E is a multi-set of pairs $E = \{uv \mid u, v \in V\}$. Each pair in E is an edge between two critical point vertices u, v and represent that those two points are connected by a sub-part of the character, the stroke fragment that we want to extract. The stroke fragments we want to extract can then be thought of as the edges between the vertices of the document graph. Note that the actual pixel coordinates of the portion we want to extract as stroke fragments are not represented as the edges or nodes but are instead included as attributes of them. Figure 3 shows a simplification of the process that details how the masks representing stroke fragments are made.

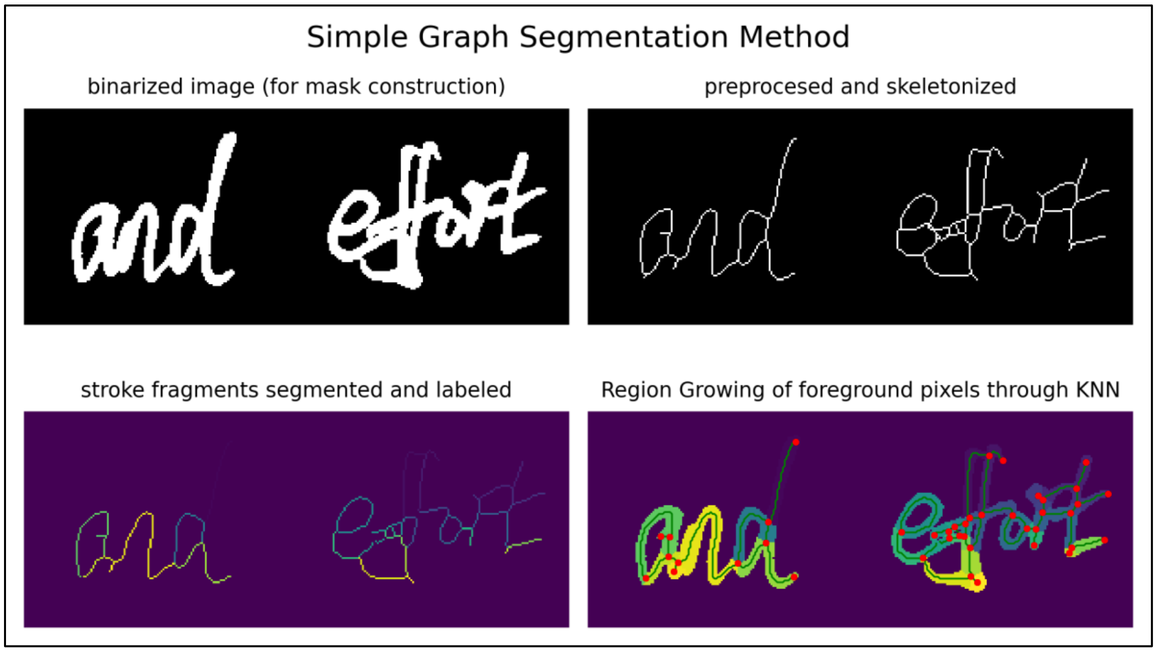
Simple Graph Segmentation Method
binarized image (for mask construction)
preprocesed and skeletonized
stroke fragments segmented and labeled
Region Growing of foreground pixels through KNN

Figure 3. Visualization of mask construction. Starting with a raw binary image to construct the masks. 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].

3.1.1 Stroke Fragmentation Process

This section shows the stroke fragmentation process in detail. The process consists of seven steps, which are: preprocessing, skeletonization, graph conversion, stroke fragment skeleton labeling, mask segmentation, filtering, and feature extraction.

3.1.1.1 Preprocessing.

For a given scanned handwritten document, it is converted into an inverted grayscale representation, and a binary representation. The grayscale representation of the document is kept for the feature extraction stage, at the end of the stroke fragmentation process, and is inverted, meaning that the pixel values are changed to $i = \max(I) - i$

[2] Due to the color-map used, the stem of the d in ‘and’ may not appear visible.

where i is the grayscale level of any given pixel and max (I) is the maximum possible grayscale value in the image. This results in the grayscale values of the background (typically white) and foreground (typically black) being reversed. Using a background grayscale value that is, or is close to, zero makes it possible to set unwanted foreground pixels (not belonging to the target stroke fragment) to be set to zero themselves. Figure 4 shows the visual difference between an image and its inverted grayscale.

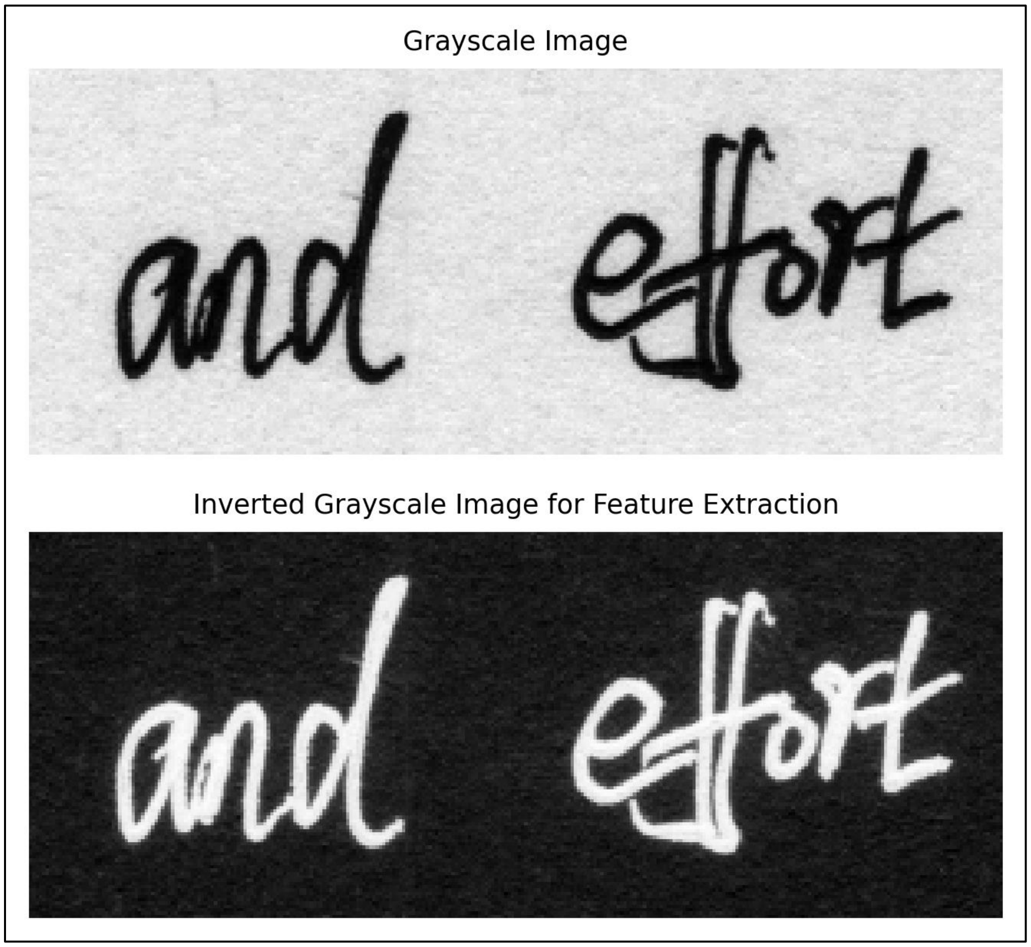

Figure 4. Grayscale of a Document versus the Inverted Grayscale. Inverted Grayscale is used to make stroke fragment extraction easier in step 7.

The binary image is used to form the masks and is obtained by using Otsu's method [25] to threshold the grayscale image. When processing the binary image, an inherent amount of noise may exist depending on both the method and quality of digitization and may influence the graph representation. Possible examples are smudges on the paper, and the visible texture of the paper, etc. A gaussian blur is first applied to the grayscale image, before thresholding, to try and remove some of this noise. note that the blur is not applied to the grayscale image when extracting the stroke fragments.

3.1.1.2 Skeletonization.

A skeletonized version of the binary image is created, such that the handwriting strokes contained in the image are reduced to a single pixel width. The skeletonization of the document attempts to preserve the general shape of the handwritten characters, and is performed by Scikit-Image with the *skimage.morphology.skeletonize* method. Their skeletonization algorithm is based on Zhang and Suen [26], which iteratively removes foreground pixels based on their 3x3 neighborhood.

3.1.1.3 Graph Conversion.

The skeletonized image is converted into a graph theory representation using the Skeleton Network library[3]. The purpose of this step is to segment the skeleton of the characters in the document into stroke fragments. These stroke fragment skeleton segments are then used in the next step to segment the pixels of the binary image into masks. The graph representing a handwritten document is defined such that the vertices of the graph correspond to the critical points in the document: the branching-points and

[3] Credit to https://github.com/Image-Py/sknw/tree/master/sknw for providing the code to generate a graph theory representation.

end-points. A branching point is a point between two crossing strokes in a document and is connected to three or more pixels in the skeleton. An end-point in a document is where a handwriting stroke begins or ends, and is connected to only one pixel in the skeleton. Every pixel that does not fit the above criteria is considered a stroke fragment pixel and is used as the skeleton of the masks. The stroke fragment pixels are stored as attributes to edges connecting two vertices in the graph.

3.1.1.4 Stroke Fragment Skeleton Labeling.

For every edge, the stroke fragment pixels of that edge are assigned an integer label $i \in \{1, 2, \dots, n \in \mathbb{Z}\}$.

3.1.1.5 Mask Segmentation.

The masks of the document are formed by segmenting the foreground pixels of the binary image along the stroke fragment segments, labeled in the previous step, attributed to the edges of the graph. The segmentation is performed via K-Nearest Neighbors (KNN), specifically by Scikit-Learn's *sklearn.neighbors.KNeighborsClassifier* method with the number of neighbors to consider set to three. The stroke fragment segments are used as inputs to KNN. And then, the foreground pixels of the original binary image are labeled based on the three stroke fragment pixels it is closest to (spatially). This results in a partition of the binary image into the different masks, each representing a stroke fragment to be extracted from the inverted grayscale.

3.1.1.6 Filtering.

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 process, and will result in a noisier dataset. In addition to large artefacts, some masks may end up being too small to give any meaningful information. A **filtering criterion** is defined to remove the masks generated by large artefacts, as well as low quality masks, from the final output. The bounding box of each mask is found, and if both width and height are 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, then the mask is rejected and not used in the feature extraction step. This has the consequence that some foreground pixels of the original image are discarded and not used during the training process.

3.1.1.7 Feature Extraction.

Masks that are not rejected are used to extract the corresponding stroke fragments in the inverted grayscale image. The bounding box coordinates are used to define the location of the sub-image in the inverted grayscale, and after the sub-image is obtained, any pixel not coinciding with the foreground pixels in the mask is zeroed out. This results in a masked sub-image containing only the pixels of the stroke fragment. Figure 5 shows a visualization of this, with the mask overlayed on the inverted grayscale and the resulting masked sub-image displayed.

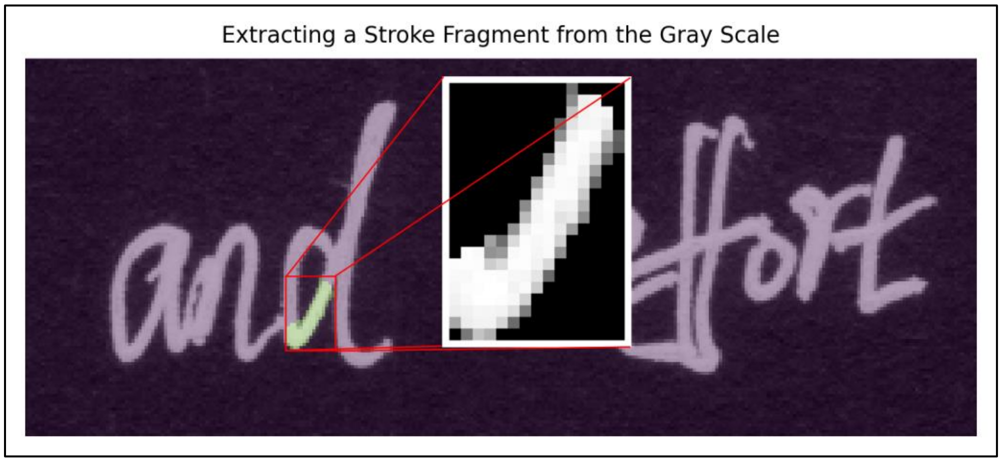

Figure 5. Extracting a Stroke Fragment from the Inverted grayscale. The mask and it's bounding box are used to extract the stroke fragment. Pixels not corresponding to the mask are zeroed out.

3.1.1.8 Summary.

In summary, the stroke fragmentation process takes a scanned handwritten document of a given Dataset and transforms it from a single image into multiple, masked sub-images containing stroke fragments that, if put together, recreate the original character they were extracted from (except for the pixels corresponding to any rejected mask). The sub-images are saved to a directory and grouped into the different writing-scripts of the document (English and Chinese) such that the stroke fragments of each writing-script are grouped into their own sub-directory. Each sub-directory then further groups the stroke fragments into sub-directories of writer classes, which are then used to infer the labels of the writing-script. Thus, the original dataset is processed into a directory of sub-directories representing the different writing-scripts, each of which contains sub-directories of the writer classes. These writing-script subdirectories can be thought of as datasets themselves, and are then used to either train or evaluate a model in the training and evaluation processes, depending on the experiment.

3.2 Training and Evaluation

The following training and evaluation of the Handwriting Identification model (HIM) is a standard process and are implemented with Tensorflow-Keras API. Writing-script directories are loaded as datasets via the *keras.utils.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). Each image must have a standard size when passed through a conventional CNN. However, the extracted sub-images do not have a standard width or weight due to the different sizes of the stroke fragments extracted. The minimum and maximum sizes, as well as the first, second, and third quartiles, of the width and height of the stroke fragment samples are shown in table 1 (extracted from CERUG with eight hundred samples).

Table 1. Five number summaries of the width and heights of the stroke samples, in pixels (from 800 samples). Includes the minimum value, maximum value, as well as the first quartile (25%), second quartile (50%), and third quartile (75%).

	Min	**25%**	**50%**	**75%**	**Max**
Width (px)	5	11	15	22	98
Height (px)	5	12	17	26	71

Both the width and height were resized to 128 pixels to avoid squishing some of the larger stroke segments. The writing-scripts used in the training and evaluation stages 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

The architecture of the Resnet-50 CNN is used as the HIM of this study [27]. The SoftMax layer of resnet-50 is altered to predict the writer classes of the dataset and the weights of the entire model are randomly initialized. The model is fit to the dataset over 40 epochs, using the NADAM optimizer with the default parameters as set in keras [28]. The *keras.callbacks.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 using a 20% random split of the training data.

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, is 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' [18]. 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.

$$\text{Top-1 (Categorical) Accuracy} = \frac{\text{TP}}{\text{TP+FP+TN+FN}}$$

$$\text{Top-10 Accuracy} = \frac{\sum_{i \in I} G(i)}{\text{TP+FP+TN+FN}}$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}$$

Where:

1. TP (True Positive) is the sum of stroke fragments correctly attributed to a writer.
2. FP (False Positive) is the sum of stroke fragments incorrectly attributed to a writer.
3. TN (True Negative) is the sum of instances correctly classified as not being written by a writer.
4. FN (False Negative) is the sum of instances incorrectly classified as not being written by a writer.
5. $G(i)$ is the sum of stroke fragments whose prediction had the correct writer, i, within the top ten most probable writers. Where the set of writers is I

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 [18] contains different writing-script per writer, for 105 writers. The SoftMax layer of the Resnet-50 CNN is altered to have 105 outputs, one for each writer. 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 evaluation set. Experiment 2 uses CERUG-EN as the training set and CERUG-CN as the evaluation set. Experiment 3 merges CERUG-CN and CERUG-EN together for use as the training set and uses CERUG-MIXED as the evaluation set. Note that experiment 3 does not fit the goal of training on one writing set and evaluating on another and is used more for comparison.

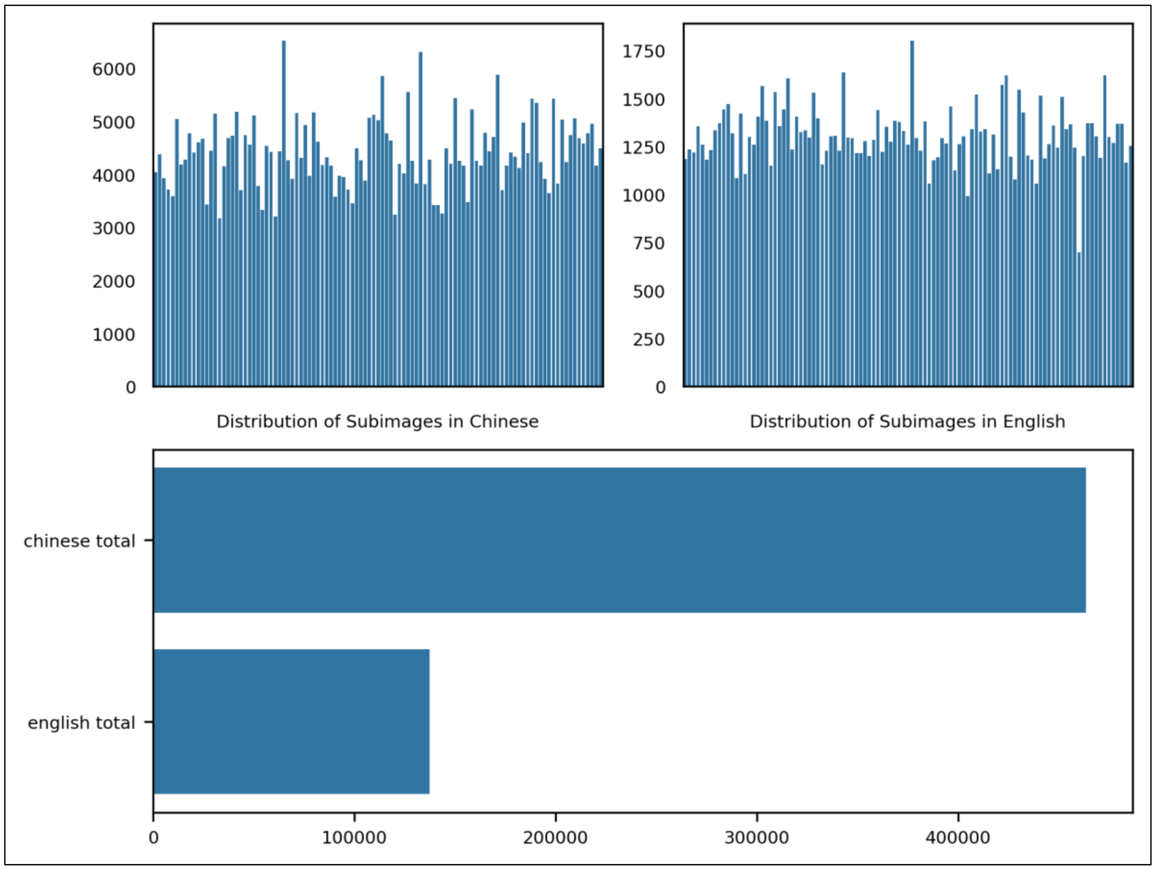

Figure 6. Visualization of class distributions in the CERUG-CN and CERUG-EN datasets vs, the total number of stroke fragments extracted from each. The number of sub-image samples in CERUG-CN greatly outnumbers the number of sub-image samples in CERUG-EN.

Figure 6. 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 different evaluation performances in experiment 1 and 2.

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 the parts of a handwriting stroke in between the critical points of a character. The pixels corresponding to the critical points aren't necessarily in the stroke fragment, since only the stroke fragment pixels attributed to the graph edges are used in the KNN region growing step. Thus, the stroke fragments generated, generally, will be line curves with no branching off parts and few, if any, will be complete loops. Figures 7 and 8 show two different samples of CERUG-CN and CERUG EN, respectively. Each shows a sample of 400 different stroke fragments of varying scales. 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 parts of a character 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 mask.

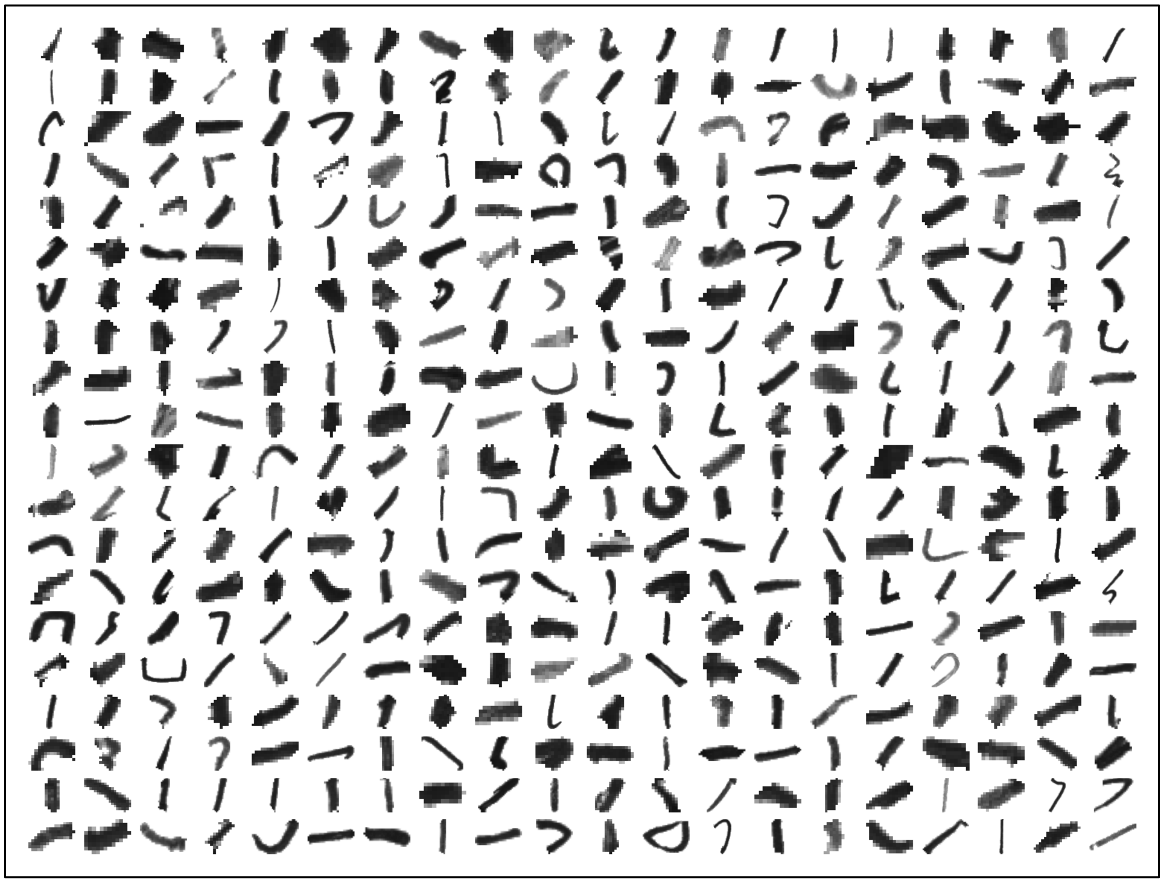

Figure 7. Random Sample of 400 sub-images extracted from CERUG-CN.

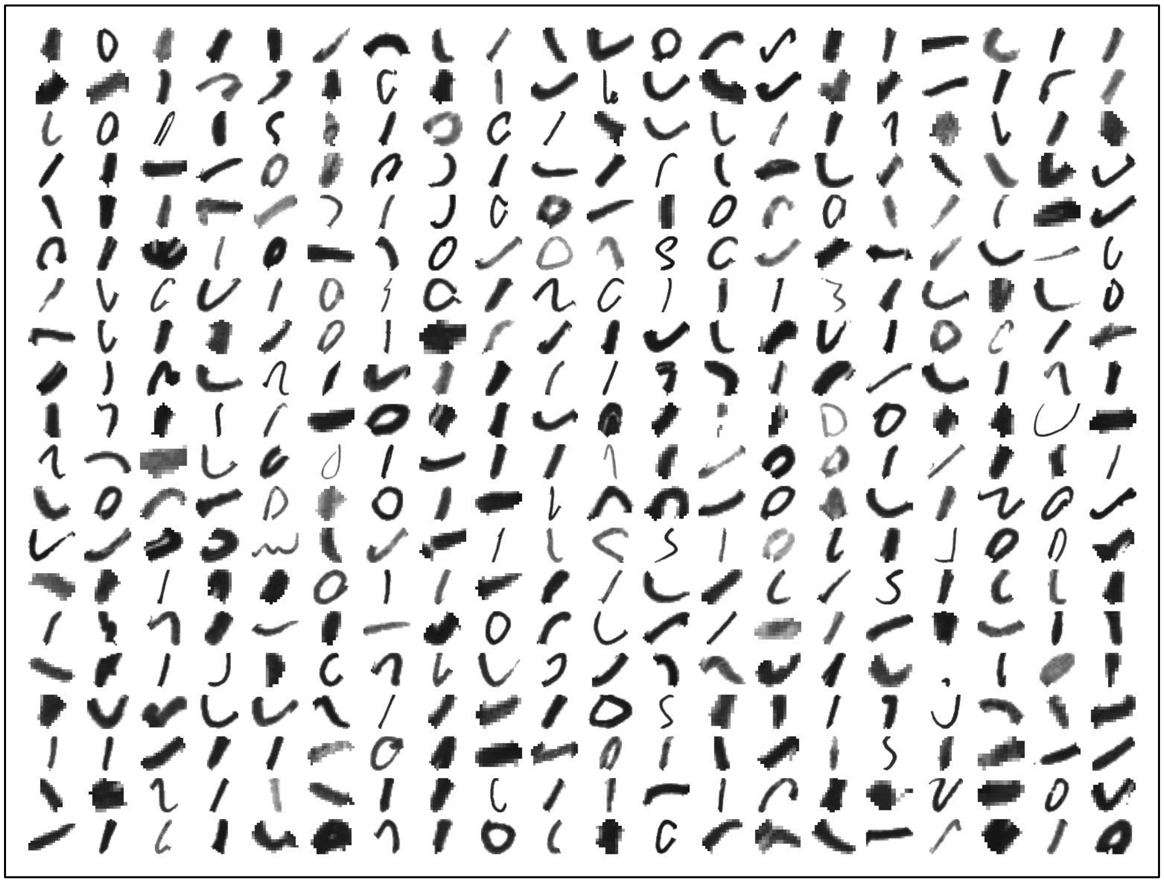

Figure 8. Random Sample of 400 sub-images extracted from CERUG-EN.

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 the Chinese writing in CERUG-CN.

4.2 CERUG Evaluation Results

Table 2 presents the experimental results from all three experiments. The performance results of the junctlets feature (as reported in the paper introducing the feature extraction method [18]) are also added for reference. Note that the two metrics are not exactly comparable because the junctlets feature compute a global feature over the entire document while the HIM of this study evaluates individual stroke fragments of the document. However the results from [18] are still a good baseline metric.

While not performing nearly as well as the junctlets feature proposed in [18], 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 even in the worst case, the correct writer will be in the ten most probable writers, out of 105 writers, for at least 71% of all the stroke fragments in the dataset. 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.

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

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

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. However, experiment 3 does not fit the particular case of MSWI evaluated in this study and is more used as a comparison.

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. To facilitate comparison, the precision and recall are combined into the F1 score, which is a harmonic average between the two.

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

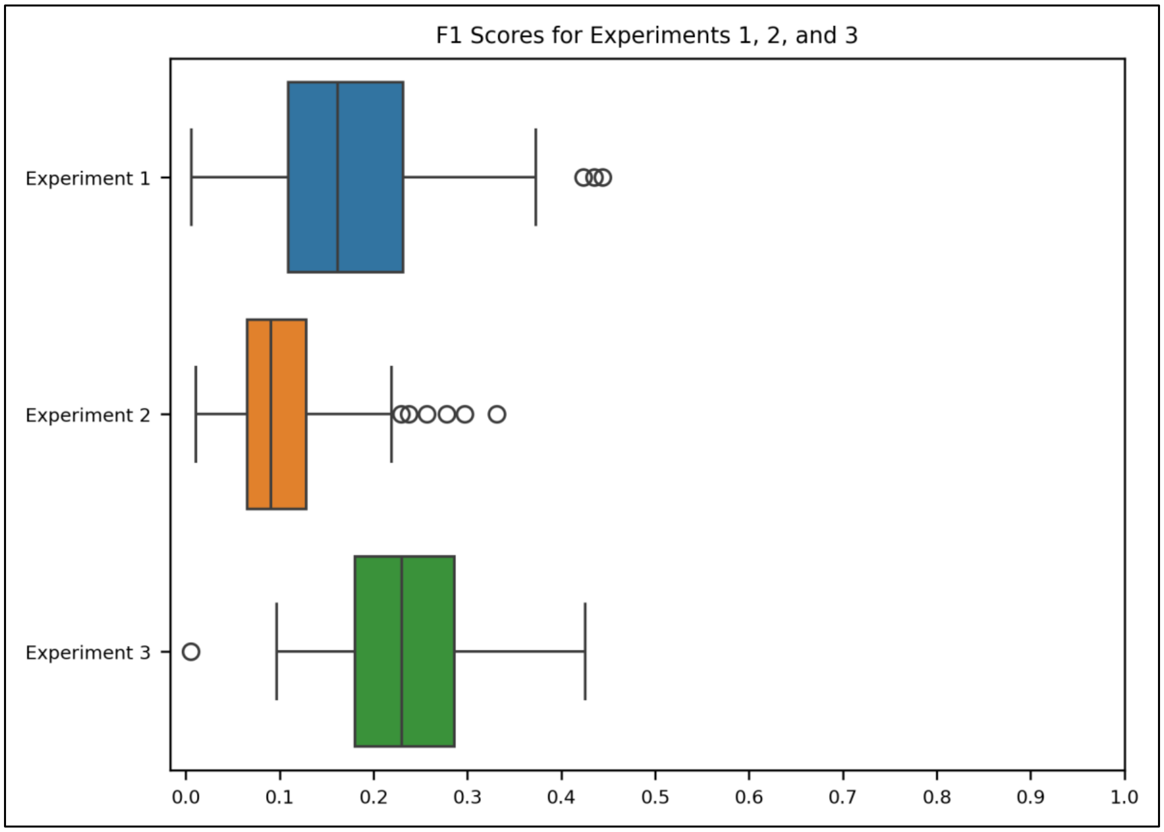

Figure 9. Box Plot of F1 Scores for Experiments 1, 2, and 3.

Figure 9 shows the boxplots of the F1 score for experiments 1, 2, and 3. Experiment 3 uses both CERUG-CN and CERUG-CN as its 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.

Table 3. 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.

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 3 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 (masked sub-images) of the writer classes, per script, and the performance of that class in a particular experiment. Table 4 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.

Table 4. Number of sub-image samples for the worst and best performers in experiments 1 and 2.

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

The following is a visual analysis of the three writers (as performed in 4.1). Figures 10, 11, and 12 (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 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 11 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.

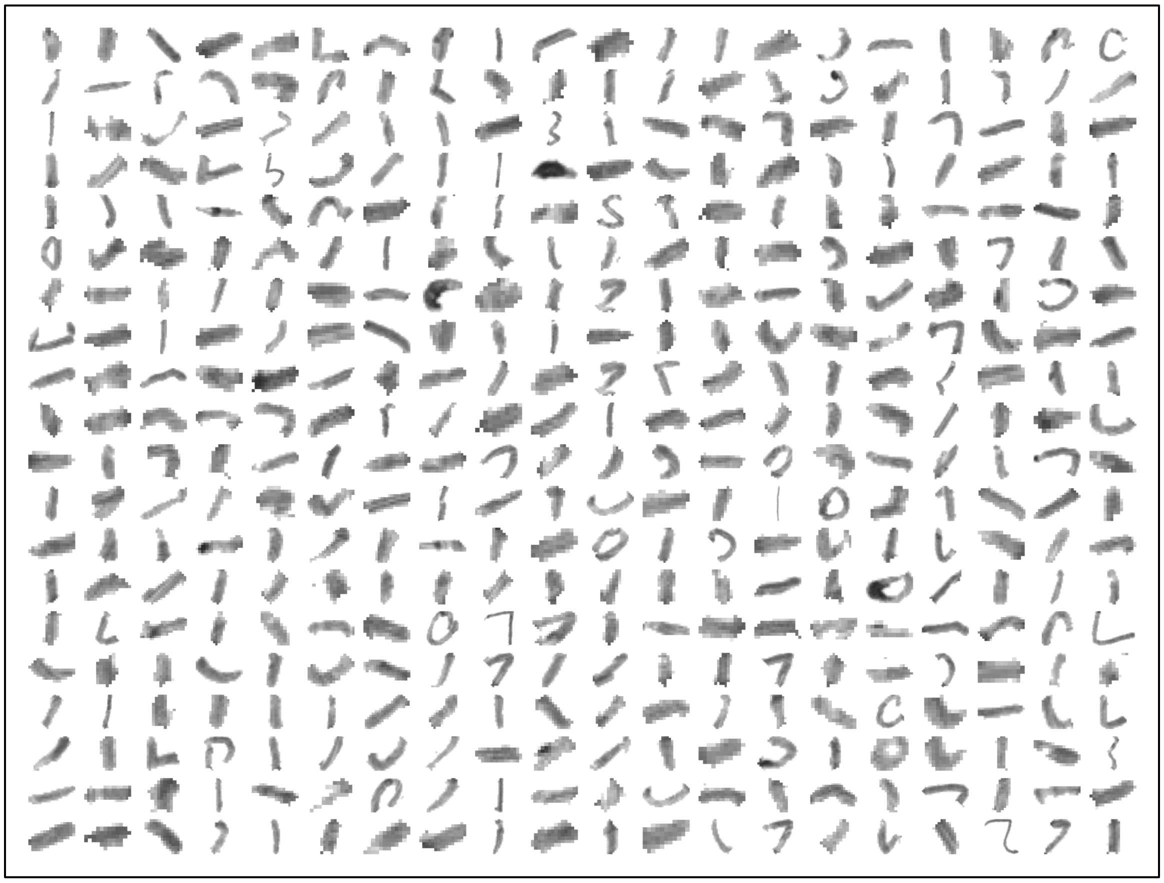

Figure 10. Random Sample of 400 sub-images extracted from Writer6464.

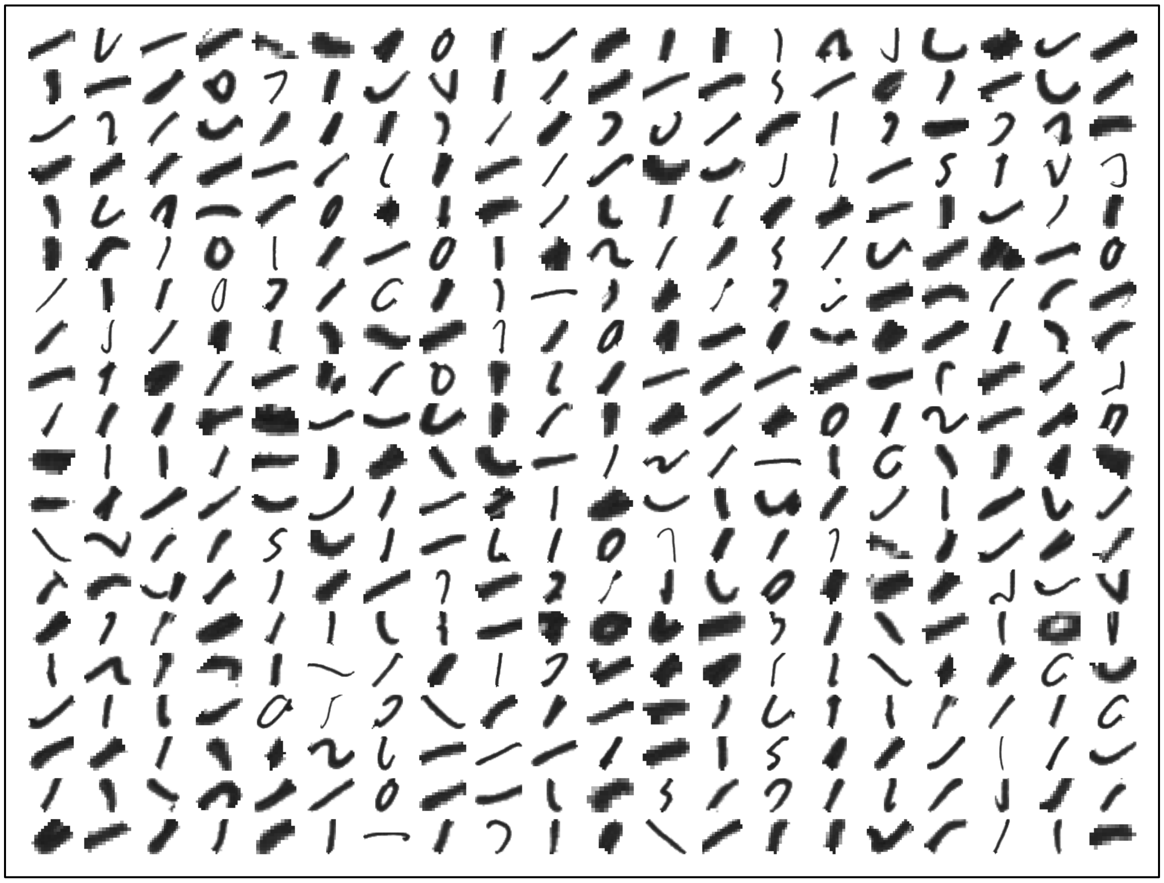

Figure 11. Random Sample of 400 sub-images extracted from Writer1616

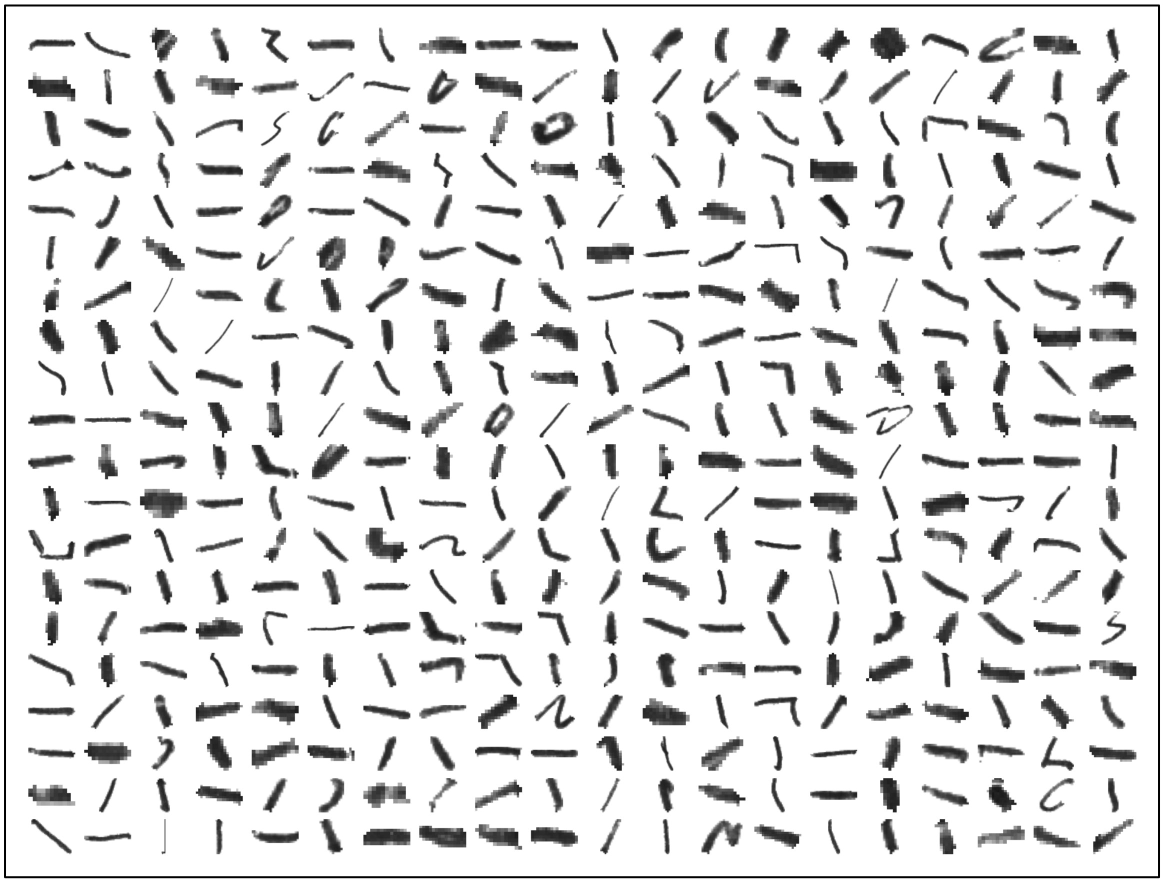

Figure 12. Random Sample of 400 sub-images extracted from Writer9101

CHAPTER V

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 sub-parts 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 simplifying character shapes have potential for being used in Handwriting Identification[4].

5.1 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

[4] The Experimental Results, Publication, Code, and Figures can be found at https://github.com/justjude97/MultiScript-Handwriting-Identification-with-Stroke-Decomposition

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 to be used as sequence data instead of spatial data.

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] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, Art. no. 2, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] T. Y. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984, doi: 10.1145/357994.358023.

[27] 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.

[28] T. Dozat, “Incorporating Nesterov Momentum Into Adam,”, *International Conference on Learning Representations Workshop Track.* 2016.

APPENDIX

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

Table A1 cont.

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

Table A1 cont.

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

Table A1 cont.

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

Table A1 cont.

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

Table A1 cont.

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

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