LEARNING TO READ BUSHMAN: AUTOMATIC HANDWRITING RECOGNITION FOR BUSHMAN TEXTS

KYLE WILLIAMS

SUPERVISED BY:
PROF. HUSSEIN SULEMAN

DISSERTATION PRESENTED FOR THE DEGREE OF MASTER OF SCIENCE IN THE DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF CAPE TOWN

AUGUST 2012

Plaigirism Declaration

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.

.....
Kyle Williams

August 2012

Abstract

The Bleek and Lloyd Collection contains notebooks that document the tradition, language and culture of the Bushman people who lived in South Africa in the late 19th century. Transcriptions of these notebooks would allow for the provision of services such as text-based search and text-to-speech. However, these notebooks are currently only available in the form of digital scans and the manual creation of transcriptions is a costly and time-consuming process. Thus, automatic methods could serve as an alternative approach to creating transcriptions of the text in the notebooks.

In order to evaluate the use of automatic methods, a corpus of Bushman texts and their associated transcriptions was created. The creation of this corpus involved: the development of a custom method for encoding the Bushman script, which contains complex diacritics; the creation of a tool for creating and transcribing the texts in the notebooks; and the running of a series of workshops in which the tool was used to create the corpus.

The corpus was used to evaluate the use of various techniques for automatically transcribing the texts in the corpus in order to determine which approaches were best suited to the complex Bushman script. These techniques included the use of Support Vector Machines, Artificial Neural Networks and Hidden Markov Models as machine learning algorithms, which were coupled with different descriptive features. The effect of the texts used for training the machine learning algorithms was also investigated as well as the use of a statistical language model.

It was found that, for Bushman word recognition, the use of a Support Vector Machine with Histograms of Oriented Gradient features resulted in the best performance and, for Bushman text line recognition, Marti & Bunke features resulted in the best performance when used with Hidden Markov Models. The automatic transcription of the Bushman texts proved to be difficult and the performance of the different recognition systems was largely affected by the complexities of the Bushman script. It was also found that, besides having an influence on determining which techniques may be the most appropriate for automatic handwriting recognition, the texts used in a automatic handwriting recognition system also play a large role in determining whether or not automatic recognition should be attempted at all.

Keywords

Transcription, handwriting recognition, pattern recognition, optical character recognition, machine learning, cultural heritage preservation.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Hussein Suleman, for his guidance, advice and support during the time that I have known him. Not only has Hussein guided me through my honours and masters projects over the last three years, but he has also provided me with opportunities that never would have arisen without his support. Hussein, thank you for everything over these last few years; for supervising and guiding my studies, for always being available to chat, for providing me with invaluable opportunities and for guiding my academic development. I will do my best to put what I've learned from you into practice as I continue to venture into the academic world.

I would like to thank everyone who's minds I have picked over the last few years. There are too many to mention, but your input was invaluable in helping me work through difficulties that I faced during the course of this study. Of these people, I would like to say a special thank you to Julian Kenwood who helped me work through technically difficult problems, not only in this study but also in other avenues, and Dr. Tony Robinson who's careful eye helped identify shortcomings in drafts of this thesis.

I would like to thank my friends and colleagues in the Digital Libraries Laboratory and Computer Science masters lab. I'm going to miss the fun and easy work environment, the constant joking and laughing, the all-nighters sleeping on the couch and the never ending discussions over how good the coffee is.

I would also like to acknowledge the support of this research by The University of Cape Town, The Center for Curating the Archive, The Telkom/NSN/Telesciences/THRIP Centre of Excellence and The National Research Foundation, without which this study would not be possible.

Finally, I would like to thank my family who have supported me immensely during the final stages of this study. To my parents, Deborah and Vincent, and my brothers and sister, Nicholas, Rachel and Justin, it was with your support that I was able to complete this study and achieve what I have achieved.

To my family...

“Ȧkḙn ǂĕn̄nă, tĭ ē, ṅ̆ !kā̃ sshŏ a͡u !χ^e óë̆-sshŏ-!kụi̊, ssĕ !χ^e ụŏṅnĭyă kkĕ́, ṅ̆ ssĕ !kúï̈tḙn ṅ̆-kă !χ^e óë. Ṅ̆ ssĕ ttum̄m-ă !kĕ́-tă-kŭ, kă kkŏ-kkŏḿmĭ, a͡u kā |nĕ !haúë̆ hĭ̃; ṅ̆ ssĕ ttum̄m-ă hĭ̃-kă kkŏ-kkŏḿmĭ, tĭ ē, hĭ̃ |kụē̃-ddă; hĭṅ ttum̄m-ĭ̃ !χ^e óë̆-tă tĭ-kkō-kă Sswā-kā-!kĕ́-kă kkŏ-kkŏḿmĭ, ĭ̃. Hé ē, hĭ̃ |kụē̃-ddă hĭṅ ttum̄m-ĭ̃, ĭ̃; a͡u !χ^e óë̆-sshŏ-!kụĭ-kkō, wā-g |nĕ |kắrră-kaă, ṅ̆ ssĕ |kắrră ssĭń, ṅ̆ ssĭṅ sshŏ kkŏ ttum̄m-ă, kkŏ-kkŏḿmĭ ē kkăń, ssĕ ttăń, hé ĕ, kkŏ-kkŏḿmĭ ē |hĭń ǁkhwĕ́-tḙn. Hé ē, ń ssĕ |nĭ kkŏm̄m, ĭ (aú hĭ-hĭ); aú kă tăttĭ ē, hĭ̃ !gwētḙn |hĭṅ ǁkhwĕ́-tḙn; a͡u !χ^e óë̆-sshŏ-!kụĭ yā-g |nĕ ttă !kă̦ !kă̦!kā̦itḙn; a͡u kă tăttĭ ē, ṅ̆ ddó̠ă̈ |nĕ ǁkóă̈-kḙn !ha̠úë̆ ; ṅ̆ ssĭń |nĕ ddó̠ă̈ ǂkắkkḙnǂkắkkḙn hĭ̃, ṅ |kāgḙn ttúkḙn.”

“Thou knowest that I sit waiting for the moon to turn back for me, that I may return to my place. The I may listen to all the people’s stories, when I visit them; that I may listen to their stories, that which they tell; they listen to the Flat Bushmen’s stories from the other side of the place. They are those which they thus tell, they are listening to them; while the other !χ^e óë̆-sshŏ-!kụĭ (the sun) becomes a little warm, that I may sit in the sun; that I may sitting, listening to the stories which yonder come, which are stories which come from the distance. Then, I shall get hold of a story from them, because they (the stories) float out from a distance; while the sun feels a little warm; while I feel that I must altogether visit; that I may be talking with them, my fellow men.”

- || Kabbo’s Intended Return Home, Specimens of Bushmen Folklore (1911)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Historical texts serve as a record of the past, detailing events, customs and beliefs that have since disappeared. These records can serve as an aid in understanding our history and informing our future. However, many historical texts are at risk due to natural forces such as physical degradation, resulting in a worldwide effort to preserve them.

One of the forms of preservation is digital preservation, in which digital/scanned versions of historical texts are created, catalogued and stored in digital libraries. These digital documents can then be made accessible via the Web where digital library services facilitate access and interaction. The forms of interaction commonly involve search and browse, which are based on the metadata associated with the digital documents. However, the text within digital documents provides an additional source of information that can be exploited to provide enhanced ways of interacting with the documents and transcriptions of the text allow for additional services such as in-text search and text-to-speech.

The Bleek and Lloyd Collection, which documents the language, culture and beliefs of the Bushman people who lived in South Africa in the late 19th century, is an example of a collection of historical documents that has been digitised and preserved in a digital library. This collection, a UNESCO Memory of the World Collection, is available online as a set of digital scans that can be browsed and searched based on the metadata associated with the collection (Suleman, 2007). As with other historical texts that have been digitised, transcriptions of these texts would allow for the provision of enhanced services. For instance, transcriptions of these texts would: enable the text in the notebooks to be indexed and thus searched and compared to other texts; enable the development of text-to-speech applications, which could be used for digital story telling; allow for linguistic information about the Bushman languages to be compiled using automatic techniques; allow for the text in the notebooks to be re-purposed, for instance to be reprinted in books; and provide enhanced access to a source of indigenous knowledge.

Image :

Transcription : !gaủ ë̆ #kua̲̍n ḳaŭ̍kĭ

Figure 1.1: A Bushman text line as an image and its associated transcription

Figure 1.1 shows an example of a line of text from one of the Bleek and Lloyd notebooks alongside its associated transcription. As can be seen from the figure, the text transcription enables all of the above-mentioned benefits, which cannot be accomplished with the digital image alone. Thus, transcriptions are clearly beneficial. However, manual transcription is an expensive, tedious and time consuming task. Automatically transcribing documents in a collection is one way of overcoming these difficulties.

1.1 Automatic Handwriting Recognition

"Automatic handwriting recognition" refers to the process by which a computer system is able to identify and classify previously unseen handwritten text and, as a result, is able to automatically transcribe handwritten documents. A common approach to automatic handwriting recognition is to make use of machine learning algorithms that are capable of learning how to recognise the text in previously unseen documents by being trained with document/transcription pairs and learning the associations between them. An overview of this process is shown in Figure 1.2 where the associations between digital documents and their transcriptions are learned during the training phase and previously unseen data is recognised during the recognition phase.

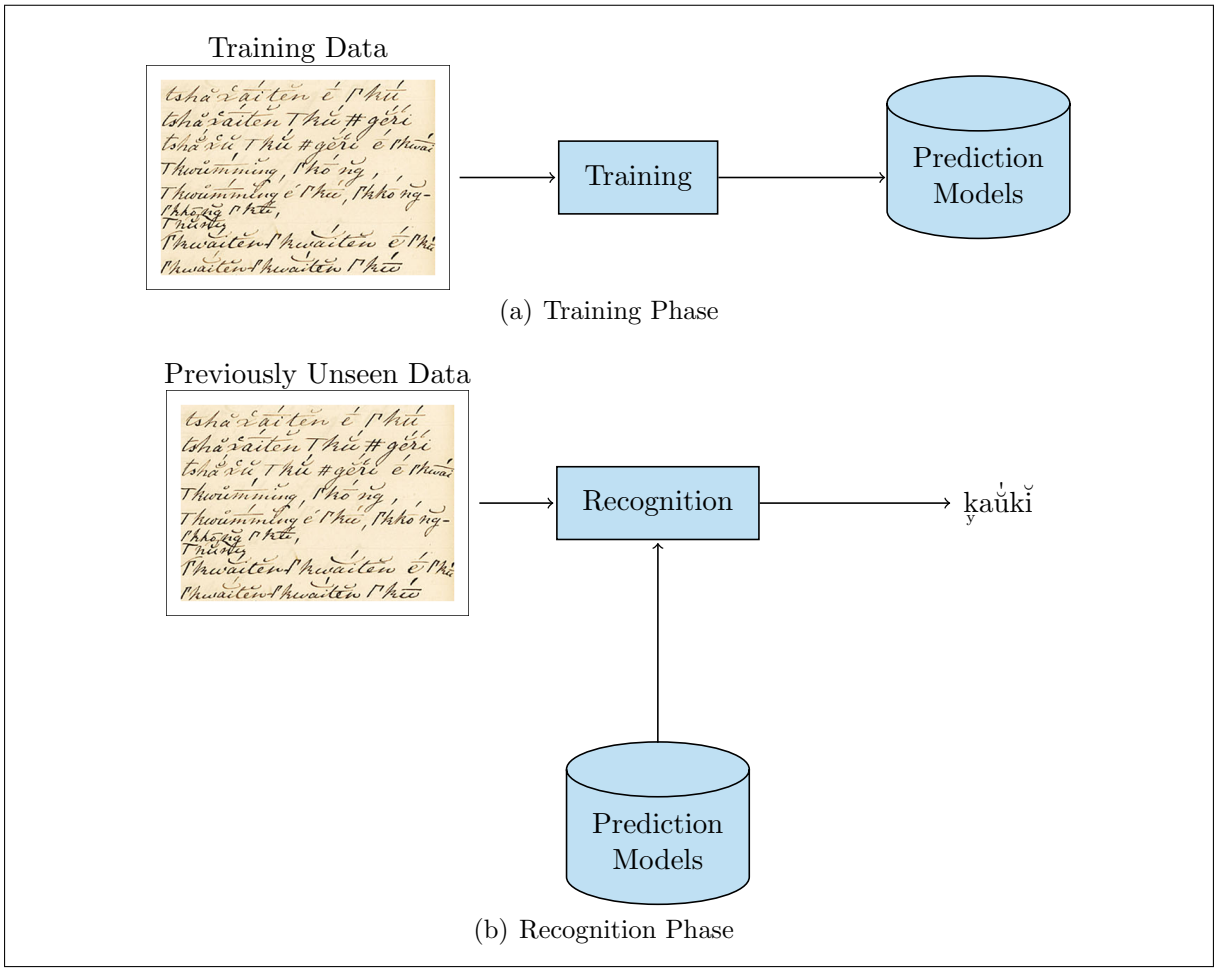


Figure 1.2: The automatic handwriting recognition process consisting of a training phase and a recognition phase

There are a number of elements that make up an automatic handwriting recognition system, such as:

Machine learning algorithms that learn the associations between digital documents and their transcriptions in order to form prediction models (Rätsch, 2008).

Descriptive features, which are compact descriptors of an object that characterise it and make it distinguishable from others (Pratt, 2007). In automatic handwriting recognition, descriptive features are extracted from document images and then, along with their corresponding transcriptions, are used to train the machine learning algorithms and also used as input for the prediction models during the recognition phase.

Segmentation levels, which are the units into which a page is divided in order for additional processing to take place, i.e. character, word or text line.

A corpus, which is used as a testbed and is made up of the document/transcription pairs that are used for training the machine learning algorithms and for recognition.

Statistical language models that contain statistical information about the distribution of characters and words in a natural language and can be used to constrain output during recognition (Marti, 2000).

It is desirable to have handwriting recognition systems that are accurate and robust to variation within the data. However, the realisation of these goals largely depends on the techniques used for automatic handwriting recognition, for instance, how machine learning algorithms and descriptive features are combined and the extent to which a statistical language model is used. In this study, a series of techniques for handwriting recognition are investigated in order to evaluate their applicability to the Bushman texts that appear in the Bleek and Lloyd Collection.

1.2 Motivation

There are a wide variety of techniques that can be used when designing a handwriting recognition system (Bunke, 2003) and the use of specific techniques could have a significant effect on recognition accuracy. Thus, the choice of which techniques are used is an important decision, particularly the choice of machine learning algorithms and descriptive features as these form the core of a recognition system. However, the choice of which techniques are most applicable may not always be obvious. For instance, it is difficult to make an informed decision as to which techniques may be best based on the findings of studies where different handwriting recognition techniques have been applied to different collections and where the experimental methodologies may have differed.

Thus, given the wide variety of techniques that are available when designing a handwriting recognition system, the main goal of this study is to investigate which techniques lead to the highest recognition accuracies when used for the automatic recognition of the Bushman texts that appear in the Bleek and Lloyd Collection. High recognition accuracies are desirable since they enable the successful provision of enhanced digital library services - something that is difficult to do with poor transcription accuracies - and reduce the amount of manual intervention that is required.

Since all of the techniques investigated in this study are applied to a single collection of handwritten documents, a direct comparison can be made between them and used to inform other studies - something that is difficult with independent studies. Furthermore, while this study specifically focuses on the Bushman texts, the findings as to which techniques are best may also be applicable to other collections of handwritten texts, especially those that contain complex diacritics.

1.3 Problem Statement

To investigate the use of Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Hidden Markov Models (HMMs) in conjunction with various descriptive features for Bushman word and text line recognition and to investigate how training data and statistical language models can affect recognition accuracy.

1.4 Scope and Limitations

The focus will be on recognising texts written by two authors in the |xam language, which is one of the Bushman languages that appears in the Bleek and Lloyd Collection, using different techniques for handwriting recognition. It would be infeasible to investigate the use of all possible techniques, thus a selection of techniques will be investigated in order to provide an overview of the performance of different approaches.

1.5 Research Questions

The research questions investigated as part of this study are:

i. *Which of a selection of Hidden Markov Models, Artificial Neural Networks and Support Vector Machines, when used in conjunction with various descriptive features, performs best when automatically transcribing handwritten Bushman texts?*

   Machine learning algorithms are used for the classification and recognition of words and characters. Previously, machine learning algorithms such as Hidden Markov Models (Rabiner, 1989), Artificial Neural Networks (Haykin, 1998) and Support Vector Machines (Cortes and Vapnik, 1995) have been used to recognise handwritten texts. The machine learning algorithms use descriptive features in both the training and recognition phases and it is natural to expect that some descriptive features perform better than others. This research question seeks to investigate which of the above machine learning algorithms, when used in conjunction with different descriptive features, performs best when automatically transcribing handwritten Bushman texts.

ii. *What is the effect of the training data on transcription accuracy?*

   The accuracy achieved when performing automatic transcription is, naturally, dependent on the data used for training. For instance, accuracy can be affected by the

amount of training data that is available and the variation within that data. This research question seeks to investigate the effect that training data has on performance when automatically transcribing handwritten Bushman texts.

iii. *To what extent can a N-gram language model improve accuracy when automatically transcribing handwritten Bushman texts?*

It may be possible to build a N-gram language model for the Bushman languages. If the language model is built, it may be possible to use it to improve recognition accuracy. This research question seeks to investigate the extent to which this can be done.

The contribution of this research is insight into the best approach to automatically transcribing handwritten Bushman texts and a comparison of the use of different techniques. It is hoped that the findings are also applicable to other handwritten texts and can be used to inform future studies, especially where the language contains complex diacritics.

This research differs from other research in that it is the first attempt at the automatic transcription of handwritten Bushman texts.

1.6 Overview of this Thesis

The rest of this thesis is structured as follows. Chapter 2 reviews the literature and discusses the background to this study. This includes an introduction to the Bushman texts, a review of techniques that have previously been used in handwriting recognition studies and a discussion of some of the theoretical aspects related to handwriting recognition. Chapter 3 describes two pilot studies that were conducted during the early stages of this study in order to determine whether or not the study was feasible. A testbed was required in order to investigate techniques for Bushman handwriting recognition and Chapter 4 discusses the creation of a corpus that was used as a testbed in this study, including a discussion of a custom technique that was developed to encode and represent the Bushman texts and the development of a tool that was used for transcribing the Bushman texts. Features play a large role in this study and thus are discussed in Chapter 5. This includes a discussion of the derivation of the features, their invariant properties and how they are used in this study. The experimental design used in this study is presented in Chapter 6, while the actual experiments are described in Chapter 7 including the presentation and analysis of their results. Lastly, conclusions and possibilities for future work are discussed in Chapter 8.

Chapter 2

Background

The previous chapter introduced the purpose and goals of this study and this chapter puts the study into perspective by discussing the background to the study and the context in which the research takes place. This chapter begins by briefly motivating the importance and need for cultural heritage preservation. This study is based on the automatic recognition of Bushman texts, and therefore the next section discusses the Bleek and Lloyd Collection and corpora for handwriting recognition. An overview of handwriting recognition is then given, followed by a description of each of the components that make up a handwriting recognition system and their use in the literature.

2.1 Cultural Heritage Preservation

In 1972 the UNESCO World Heritage Convention identified the need for identifying, conserving, protecting, presenting and ensuring transmission to future generations of cultural heritage and natural heritage materials (Unesco, 1972). Digital preservation strategies support these requirements through the creation of mechanisms for preservation, presentation, access and future accessibility. Digital preservation allows for artefacts to be accessed regardless of their physical location and, given the often fragile nature of the artefacts, removes the risk of damage due to physical interaction with the artefacts. Furthermore, digital preservation can even assist in the restoration of artefacts that have become damaged (eHeritage, 2010).

In this section, two aspects related to cultural heritage preservation are discussed. The first of these is the Bleek and Lloyd Collection (Suleman, 2007), a collection of notebooks, dictionaries and artwork that document the language and culture of the Bushman people who lived in South Africa in the late 19th century. Thereafter the creation of corpora, which are collections of texts and manuscripts, and their use in handwriting recognition are discussed.

2.1.1 The Bleek and Lloyd Collection

The Bushman people are South Africa's oldest human inhabitants (Lee and Balick, 2007) and it is likely that they have a unique view of the world. However, many of the Bushman languages, such as the |xam Bushman language, have completely died out (Suleman,

2007). As with many ancient cultures, Bushman culture and beliefs were metaphorically encoded and shared in the form of stories. These stories were written down in notebooks by German linguists, Wilhelm Bleek and Lucy Lloyd in the 1870s and, together with art and dictionaries, have come to be known as the Bleek and Lloyd Collection (Suleman, 2007). In its entirety, the Bleek and Lloyd Collection is a collection of notebooks, art and dictionaries, which document the languages and culture of the Bushman people of southern Africa and, more specifically, the languages and culture of the |xam and !kun Bushman people. The notebooks in the collection contain metaphorical Bushman stories and, in most cases, their corresponding English translations appear alongside them. The drawings in the collection complement the stories in the notebooks, while the dictionaries contain English words and their corresponding Bushman translations. Figures 2.1, 2.2 and 2.3 show examples of a notebook page, dictionary entry and artwork from the Bleek and Lloyd Collection, respectively.

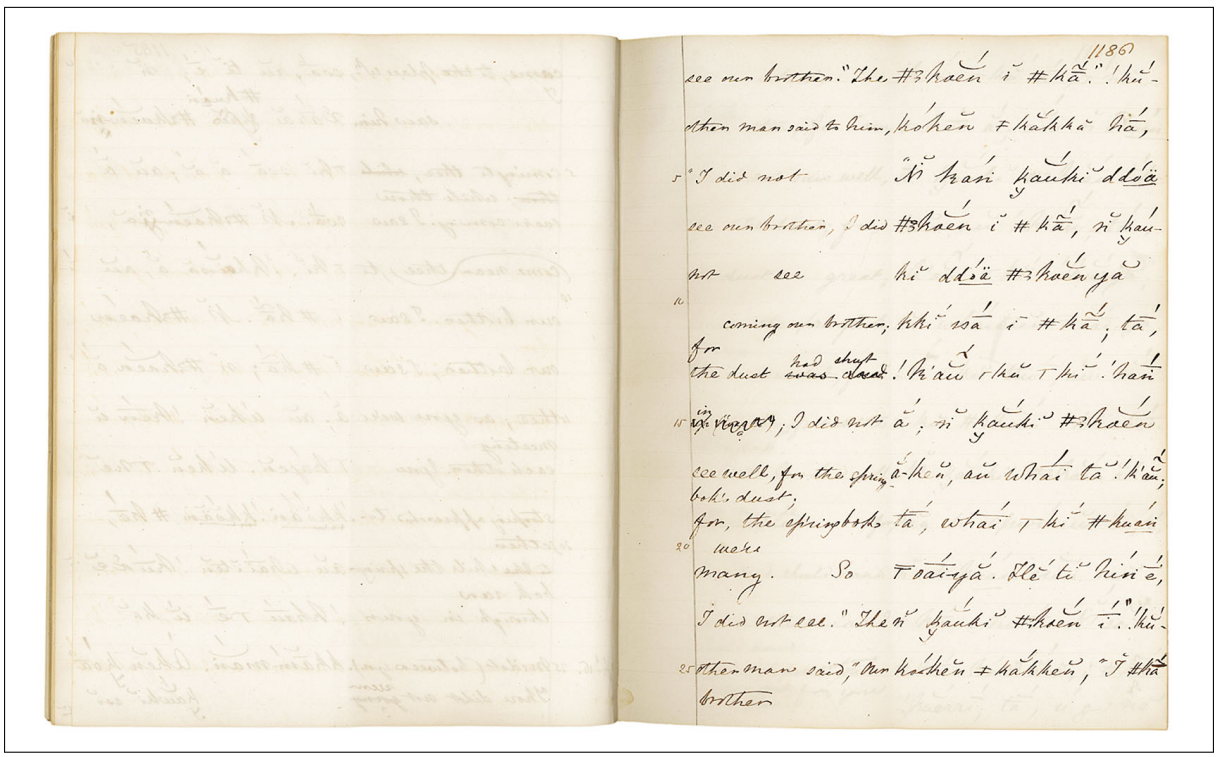


Figure 2.1: A notebook page from the Bleek and Lloyd Collection

The Bleek and Lloyd Collection is jointly owned by the University of Cape Town, The National Library of South Africa and the Iziko National Museum of South Africa and is made up of 157 notebooks containing 14128 pages, 752 drawings and over 14000 dictionary entries (Suleman, 2007). In 2003 the Lucy Lloyd Archive and Research Centre at the Michaelis School of Fine Arts undertook to digitally preserve the collection and in 1997 the collection was recognised by UNESCO as a Memory of the World Collection (Suleman, 2007), illustrating the importance of the collection.

2.1.1.1 Bushman Diacritics

The script used to represent the Bushman text is complex due to the diacritics that appear above characters, below characters and both above and below characters. Diacritics can

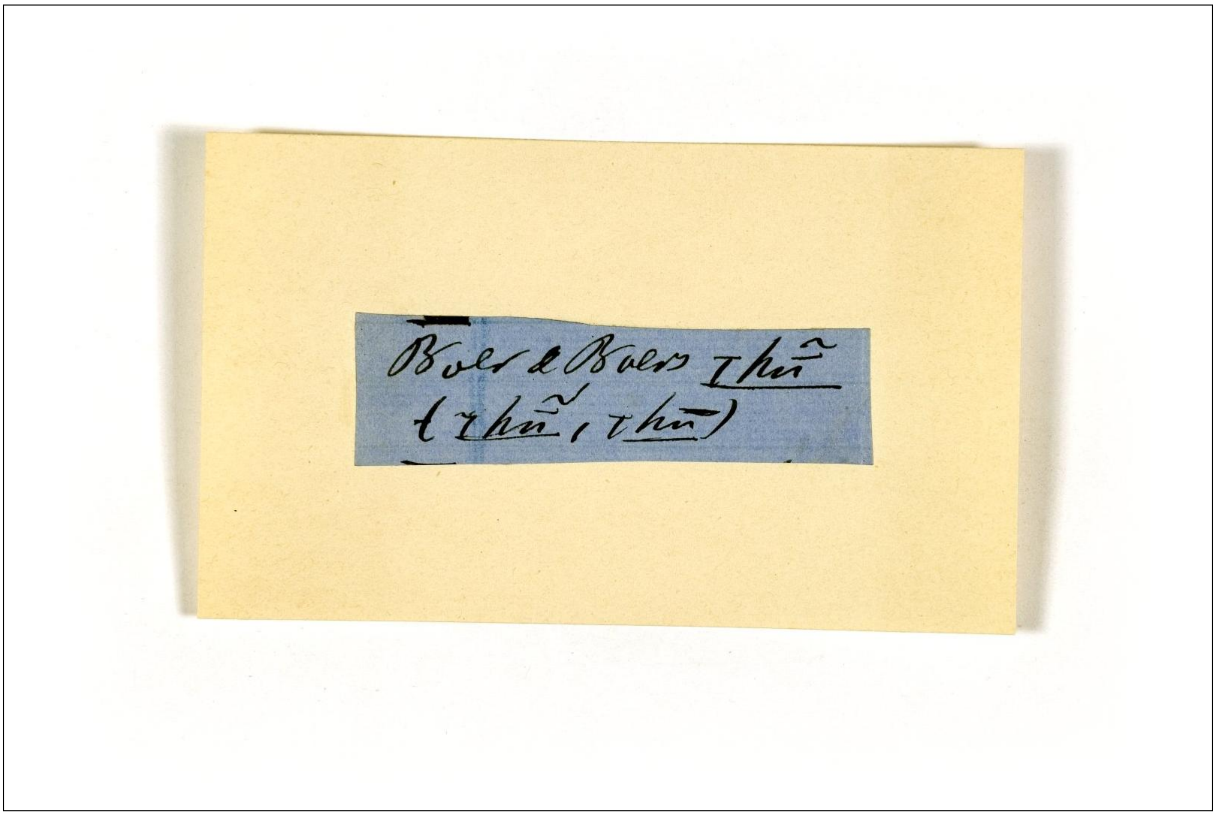
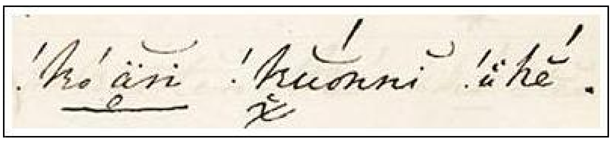


Figure 2.2: A dictionary entry from the Bleek and Lloyd Collection

also span multiple characters. The result is that the text cannot be represented using standard or non-standard Unicode. Thus far, more than 64 different combinations of single and stacked diacritics have been discovered[1] that appear above characters, 13 that appear below characters and 60 that appear both above and below characters. Table 2.1 shows some of the types of diacritics that appear in the texts, starting with simple diacritics that appear above characters, then diacritics that appear above and below characters and then diacritics that are stacked and that span multiple characters.

Table 2.1: Images of Bushman diacritics from the notebooks and their transcriptions

Image	Transcription
	!kǔ̍ ï̆ yȧkeň !kui̍̆-yǎ̍ .
	!gaủ ĕ̈ #kuạ̍n ḵaǔ̍kĭ̆
	!kȯä̆n̆ ḵ ŭ̍onnĭ̆ !ŭ̍hĕ̍ .

[1]New diacritics are constantly being discovered as the text is explored.

Figure 2.3: A piece of artwork from the Bleek and Lloyd Collection

The Bleek and Lloyd Collection presents a case study for digital preservation. The collection has gone through the various stages of digital preservation, including its identification, which resulted in the creation of metadata, its preservation in digital library systems (Suleman, 2007; Williams et al., 2010) and ensuring transmission and access through publications (Skotnes, 2007) and publicly accessible websites[2]. Furthermore, different mechanisms have been developed to enable interaction with the collection (Williams and Suleman, 2010). This study seeks to further the ways in which the Bleek and Lloyd Collection can be accessed and interacted with by providing transcriptions of the Bushman texts that appear in the notebooks.

2.1.2 Handwriting Recognition Corpora

A corpus is a collection of texts or manuscripts, such as the notebooks in the Bleek and Lloyd Collection (Suleman, 2007) or the George Washington Manuscripts (Rath and Manmatha, 2007), which is a collection of manuscripts that were written by George Washington and copied and edited by his secretaries and that have been digitised by the Library of Congress (Kane et al., 2001). Transcriptions of the texts in a corpus allow for additional digital library services to be provided, such as:

- The creation of an index of the manuscripts.

[2]http://lloydbleekcollection.cs.uct.ac.za/

- Facilities to find similar passages in the text.
- Re-rendering and re-printing of the texts, which could be useful for hard to read historical texts.
- The ability to search the text based on keywords.

Transcriptions of corpora also play a role in the handwriting recognition process, in which they are used to train recognition engines in order to identify unseen data. There are a number of handwriting recognition corpora that have been used for handwriting recognition. For instance, the ETL9B database is a database of handwritten Japanese characters that contains 200 samples of 3036 character classes (Liu et al., 2010). This database has been used in a number of character recognition experiments and recognition accuracy of over 99% has been achieved (Kimura et al., 1997). The IAM database (Marti and Bunke, 1999) is a database of handwritten English and contains over 40 000 handwritten words, which come from almost 5000 text lines, written by approximately 250 different writers. The text in the IAM database is based on the Lancaster-Oslo/Bergen corpus (Johansson et al., 1978). The IAM database has been used in a number of handwriting recognition studies (Graves et al., 2009; Espana-Boquera et al., 2011; Vinciarelli et al., 2004). A version of the IAM database for historical handwriting also exists (Fischer et al., 2010). The CEDAR database (Huang and Srihari, 2008) contains handwritten city names, state names, zip codes and alphanumeric characters, which were scanned and collected at a post office, while a database of handwritten Arabic words, written by 100 writers, also exists (Al-Ma'adeed et al., 2002).

2.1.2.1 Corpus Creation

The existence of a high quality corpus is necessary for any system that performs automatic recognition of text, whether handwritten or machine printed. Corpora for English or other well-understood and well-studied languages are relatively easy to access and make use of. Examples of widely-used corpora are the IAM database (Marti and Bunke, 1999) and the George Washington manuscripts (Rath and Manmatha, 2007). However, for less well-studied languages and scripts, it is often necessary to create a new corpus that can be used for experimentation since it is unlikely that a suitable one already exists.

Creating corpora for modern languages is relatively easy compared to creating corpora for historical languages. Usually, creating corpora for modern languages involves getting a group of users to write words on forms that were specifically designed for creating the corpora. For instance, M. Agrawal and Madhvanath (2004) used HP Tablet PCs to collect data and create a corpus for complex Indic scripts and Al-Ma'adeed et al. (2002) used paper forms on which specific Arabic words were written and then scanned to create an Arabic corpus.

The creation of corpora for historical texts is, however, more difficult than that of modern texts. For instance, it is not possible to make use of forms to easily and accurately capture specific data. Historical texts also introduce a number of difficult problems related to the segmentation of lines and words due to poor handwriting and the effects of age, such as ink-bleed and paper degradation. Fischer et al. (2010) note that, in many cases, transcriptions of historical texts can only be performed by language specialists, whereas for modern texts it can be performed by lay-persons. This is of course open to debate

since it is possible to train lay-persons in transcribing the text or, alternatively, use a crowd-sourcing approach, which has, in many cases, been shown to be more accurate than specialists (Surowiecki, 2005).

Fischer et al. (2010) describe a tool for the creation of a corpus for the IAM Historical Handwriting Database (IAM-HistDB). The specific manuscripts that they make use of all appear to have been scribe-written and are presented in a neat fashion. The tool described by Fischer et al. contains algorithms that automatically segment text on a page and that allows for users to manually correct any errors. The transcription of the text is not performed as transcriptions for the text already exist. Instead, transcription alignment is automatically performed and users are able to perform alignment correction.

Setlur et al. (2003) describe the complexities of the Devanagari script, which is the basis of many Indian languages and the lack of consistency among researchers in terms of the representation of the script, specifically in terms of what the granularity of a character should be. They describe a Java-based tool for creating a corpus for Devanagari script recognition, which automatically segments words and lines and also allows for user interaction. Once lines and words have been segmented, the Devanagari script is then captured using a simulated keyboard or by typing transliterated text on the keyboard. The text is stored in Unicode and the Devanagari output can be previewed by rendering the Unicode using a Devanagari font.

This study requires the existence of a corpus for Bushman languages and the creation of one is described in Chapter 4. However, before describing the creation of the Bushman corpus, the next section provides an overview of automatic handwriting recognition, followed by a more detailed description of each of its components in the sections thereafter.

2.2 Automatic Handwriting Recognition

Automatic handwriting recognition, driven by the goal to equal or surpass human recognition, as well as its commercial applications, has been studied for over 40 years (Bunke, 2003). From a commercial perspective, it has been successfully used for address reading and cheque processing, both of which have highly constrained or small vocabularies. However, constraints exist in both of these applications and knowledge of the task is readily available (Bunke, 2003). Automatic handwriting recognition in unconstrained environments for which little task-specific knowledge exists, such as in the case of handwritten historical documents, is a more complex task since additional information often does not exist and therefore cannot be used to improve recognition results. Handwritten text has a number of characteristics that further complicate the segmentation and recognition process. For instance: text lines can be skewed; there can be variation in character formation by a single author as well as variation in character formation and writing style between multiple authors; and there can exist a lack of uniformity in character size, character and word spacing and character slant. Furthermore, for historical documents a number of additional complications are introduced such as the effects of degradation over time and ink bleed, in which the ink from the other side of a page shines through (Manmatha and Srimal, 1999).

Automatic handwriting recognition can be split into two categories: on-line handwriting recognition and off-line handwriting recognition (Fischer et al., 2009). On-line handwriting recognition involves input via an apparatus such as a stylus or mouse and allows for

temporal information to be exploited in the recognition process. Off-line handwriting recognition, on the other hand, involves the input of handwriting captured by a scanner or camera and, as such, lacks temporal information (Bunke, 2003). This research deals with the problem of automatic handwriting recognition of historical documents in the Bleek and Lloyd Collection for which temporal information does not exist.

Automatic handwriting recognition usually involves a training phase and a recognition phase (Figure 2.4) and each of these phases can be broken down into a number of steps. The training phase usually involves the preprocessing of training data using techniques such as binarisation and segmentation, followed by feature extraction and then model learning using a machine learning algorithm. The output of this phase is a prediction model for the learned data. The recognition phase usually involves the same preprocessing of testing data, followed by recognition using the models derived from the training phase. The recognition phase often includes a post-processing stage in which statistical language models are used to improve results (Marti and Bunke, 2002). The output of this phase is the transcribed data.

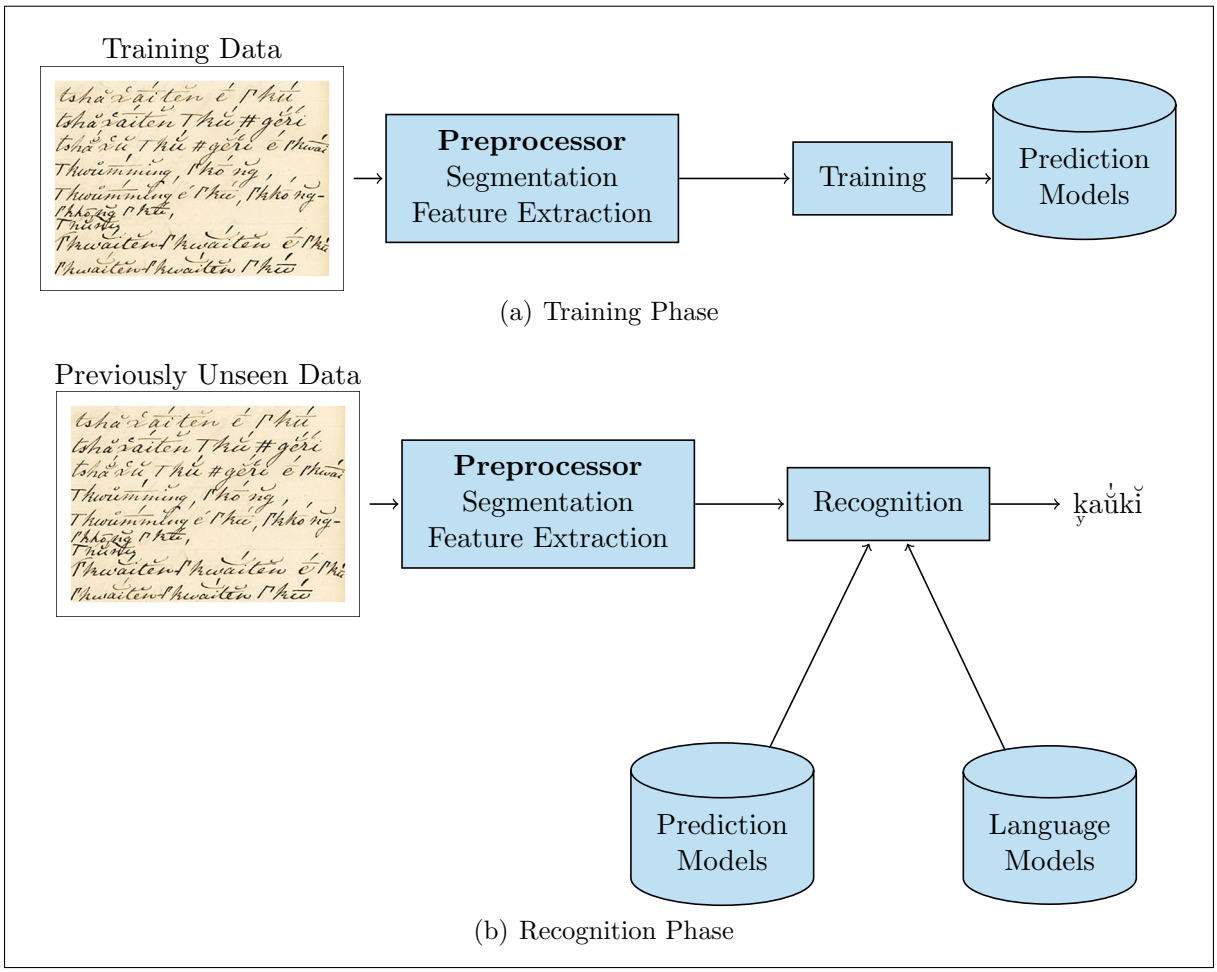


Figure 2.4: The automatic handwriting recognition process consisting of a training phase and a recognition phase

There have been a number of attempts at automatic handwriting recognition using a variety of preprocessing/segmentation approaches, descriptive features, machine learning algorithms and statistical language models. All of the above-mentioned techniques, to some degree, usually play a part in the process of handwriting recognition. In the following

sections, each of the aspects related to handwriting recognition will be discussed and, where appropriate, examples of their use will be drawn from the literature.

2.3 Segmentation

Segmentation is often used during the preprocessing stage in order to separate lines of text, words, characters and other units so that they can be individually recognised. Figure 2.5 shows an example of line, word and character segmentation.

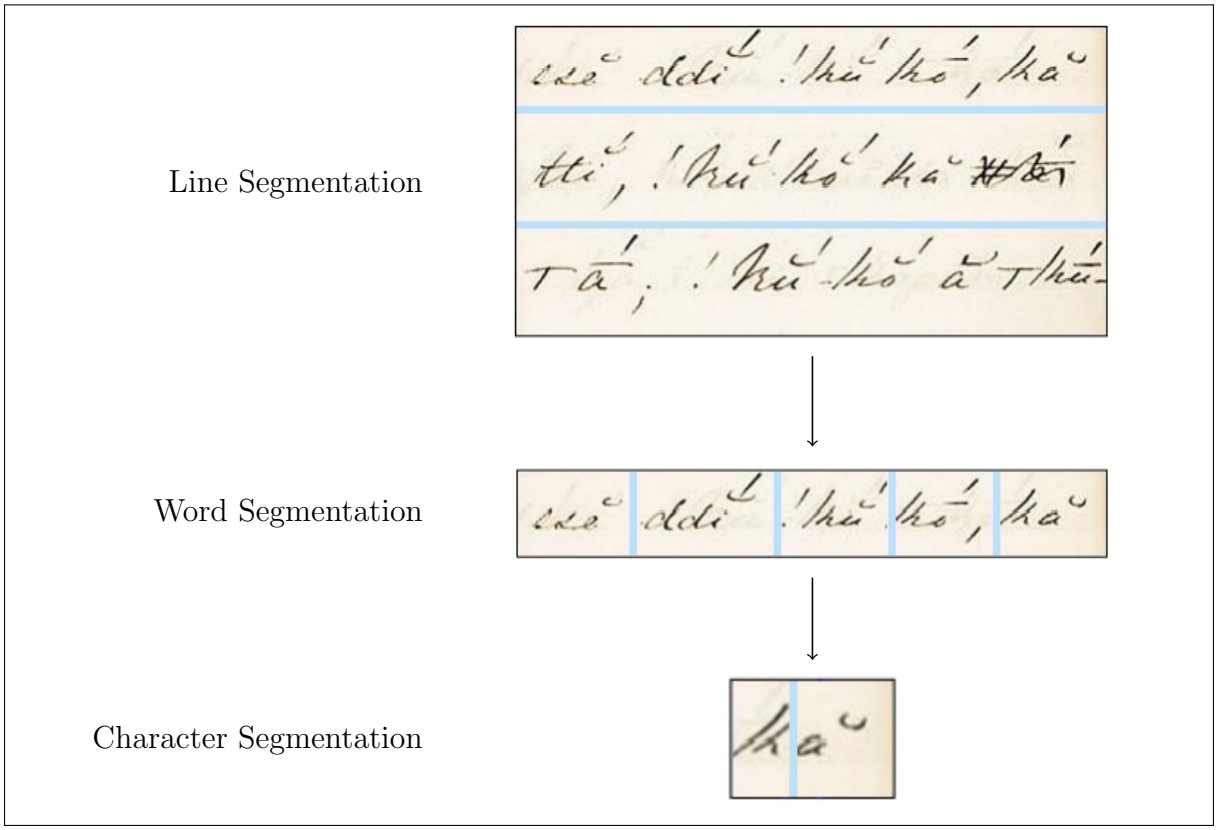


Figure 2.5: Line, word and character segmentation

There are a number of elements that affect the success with which segmentation can be performed. For instance, the spacing of words and characters, the script used, the spacing between lines, whether the writing is cursive or hand-printed, noise in the image and the other issues that affect handwritten documents and that were discussed in the previous section.

Therefore, it is naturally easier to segment certain types of text. For instance, it is easier to segment clearly separated lines and words rather than ones that overlap and, in a similar sense, it is easier to segment block-print text than cursive script.

In this section, some of the different approaches to segmentation will be discussed and, where appropriate, their use in handwriting recognition studies will be discussed. The discussion takes place in a top-down fashion, beginning with a discussion of line segmentation and then going onto word segmentation, followed by character segmentation and, lastly, segmentation at lower levels.

2.3.1 Line Segmentation

Line segmentation involves segmenting images of text into lines. The segmented lines are often further segmented into words and characters, but are also fed directly into handwriting recognition systems such as those based on Hidden Markov Models (HMMs) and Recurrent Neural Networks (RNNs) and which allow for word and character segmentation to be avoided. One of the difficulties in line segmentation relates to variation of skew both within and between different text lines (Aradhya and Naveena, 2011). In cases where the author took care to write along a ruled margin, as is the case with the Bleek and Lloyd Collection (Suleman, 2007), skew is not necessarily a problem. However, in other cases it is and can be corrected as a preprocessing step (Marti and Bunke, 2002). Alternatively, segmentation algorithms can explicitly account for skew in text lines.

Another consideration when doing handwritten line segmentation involves touching, which occurs when two components from different lines touch and overlap, which occurs when components from one line extend into another line (Likforman-Sulem et al., 2007). Figure 2.6 shows overlapping and touching components.

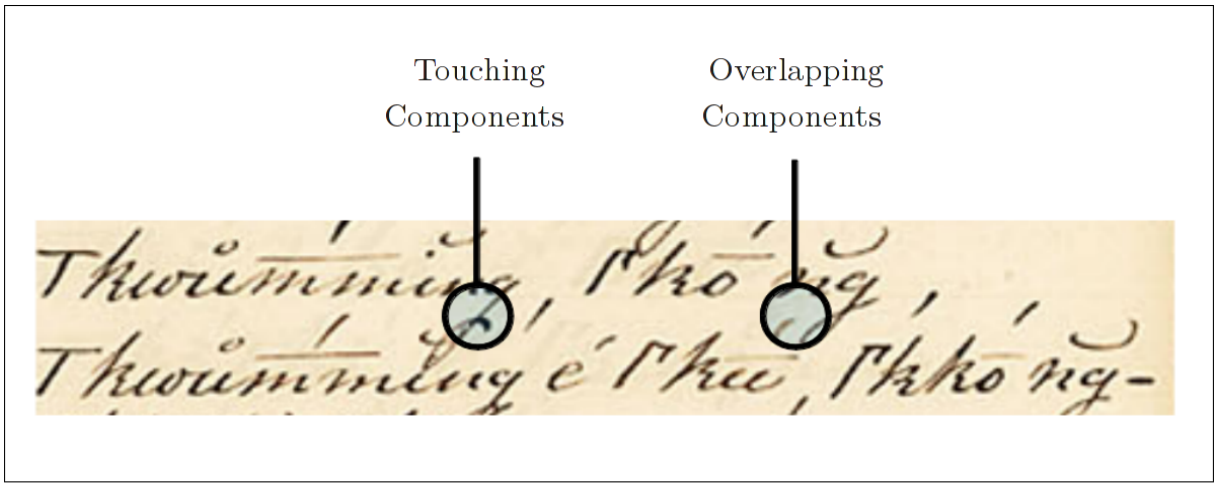


Figure 2.6: Overlapping and touching components in a text line

Existing techniques for line segmentation include projection profile based techniques, where the distribution of pixels along the horizontal axis is used to identify the spaces between lines. For instance, a projection profile technique was used to segment lines of handwritten documents in the Gurmukhi script (Kumar et al., 2010b). A projection profile technique was also used by Marti and Bunke (2001) in investigating the effect that vocabulary size and language models have on unconstrained handwriting recognition. Another technique for line segmentation involves smearing the image to create blobs in the foreground (Aradhya and Naveena, 2011), for instance, using the The Run Length Smoothing Algorithm (Wong et al., 1982).

Aradhya and Naveena (2011) proposed a line segmentation technique for segmenting texts written in the Kannada script. Their proposed technique involved identifying the Connected Components (CCs) in an image and then growing the bounding boxes of the CCs in a horizontal direction to create single CCs for each text line. Piece-wise projection profiles have also been used where a document is separated into vertical strips and then the projection profiles for each strip are calculated and line segmentation points for each

vertical strip are identified (Pal and Datta, 2003). The piece-wise separation points can then be joined and the text lines segmented (Pal and Datta, 2003).

Kumar et al. (2010a) developed a technique for segmenting handwritten Arabic text lines, which, like the Bushman texts, contain diacritics. Their technique initially removes diacritics, then segments the text lines and reattaches the diacritics to the text lines afterwards. The Hough Transform (Duda and Hart, 1972) has also been used for text line segmentation. For instance, Likforman-Sulem et al. (1995) used the Hough transform to hypothesise segmentation points in the Hough domain and then validated them in the image domain.

2.3.2 Word Segmentation

Word segmentation involves segmenting lines of text into words or, alternatively, segmenting whole pages of text into words without first segmenting them into lines. Segmentation of machine printed text lines into words is relatively simple compared to the case for handwritten documents. This is due to machine printed text having uniform and well-defined spaces between and words. This is different to the case for handwritten documents, where there is often a lack of uniformity and consistency in spacing between words (Seni and Cohen, 1994).

Gap metrics are commonly used for word segmentation, where the spaces between adjacent characters are used to identify the spaces between words. Makridis et al. (2007) made use of a gap metric-based word segmentation algorithm in which all CCs in a text line were identified. The CCs were then sorted by their x-coordinate and adjacent CCs that had a distance greater than some threshold between them were identified as belonging to two different words. Mahadevan and Nagabushnam (1995) used the Euclidean distance between the centers of gravity of the convex hulls of two adjacent CCs as a gap metric. Louloudis et al. (2009) took a slightly different approach to the gap metric calculation in which they calculated the distance between overlapping components rather than CCs, where overlapping components are CCs whose vertical projection profiles overlap.

In studies by Sargur N. Srihari (1997) and Huang and Srihari (2008), Artificial Neural Networks (ANNs) were used to identify word segmentation points. Manmatha and Srimal (1999) segmented words in text lines based on "blobs" in a scale space representation of an image. The Hough transform (Duda and Hart, 1972) has also successfully been used to segment handwritten words (Satadal Saha and Basu, 2010). In a previous study involving the Bleek and Lloyd Collection, handwritten words in the |xam Bushman language were segmented by exploiting the domain knowledge that the Bushman words were underlined (Williams and Suleman, 2010). The lines underlying the Bushman words were identified using CC analysis and projection profiles were used to identify word boundaries.

In handwriting recognition studies, Fischer et al. (2009) used perfectly segmented lines and words in order to transcribe text in handwritten medieval documents that were written in Middle High German. Fischer et al. automatically segmented lines and words; however, any segmentation errors were manually corrected. Projection profiles were used by Vamvakas et al. (2008) to segment words in historical documents. In recognising scribe-written Syriac manuscripts, Clocksin and Fernando (2003) segmented words based on the distances between the bounding boxes of CCs.

2.3.3 Character Segmentation

Character segmentation involves segmenting words or text lines into their individual characters. The characters can either be in a cursive script or hand-printed and the purpose of the segmentation task is to find the point that separates adjacent characters. As was the case with words, non-uniformity with regards to the spaces between characters contributes towards the difficulty in segmenting them.

Vamvakas et al. (2008) segmented characters in historical documents using a technique based on foreground and background analysis (Chen and Wang, 2000). To segment the characters in scribe-written Syriac manuscripts, Clocksin and Fernando (2003) used a method whereby they scored each pixel with a likelihood of being a segmentation point, where those with the highest likelihood had a narrow horizontal stroke and were close to the baseline.

To segment Bangla characters, Pal and Datta (2003) identified the CCs in a word and then used a set of heuristics to determine whether each CC contained an isolated character or connected characters. Khan and Mohammad (2008) used an ANN to assist in character segmentation. The ANN was trained with correct segmentation points and then used to classify unseen segmentation points as either being correct or incorrect.

In dealing with Vietnamese characters, which contain diacritics, Nguyen and Bui (2008) note that separating diacritics from handwritten base characters is a difficult problem. As a result, they performed recognition of Vietnamese characters without separating the diacritics. This is different to the case of Kumar et al. (2010a) who, when segmenting text lines, removed diacritics and then reattached them at a later stage.

Words can also be segmented into intermediate constituencies between the word representation and individual characters. An example of segmentation at an intermediate level between words and characters was performed by Chen et al. (2010) where they segmented Arabic words into sub-words also referred to as parts of Arabic words (PAWs). The segmentation of Arabic words into PAWs was taken further by Sari et al. (2002) where they identified local minima in the PAWs and then used morphological rules to determine whether or not the local minima were valid segmentation points and segmented the individual characters accordingly.

This review has demonstrated that there are a number of possibilities for the level at which segmentation can take place and some common segmentation strategies have been discussed. It is expected that the level at which segmentation takes place is dependent on the nature of the data, where complexities of the script or the layout of text in a manuscript may require that a certain segmentation strategy be adopted. The intended use of the segmented data may also have an effect on the segmentation strategy chosen as well as the descriptive properties of the features used to describe the data and the capabilities of the machine learning algorithms used in training models and recognising unseen data. In the next section, descriptive features and the role that they play in handwriting recognition is discussed.

2.4 Descriptive Features

Descriptive features are primitive characteristics of an image that make it distinguishable (Pratt, 2007). For instance, the curvature of shapes or the distribution of pixels in

an image could be considered as descriptive features. Objects in a digital image can be described by their descriptive features and unidentified objects in an image can be recognised by comparing their descriptive features to the descriptive features of known objects (Nixon and Aguado, 2008). This is the premise on which handwriting recognition works, where the features of known words and characters are used to train recognition models. The recognition models can then be used to classify unknown words and characters based on their descriptive features.

Nixon and Aguado (2008) describes four important properties of descriptive features in order for them to be useful for recognition:

- Two objects should only have the same descriptors if they are the same.
- Descriptors should be congruent such that two similar objects will have similar descriptors.
- It is convenient for descriptors to be invariant to scale, rotation and translation.
- Descriptors should be compact so that objects are described in an efficient way.

Nixon and Aguado note that there is no set of descriptors that accurately and completely describes general objects and the best descriptive features to use largely depend on the application. Different descriptive features have been used in handwriting recognition for a number of different scripts and collections, both modern and historical. In this section, the use of different descriptive features for handwriting recognition will be discussed.

2.4.1 Features for Handwriting Recognition

There are a number of different features that have been used for handwriting recognition. For instance, Undersampled Bitmaps (UBs), which are based on the density of the normalised number of black pixels within a block in an image, were used by Oliveira et al. (2006) for the recognition of handwritten digits. Oliveira et al. found UBs to perform better than principal component analysis. UBs were also used by Vamvakas et al. (2008) as the features for a system for optical character recognition of historical documents. A similar set of features was used by Vinciarelli et al. (2004).

Geometric Moments (GMs), which are statistical global descriptors of the shape of an image, were used by Clocksin and Fernando (2003) as features for classifying and recognising scribe-written historical Syriac documents. Essentially, moments are weighted averages of pixel intensities (or a function thereof), which is what allows for them to be used as features for handwriting recognition. Clocksin and Fernando used the normalised central moments as well as Hu's (1962) second, third and fourth moments as features for whole character images, overlapping and non-overlapping windows in a character image and a polar transform (Tistarelli and Sandini, 1993) of a character image. Abd and Paschos (2007) also made use of GMs as features for recognising Arabic characters; however, their system was a hybrid system in which structural features, including the number of dots and number of holes in an image, were also used.

Histograms of Oriented Gradients (HoGs) were used as features by Howe et al. (2009) for the transcription of Latin manuscripts. HoGs describe the shape and appearance of

objects in an image by the distribution of local gradients. The images used by Howe et al. had been binarised and the gradients fell into eight directions, plus a grouping for areas with zero gradient. The images were partitioned into grids and the HoGs were extracted at three resolutions.

Marti and Bunke (2002) proposed nine geometric features, which are extracted from an image using a sliding window. The first three of these features characterise the window from a global point of view, while the other six are used to give more details about the writing. These features have been used by Marti and Bunke in a number of studies involving handwriting recognition (Marti and Bunke, 2002; Marti, 2000; Marti and Bunke, 2001). Among other uses of the feature set, Fischer et al. (2009) used it for the automatic transcription of perfectly segmented medieval historical documents and Indermühle et al. (2008) used it in a study comparing writer-specific training and HMM-adaptation.

Favata and Srikantan (1996) used the Gradient, Structural, Concavity (GSC) set of features for a variety of problems, including digit recognition, printed character recognition and handwritten character recognition. The GSC feature set detects gradient features at the local level, structural features at the intermediate level and concavity features at the global level, and uses them to classify and recognise images (Favata and Srikantan, 1996). The GSC feature set was also used by Awaidah and Mahmoud (2009) for the recognition of Arabic numerals.

Gabor Filters (GFs) have been used in character recognition to extract stroke information from characters (Wang et al., 2005). This is possible due to GFs being orientation-specific (Chen et al., 2010) and, since the Gabor Filter is applied at multiple orientations, these features capture information about the orientation of strokes in an image. GF-based features have been shown to work well for automatic handwriting recognition since they operate directly on greyscale images rather than requiring the images to be binarised (Chen et al., 2010). Chen et al. (2010) found GF-based features to perform better than the GSC (Favata and Srikantan, 1996) set of features and graph-based features when recognising handwritten Arabic sub-words. Furthermore, Chen et al. found that using a combination of GF-based and GSC features further improved results. However, in a study comparing structural features, Fourier descriptors, GF-based features and pixel representation features, Haboubi et al. (2009) found GF-based features to perform relatively poorly compared to the other features when recognising Arabic text. The recognition accuracy of the GF-based features was 19.3% while, for structural features, Fourier descriptors and pixel representation features, it was 71.1%, 79.2% and 87.1% respectively. In a study comparing GF-based features and gradient features for recognising handwritten digits in two separate handwritten character databases and a printed Japanese character database, Liu et al. (2005) found that the GF-based features performed better than the gradient features for two of the three databases. In recognising handwritten Kannada Kagunita, which are compound characters that are formed by combining vowels and compounds in the Kannada script, Ragha and Sasikumar (2010) used GFs to acquire a directional version of an image. They extracted moment features from the directional images and found that they provided an improvement on recognition accuracy compared to the original images.

The Discrete Cosine Transform (DCT), related to the Fourier transform, was originally proposed by Ahmed et al. (1974) and expresses a function or signal in terms of a sum of different cosine functions at different frequencies (Nguyen and Bui, 2008). In the original paper on the DCT, Ahmed et al. (1974) note that it can be used as features for pattern

recognition, especially the first coefficients in which most of the variance in an image is encoded. DCT coefficients have been used as features for handwriting recognition. For instance, Nguyen and Bui (2008) used optimum cosine descriptors for multiple strokes in order to recognise Vietnamese characters. AlKhateeb et al. (2008) used DCT coefficients for the recognition of handwritten Arabic words. AlKhateeb et al. found that with less that 15 DCT coefficients their ANN did not converge and that additional coefficients beyond the first 40 did not offer any improvement in results. DCT coefficients were used by Brakensiek and Rigoll (2001) where they extracted the coefficients from a sliding window and then used them for character and word recognition.

Rath and Manmatha (2003) extracted features from columns of words that had been preprocessed to remove slant and skew and that did not contain ascenders or descenders from other words. Both single value features and multi-value features were extracted from each column of a word image. Single value features included the projection profile, partial projection profiles, upper/lower word profiles, background-foreground transitions and greyscale variance. Multi-value features included features from Gaussian smoothing and Gaussian derivatives (Rath and Manmatha, 2003). This same set of features was used by Lavrenko et al. (2004) for word recognition of the text in the manuscripts from the George Washington Collection at the United States Library of Congress.

Ruiz-Pinales and Lecolinet (2000) used the Hough transform to derive local directional features from zones in an image for cursive handwriting recognition. Structural features, which relate to the structure of the image (Heutte et al., 1998), have been used for recognising Arabic handwriting. Examples of these structural features include lines, dots and loops (Lorigo and Govindaraju, 2006). Arabic characters, like characters in the Bushman languages, contain letters that have the same bases but differ due to their diacritics.

As this section has shown, there are a large variety of features that have been used for handwriting recognition. The discussion above is, of course, in no way a complete discussion of all features, the number of which some have speculated ranges in the hundreds (Arica, 1998). However, this section has given a general overview of some of the features that have been used in the literature and provided a background to their use in handwriting recognition. It is difficult to perform a direct comparison among these features due to the different ways that they have been used. However, in Chapter 5 the features that are used in this study are analysed in greater detail in terms of their invariant properties. Furthermore, this study provides a direct comparison of the use of several different features to recognise handwriting while other variables remain fixed. Descriptive features, such as the ones described above, are often used in training machine learning algorithms so that they can be used to recognise unseen data. In the next section, machine learning will be introduced and some of the ways that machine learning algorithms have been used for handwriting recognition will be discussed.

2.5 Machine Learning

The field of machine learning has evolved from the research area of artificial intelligence and the goal to make machines able to mimic the intelligence of humans (Rätsch, 2008). Machine learning specifically focuses on the ability of machines to "learn," which is understood as inductive inference (Rätsch, 2008). One can differentiate between unsupervised

learning, in which an attempt is made to discover hidden irregularities, or supervised learning in which labels are associated with training data for the purpose of classification (Rätsch, 2008). This study focuses on supervised learning and therefore it is the only type of learning discussed here.

Formally, the goal of supervised learning for classification or prediction is to find a functional mapping $f()$ between input data I, in the form of features describing a pattern, and a class label C, such that (Rätsch, 2008):

$$C = f(I) \tag{2.1}$$

Examples of the features that describe a pattern were discussed in Section 2.4.

There are a number of classification or machine learning algorithms, including k-nearest neighbour, linear discriminant analysis and decision trees (Rätsch, 2008). However, for handwriting recognition, Support Vector Machines, Artificial Neural Networks and Hidden Markov Models have proven to be popular and are the machine learning algorithms investigated in this study. In this section, each of these machine learning algorithms will be introduced and the ways in which they have been used for handwriting recognition will be discussed.

2.5.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a Markov model in which the states of the model are not directly observable, but rather can be observed through another stochastic process (Rabiner, 1989). This is in contrast to regular Markov models in which states translate to observable physical events. In this section, HMMs and their use in handwriting recognition will be discussed. The discussion begins with a description of the discrete Markov process and then a description of HMMs and their properties. Lastly, some examples of where HMMs have been used for handwriting recognition will be discussed. This section discusses HMMs from a relatively high level.

2.5.1.1 Discrete Markov Process

In a Markov model, a system can be described as being in any one of N states $S_1, S_2, ..., S_N$ at any given time (Rabiner, 1989). At a regularly-spaced time interval the system is able to move from its current state to any other state in the model (including back to itself) with some probability. The state that the system is in at time t is given by q_t. The probability description of the current state being q_t is dependent on a full specification of the model and is given by (Rabiner, 1989):

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_h, ...]. \tag{2.2}$$

However, in the case of the discrete, first-order Markov chain, this probability description is only dependent on the current stage and previous stage such that:

$$P[q_t = S_j | q_{t-1} = S_i]. \tag{2.3}$$

This allows for the creation of a set of state transition probabilities for a transition a_{ij} from S_i to S_j, such that:

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i] \quad 1 \leq i, j \leq N, \tag{2.4}$$

where the state transitions have the following properties:

$$a_{ij} \geq 0 \tag{2.5}$$

$$\sum_{j=1}^{N} a_{ij} = 1. \tag{2.6}$$

An example of a Markov chain with 5 states as well as the transition probabilities between the states is shown in Figure 2.7.

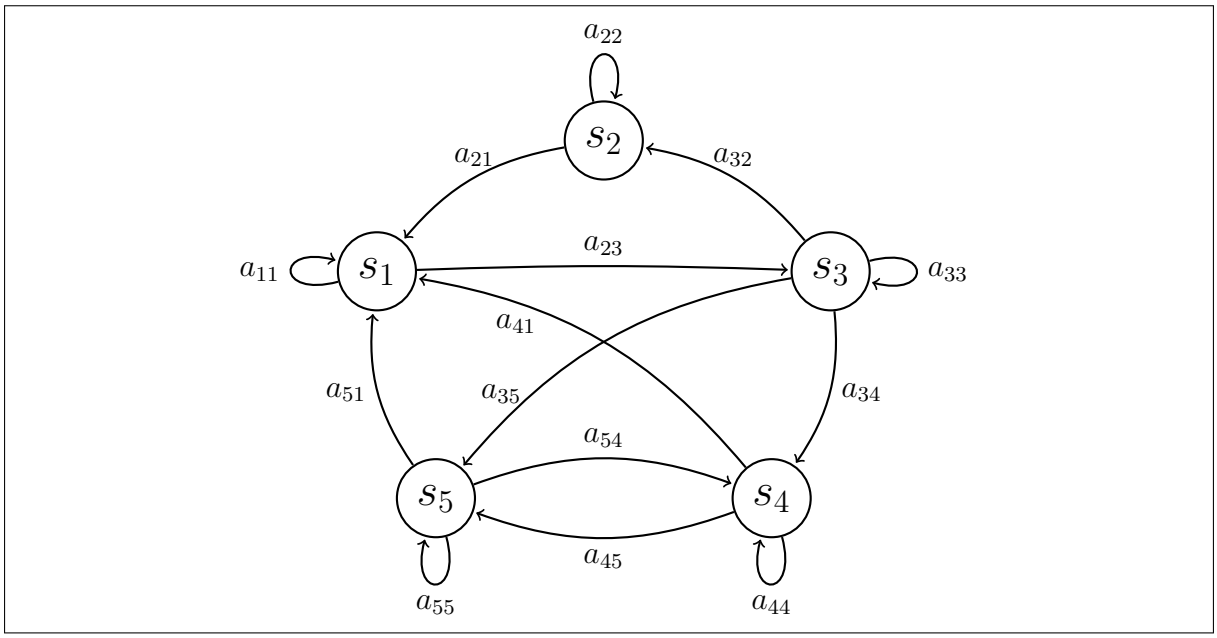


Figure 2.7: A Markov chain with 5 states and state transition probabilities between the states

Given a model that consists of 3 states, a sequence of observations $O = \{S_1, S_2, S_1, S_3, S_2\}$, and a state transition matrix A where,

$$A = a_{ij} = \begin{bmatrix} 0.3 & 0.1 & 0.6 \\ 0.1 & 0.7 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix},$$

then the probability of the observation sequence given the model and state transition probability matrix can be expressed and evaluated as:

$$\begin{aligned} P(O|Model) &= P[S_1, S_2, S_1, S_3, S_2|Model] \\ &= P[S_1] \cdot P[S_2|S_1] \cdot P[S_1|S_2] \cdot P[S_3|S_1] \cdot P[S_2|S_3] \\ &= \pi_1 \cdot a_{12} \cdot a_{21} \cdot a_{13} \cdot a_{32} \\ &= 1 \cdot (0.1)(0.1)(0.6)(0.1) \\ &= 0.6 \times 10^{-3}, \end{aligned}$$

where π_i is the initial state probability and is given by:

$$\pi_i = P[q_1 = S_i] \quad 1 \leq i \leq N. \tag{2.7}$$

In the discussion above the states at each time step have been observed. The next section will discuss HMMs in which the states at each observation are hidden.

2.5.1.2 Hidden Models

In a HMM, the states are not directly observable and a key issue is determining how many states should be in the model and what the states correspond to (Rabiner, 1989). In essence, a HMM is defined by a number of elements, which are illustrated in Figure 2.8 and are discussed below.

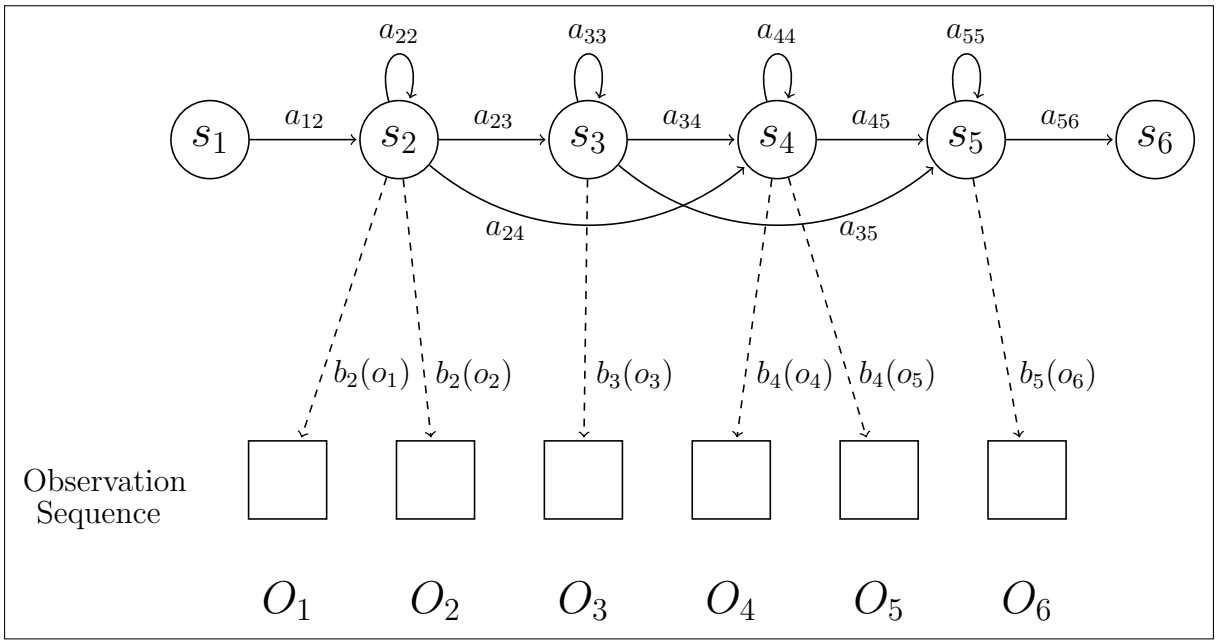


Figure 2.8: A Markov model

Model Properties A HMM is described by the following parameters:

- **The number of states in the model:** In a Markov model there are N states. In a regular Markov model these states are directly observable, however, in a HMM these states are not observable and only the observation sequence O is known. In Figure 2.8 a Markov model is shown with a total of six states. In many cases, all states are connected and reachable from every other state in what is known as an ergodic model. However, there are some model designs, such as the left-to-right model, in which every state is not reachable from every other state (Rabiner, 1989).
- **The number of observation symbols per state:** For each state in the HMM, there are M distinct observation symbols (Rabiner, 1989). The symbols correspond to the output of the system being modelled. These observations make up the observation sequence O.
- **The state transition probability distribution:** There is a distribution A of state transition probabilities between any pair of states S_i and S_j such that

$$A = a_{ij} \quad 1 \leq i, j \leq N. \tag{2.8}$$

- **The observation symbol probability distribution:** In state j, there is a symbol observation probability distribution $B = b_j(k)$ where:

$$b_j(k) = P[v_k \text{ at } t | q_t = S_j], \quad 1 \leq j \leq N, 1 \leq k \leq M, \tag{2.9}$$

  where v_k at t is the symbol observation at time t.

- **Initial state probability:** There is an initial state probability π where:

$$\pi_i = P[q_1 = S_i] \tag{2.10}$$

Parameter Estimation The parameters of a HMM need to be estimated in order to use the model for recognition. Given a set of training samples for a particular model, one of the ways that the parameters of the model can be determined automatically is by using a re-estimation procedure known as the Baum-Welch Re-Estimation (Young et al., 2006). An explanation of the Baum-Welch Re-Estimation technique is beyond the scope of this section, however, in essence it works by first making a rough guess as to what the parameters might be and then re-estimating the parameters by computing posterior estimates and maximum likelihood estimates (Baum et al., 1970).

Recognition As is shown in Figure 2.8, the output of a HMM is an observation sequence $O = o_1, o_2, ..., o_T$. Using character recognition as an example, given a HMM M_i for each character C_i and an observation sequence O, then:

$$P[O|C_i] = P[O|M_i]. \tag{2.11}$$

Using Equation 2.11, the probability of the observation sequence belonging to each character class can be calculated. This is commonly performed using the Viterbi algorithm (Rabiner, 1989).

2.5.1.3 Use in Handwriting Recognition

HMMs have been used for recognition of handwritten texts in a number of studies. They have been used both for line and word recognition and one of the benefits that they provide is that do no not require text lines to be segmented into words and for words to be segmented into characters since the segmentation is a result of the Viterbi decoding (Indermühle et al., 2008). In this section some studies that have used HMMs will be discussed. Howe et al. (2009) used a HMM-based recogniser in a study in which they investigated freeform character recognition in historical documents. Su et al. (2007) made use of a HMM-based recogniser for the transcription of handwritten Chinese text. Su et al. modeled 1965 Chinese character models and were able to achieve a maximum accuracy rate of 55.58%. Marti and Bunke (2002) made use of a HMM-based recogniser in a study in which they investigated the effect of vocabulary size and statistical language models on a cursive handwriting recognition system. Indermühle et al. (2008) used a HMM-based recogniser in a study in which they compared writer-specific training and HMM-adaptation and found that the best strategy for building a handwriting recognition system for a specific author was to take a general handwriting recognition system and then adapt it to that specific author with a small amount of training data created by

that author. Fischer et al. (2009) used a HMM-based recogniser for the recognition of scribe-written medieval texts and achieved a word accuracy rate of 88.86%.

In a pilot study that took place as part of this research (see Chapter 3) a HMM-based recogniser was created to recognise a subset of the Bushman texts in the Bleek and Lloyd Collection (Williams and Suleman, 2011).

Zimmermann and Bunke (2002) and Günter and Bunke (2003) have studied techniques for optimising the parameters of a HMM for handwriting recognition and showed how these optimisations could increase recognition accuracy by around 10% (Günter and Bunke, 2003). HMMs were also used for text line recognition by Toselli et al. (2007) where they compared the effort required to create perfect transcriptions of text. The comparison was made between a fully automated system where errors were corrected post-recognition and an interactive process whereby a human corrected recognition errors as they occurred and it was found that correcting errors as they occurred led to a 22% reduction in the effort required to produce error free transcriptions.

This section has discussed HMMs and their use for handwriting recognition. In the next section, ANNs, which have also successfully been used for handwriting recognition will be discussed.

2.5.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) attempt to mimic the biological neural networks that exist in the human brain for very specific tasks and are usually implemented in software or electronics (Haykin, 1998). ANNs have a number of beneficial properties such as: they are non-linear; they have input-output mapping; they are adaptive; they have an evidential response; they provide contextual information; they are fault tolerant; and they are uniform in terms of design and analysis (Haykin, 1998).

In this section, ANNs will be discussed and the discussion is based on Haykin's (1998) book on Neural Networks. The discussion begins with an introduction to ANNs and how they work followed by some examples of where ANNs have been used for handwriting recognition.

2.5.2.1 The Neuron

A neuron is an information processing unit and is fundamental to the working of an ANN (Haykin, 1998). A basic model of a neuron is shown in Figure 2.9 and the various components that make up a neuron are discussed below.

Input Signals The input signals $x_1, x_2, ..., x_n$ are the signals that are fed into the neuron, k, which performs the summing function.

Synaptic Weights The synaptic weights are the strengths assigned to each of the connections between the inputs and the neuron k. These synaptic weights have either positive or negative values (Haykin, 1998). The synaptic connection from input i to neuron k is represented by W_{ki}.

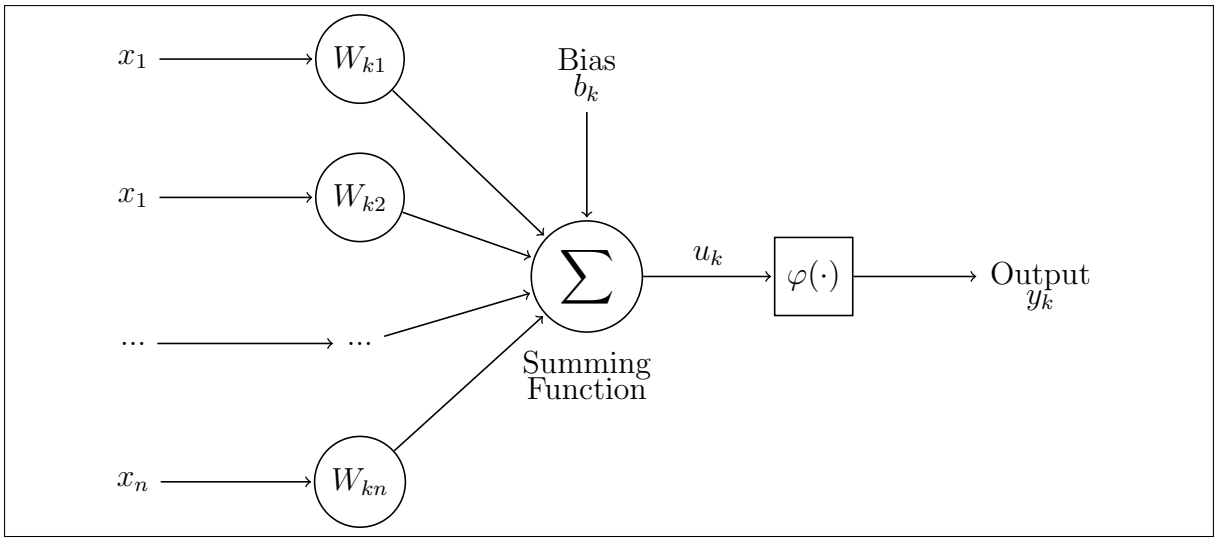


Figure 2.9: A neuron

The Bias The bias, b_k, is an externally applied value that has the effect of either increasing or decreasing the net value of the summation of the synaptic weights in the model (Haykin, 1998).

Summing Function There is a summing function that sums the products of each of the input signals and their corresponding synaptic weights. The sum of these products, u_k is given by:

$$u_k = \sum_{j=1}^{n} x_j W_{kj} \tag{2.12}$$

The addition of the bias then produces the output of the summing function which is given by v_k such that:

$$\begin{aligned} v_k &= b_k + u_k \\ &= \sum_{j=0}^{n} x_j W_{kj}, \quad x_0 = +1, W_{k0} = b_K \end{aligned} \tag{2.13}$$

Activation Function The activation function, $\varphi(\cdot)$, limits the amplitude of the output of a neuron (Haykin, 1998). The amplitude function is also commonly referred to as the squashing function and it squashes the range of the output of the neuron to some finite range, normally [-1;1] or [0;1] (Haykin, 1998). The output, y_k, of the neuron, k, can then be described by:

$$y_k = \varphi(v_k) \tag{2.14}$$

There are generally three basic types of activation functions that are used: the threshold function, the piecewise function and the sigmoid function.

Activation functions usually have the range [0;+1], though it is possible to adapt them such that they have the range [-1;+1] (Haykin, 1998).

2.5.2.2 Network Architectures

ANNs can have a specific architecture (structure) and there are, in general, three different network architectures: single-layer feed-forward networks, multi-layer feed-forward networks and recurrent networks (Haykin, 1998). In this section, only multilayer ANNs are discussed since it is the only architecture used in this study.

Multi-layer Feed-forward Networks Multi-layer feed-forward networks contain one or more hidden layers of neurons that intervene between the input layer and the output layer (Haykin, 1998). The addition of hidden layers allow for the extraction of higher-order statistics and is especially useful when the input layer is large (Haykin, 1998). In a multi-layer network, the input to each layer generally comes from the preceding layer and the output from the final layer makes up the output of the whole network (Haykin, 1998). An example of a multi-layered feed-forward network is shown in Figure 2.10. In the figure, there are nine input nodes, a hidden layer consisting of four hidden neurons and an output layer consisting of two neurons. The network in Figure 2.10 is said to be fully connected since every neuron in one layer is able to reach every neuron in its adjacent layer. In cases where this is not true, the network is said to be partially connected (Haykin, 1998).

2.5.2.3 Learning

Learning in a neural network takes place through the adjusting of the synaptic weights and bias and the learning process in an ANN is essentially made up of the following series of steps:

1. **Stimulate:** The ANN is stimulated by a series of inputs in its environment.
2. **Adapt:** The ANN undergoes a series of changes by adapting its free parameters (synaptic weights and bias) based on the stimulation it has received.
3. **Respond:** The ANN should respond differently as a result of the stimulation and subsequent adaptation (Haykin, 1998).

The way in which the ANN adapts its free parameters is determined by the learning algorithm used. There are five basic learning rules for ANNs and these are: error-correction learning, memory-based learning, Hebbian learning, competitive learning and Boltzmann learning (Haykin, 1998).

Supervised Learning in a Neural Network Supervised learning is the type of learning used in this study. Unsupervised learning is beyond the scope of this study and as such is not discussed here. In supervised learning, the ANN is provided with the desired output for a specific training vector. The difference between the actual output and the desired output is then determined and the free parameters of the ANN are adjusted iteratively until the ANN is able to produce the correct output and can then be used for recognition (Haykin, 1998). This type of learning is known as error-correction learning and is depicted in Figure 2.11. The figure shows how both the "teacher" and the learning system are presented with the same training vector and the "teacher" has knowledge of

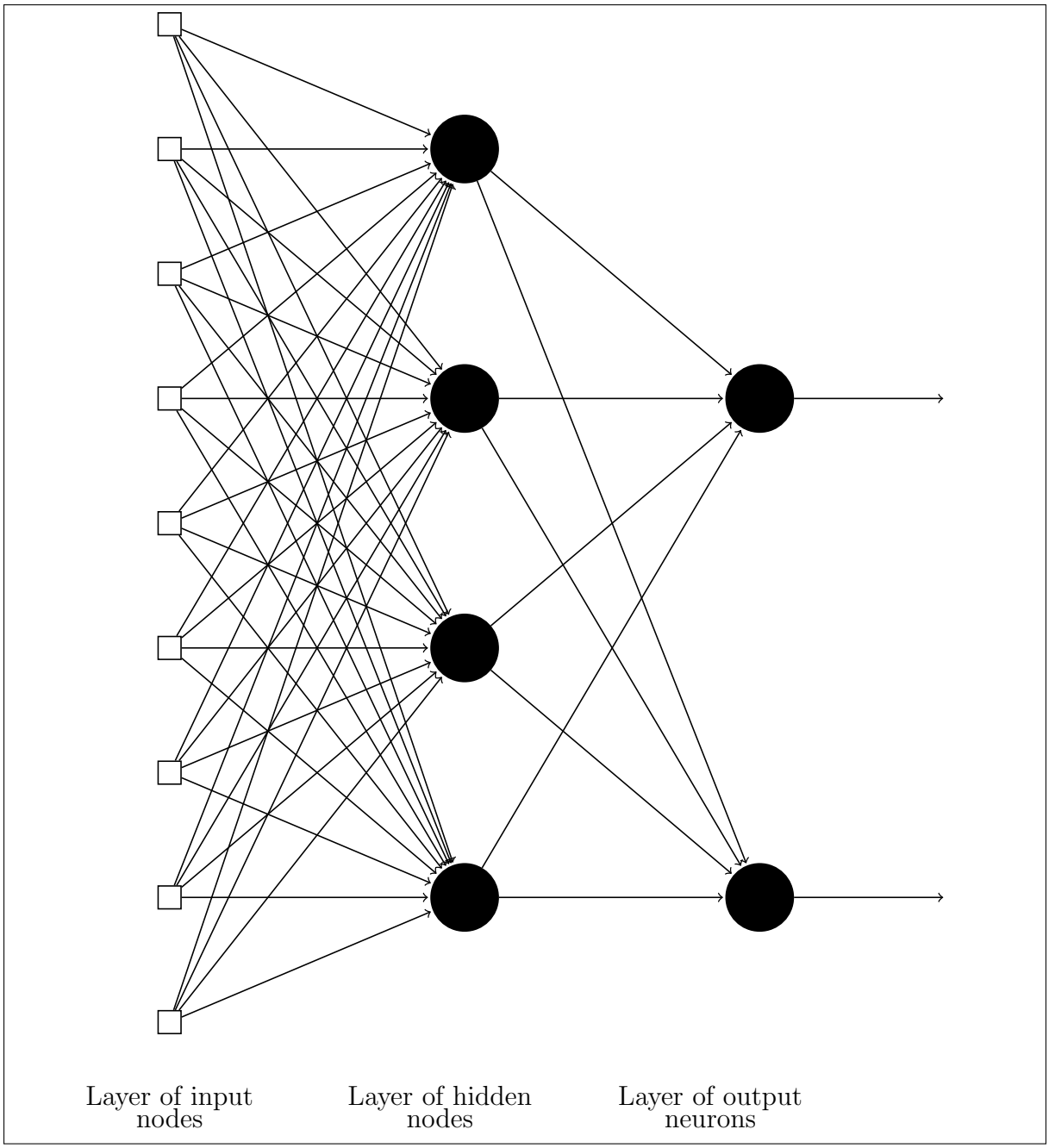


Figure 2.10: A multi-layered feed-forward network

what the desired output is. This desired output is compared to what the learning system produced as output to produce an error signal that is used with the training vector to adjust the network parameters.

2.5.2.4 Use in Handwriting Recognition

ANNs have been used in a number of studies to recognise handwriting. For instance, Ruiz-Pinales and Lecolinet (2000) used the Hough transform to compute local directional features for cursive handwriting recognition and then used a three-layered ANN for recognition. Ragha and Sasikumar (2010) used a multi-layer ANN to recognise Kannada

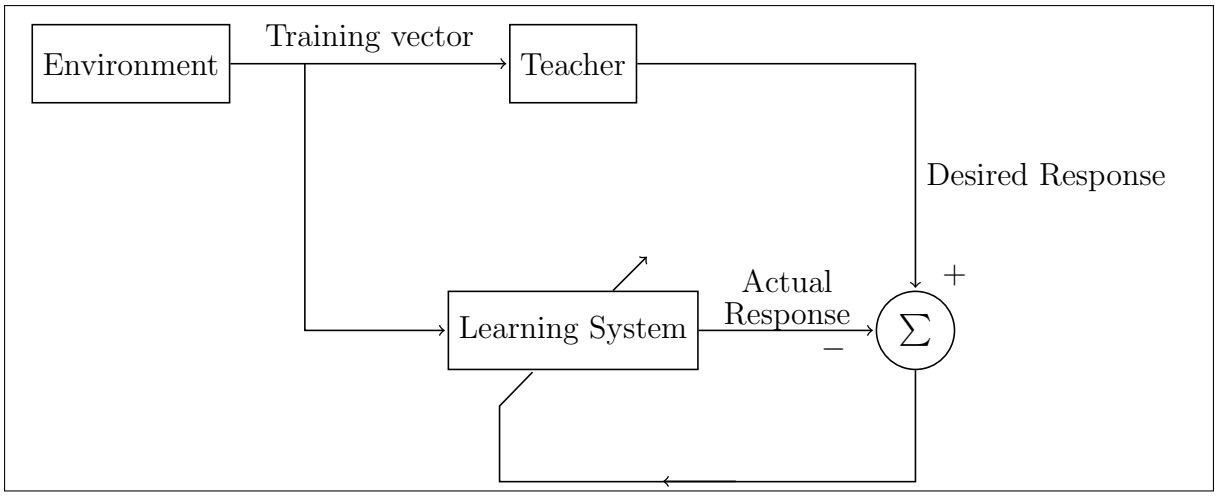


Figure 2.11: Supervised learning in a ANN

Kagunita characters. Singh et al. (2009) used an ANN to recognise handwritten Hindi characters from which they had extracted gradient features using a Sobel operator.

A Recurrent Neural Network (RNN), which is a network architecture in which there is at least one feedback loop (Haykin, 1998), was used by Graves et al. (2009) to recognise texts in the IAM database and found to perform significantly better than a state-of-the art HMM by achieving a recognition rate of 74.1%, which was a relative error reduction in excess of 40% in some cases. Espana-Boquera et al. (2011) used a hybrid ANN-HMM for recognising the same texts in the IAM database under the same set of test conditions and, interestingly, found the two systems to perform exactly the same despite the differences between the RNN and hybrid ANN-HMM model architectures. A hybrid ANN-HMM system was also used by Tay et al. (2003) for recognising handwritten cursive French words.

Fischer et al. (2009) created a system for recognising handwritten medieval manuscripts. Their system made use of a bi-directional long short-term memory (BLSTM) ANN, which is a special case of a RNN, as well as HMMs for each letter, which were then concatenated to build HMMs for words. The ANN contained two hidden layers and Fischer et al. et al found the ANN-based classifier to perform better than the HMM-based classifier with each classifier achieving word accuracy rates of 93.32% and 88.69% respectively. The same BLSTM ANN was used by Frinken et al. (2009) in a study in which they analysed a combination of classifiers for handwriting recognition.

Frinken and Bunke (2010) created a semi-supervised handwriting recognition system based on the BLSTM ANN. The system was trained with labelled data and then used to recognise unlabelled data. The recognised data was then sorted by recognition confidence and the data with the highest levels of confidence were used to retrain the ANN. A series of experiments showed how this semi-supervised approach significantly improved the performance of the recognition system.

This section has discussed ANNs and their use in handwriting recognition. In the next section, Support Vector Machines are discussed.

2.5.3 Support Vector Machines

The Support Vector Machine (SVM) was proposed by Cortes and Vapnik (1995) as a machine learning technique for a 2-group classification problem. The goal of a Support Vector Machine (SVM) is to find hyper-planes capable of separating points in hyperspace (Cristianini and Shawe-Taylor, 2000). In this section SVMs will be introduced, beginning with a description of SVMs and then a discussion of some studies that have used SVMs for handwriting recognition.

Given a set of labelled data $(x_i, y_i), i = 1, 2, ..., n$, where $x_r \in R^n$ and $y \in \{1, -1\}^n$, the x_i are mapped to a higher dimensional space, which is potentially infinite, by the function ϕ and, for a linear SVM, the SVM attempts to find a linear separating hyper-plane in this new space that maximises the space between different classes (Hsu et al., 2003). An example of this is shown in Figure 2.12.

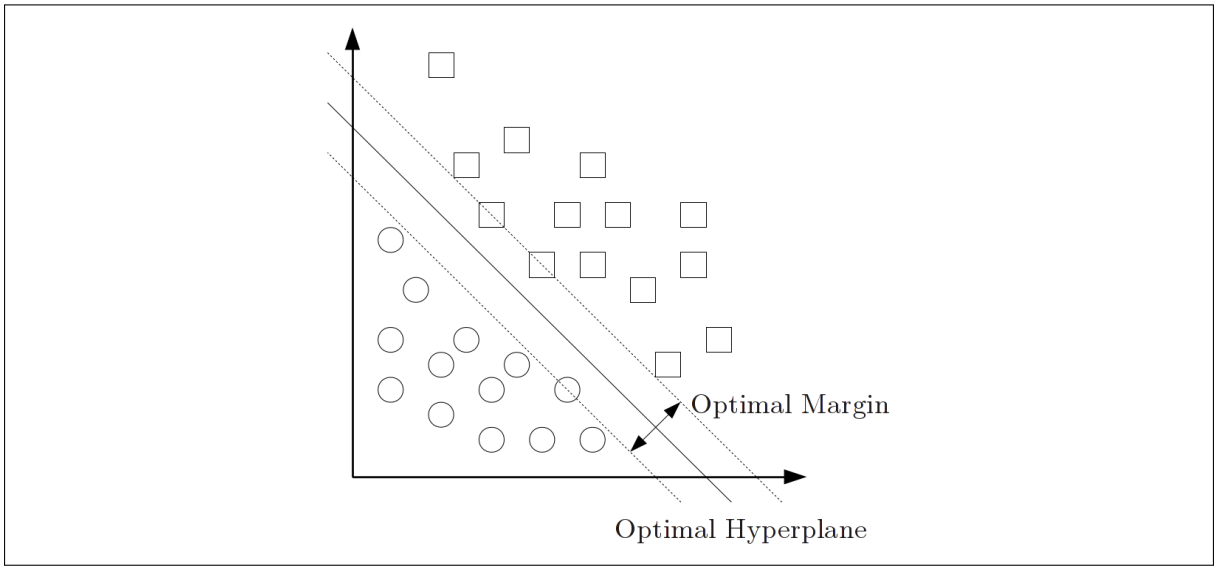


Figure 2.12: Optimal hyperplane that maximises the space between classes

SVMs are traditionally linear classifiers, however, non-linear classifiers can be created (Boser et al., 1992).

There are four kernel functions which are popularly used (Hsu et al., 2003), where γ, r and d are the parameters of the kernel.

- **Linear Kernel:** $K(x_i, x_j) = x_i^T x_j$.
- **Polynomial Kernel:** $K(x_i, x_j) = \gamma(x_i^T x_j + r)^d, \gamma > 0$.
- **Radial Basis Function (RBF) Kernel:** $K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2), \gamma > 0$.
- **Sigmoid Kernel:** $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$.

The use of SVMs for classification involves data scaling, model selection, training and testing.

2.5.3.1 Multi-Class Support Vector Machines

SVMs were originally proposed for 2-class classification, however, they can be used for multi-class classification where the number of classes exceeds 2. There are generally two approaches to multi-class classification. The first approach is known as the "one-vs-all" approach and involves constructing N support vector machines for the N classes and then classifying one class against the other $N-1$ classes (Abd and Paschos, 2007). The second approach, known as the "one-vs-one" approach, involves constructing $\frac{N(N-1)}{2}$ support vector machines in order to create 2-class classifiers for every pair of different classes.

2.5.3.2 Use in Handwriting Recognition

In proposing a methodology for Optical Character Recognition (OCR) for historical documents, Vamvakas et al. (2008) made use of a SVM with a RBF kernel. The same SVM and kernel were used by Clocksin and Fernando (2003) to recognise scribe-written Syriac text and by Chen et al. (2010) for Arabic sub-word recognition. Similarly, Abd and Paschos (2007) also used a SVM with a RBF kernel to recognise Arabic characters. A SVM with a linear kernel was used by Shrivastava and Gharde (2010) to recognise handwritten Devanagari digits.

Nguyen and Bui (2008) used a SVM with a RBF kernel as well as an ANN to recognise Vietnamese characters. Nguyen and Bui found the RBF SVM-based classifier to perform better than the ANN-based classifier because it minimised the structural risk. Nguyen and Bui used a two layer recognition system in which the first layer involved recognising characters without diacritics, while the second layer involved recognising the diacritics.

Feng and Manmatha (2005) performed an investigation comparing SVMs, maximum entropy models and naive Bayes learning algorithms and their applicability to the publicly available George Washington collection of letters. They compared their results to a study on the application of a HMM to the same dataset (Lavrenko et al., 2004). Feng and Manmatha found that naive Bayes outperformed the SVM, maximum entropy models and HMM, with a maximum accuracy of 64% while the others scored 52.8%, 52.3% and 58.6% respectively. Even though the naive Bayes outperformed the SVM classifier, SVM classifiers still remain a popular machine learning algorithm and for that reason are included in this study. The naive Bayes is not included for reasons related to scope, but future work could investigate the use of other classifiers for recognising the Bushman texts.

In a pilot study for this research, a SVM with a RBF kernel was used to recognise characters in a neatly rewritten version of a Bushman story from the Bleek and Lloyd Collection and a recognition accuracy of approximately 80% was achieved (Williams, 2010).

This section has briefly introduced SVMs and their use in handwriting recognition. The studies cited have shown that the RBF kernel is a popular choice for SVMs. A possible reason for this is that the RBF is more discriminant than the linear kernel (Chen et al., 2010) but has fewer parameters than the polynomial kernel. A RBF kernel requires that the kernel parameter γ and penalty C be specified, for which the optimal values can easily be found using a grid search (Chang and Lin, 2001). For other kernels that have more

parameters, a grid search is not appropriate and other methods need to be employed in order to determine the parameter values.

In handwriting recognition, machine learning algorithms are trained to recognise unseen data. In this section, three popular machine learning algorithms have been discussed, including a theoretical description and their use in handwriting recognition in the literature. It was shown that the different machine learning algorithms have been used for studies in recognising a wide variety of languages and scripts. The use of HMMs, ANNs and SVMs for the automatic recognition of Bushman texts is described in Chapter 6. In the next section, statistical language models, which commonly form part of a handwriting recognition system, are discussed.

2.6 N-Gram Language Models

In natural language the distribution of words and characters is not uniform and, as such, statistical information about these distributions can be used to improve handwriting recognition results and correct errors (Marti, 2000). In order to create a language model, assumptions need to be made about the statistical nature of a language or language characteristics can be extracted from representative samples of the language using automatic techniques (Marti, 2000).

An N-gram language model is a statistical language model in which the probability of a unit is based on the previous $N-1$ units that occurred before it (Jurafsky and Martin, 2008). For instance, in the phrase, *the dog ...*, the word *barked* is likely to have a higher probability of following the sentence than the word *meowed* and this can be determined based on an N-gram language model that models the probability of the words *barked* and *meowed* coming after the word *dog*. For a unigram language model, the probability of a unit occurring is simply based on the actual distribution of units in the language (Marti, 2000).

Formally, given a sequence of units, $u_1, u_2, u_3, ..., u_{n-1}, u_n$ that can also be represented as u_1^n, the probability of them occurring can be represented by:

$$P(u_1, u_2, u_3, ..., u_{n-1}, u_n), \tag{2.15}$$

and using the Chain Rule of Probability, this can then be calculated by:

$$P(u_1^n) = P(u_1)P(u_2|u_1)P(u_3|u_1^2)...P(u_{n-1}|u_1^{n-2})P(u_n|u_1^{n-1}). \tag{2.16}$$

Because calculating the probability of a unit occurring based on all units that occurred before it is complex for a long sequence of words, an N-gram language model calculates the approximate probability of a unit occurring, based only on the $N-1$ units that occur before it (Jurafsky and Martin, 2008). For instance, in a bigram language model the probability of a word occurring can be approximated by,

$$P(u_n) = P(u_n|u_{n-1}). \tag{2.17}$$

This approximate conditional probability can be generalised for any N, such that

$$P(u_n|u_1^{n-1}) \approx P(u_n|u_{n+N-1}^{n-1}). \tag{2.18}$$

A shortcoming with N-Gram language models is that no training corpus can ever be general enough to accurately represent an entire language, resulting in zero probabilities for certain occurrences of words (Jurafsky and Martin, 2008). In order to correct this, a technique called smoothing is used to replace zero probabilities with non-zero values (Jurafsky and Martin, 2008). Another technique for improving N-gram language models when there is not enough training data is called back-off and involves using a lower order N-gram in place of higher order N-grams (Jurafsky and Martin, 2008).

2.6.0.3 Use in Handwriting Recognition

N-gram language models have been used in handwriting recognition to improve recognition accuracy by incorporating statistical information about the language being recognised into the classifier. Vinciarelli et al. (2004) tested unigram, bigram and trigram statistical language models on three different English datasets in order to see the extent to which they were able to improve recognition results. For the Cambridge dataset, which was written by a single person, Vinciarelli et al. showed that unigrams, bigrams and trigrams performed equally well in improving recognition and accuracy, but noted that the N-grams were only useful in recognising short functional words. For the IAM dataset, which was written by multiple authors, it was shown that an increase in the order of the language model produced a significant improvement. Lastly, for the Reuters dataset, Vinciarelli et al. showed that bigrams and trigrams performed better than unigrams, with bigrams performing marginally better than trigrams.

Marti and Bunke (2001) demonstrated the influence that the size of a vocabulary has on the effect of a N-gram language model. Marti and Bunke showed that for a small vocabulary, a bigram language model improved accuracy by 2.74% from 78.53% to 81.27%. However, on a larger vocabulary that contained 7719 words, the use of a bigram language model improved accuracy from 40.47% to 60.05%.

Brakensiek and Rigoll (2001) note the benefits of character-based N-gram language models for an improvement in recognition of unseen vocabularies. Brakensiek and Rigoll showed how the use of 5-gram and 7-gram character language models decreased relative error rates by 70% for degraded documents and 50% for documents in a handwritten database. In another study, John F. Pitrelli (2001) demonstrated how the use of a unigram language model corrected 45% of the errors made by a system which made use of a character-based language model. Lastly, when recognising words in letters from the George Washington Collection, Lavrenko et al. (2004) showed how a unigram word language model reduced word error rates from 53% to 45% and how a bigram word language model further reduced word error rates to 41%.

In this research, a N-gram language model is built for the Bushman languages that appear in the Bleek and Lloyd Collection and then evaluated in order to determine the extent to which it can improve recogniser performance.

2.7 Discussion

This chapter has provided a background to this study. The discussion began with an introduction to cultural heritage preservation and its goals of identifying, conserving, protecting, presenting and ensuring the transmission of cultural heritage materials to future

generations. Transcriptions of handwritten documents assist in fulfilling these goals by ensuring that the documents' contents are available, even if the physical artefacts themselves no longer exist. However, manual transcription is a tedious and time consuming task and automatic transcription can provide an alternative means of achieving the same result. This chapter has discussed some of the techniques that can be used for automatic transcription and has shown how widely they vary. Furthermore, it has suggested that it is not always obvious which techniques may be the most appropriate when creating a handwriting recognition system and that the choice of techniques depends on a number of factors, such as the nature of the documents that are to be recognised. This chapter has also discussed some of the core concepts related to handwriting recognition including the use of corpora for recognition, segmentation, descriptive features, machine learning algorithms and statistical language models and, in doing so, has provided a background to the vast amount of research done in this area and the research presented in this thesis.

Chapter 3

Pilot Studies

Two pilot studies were conducted at an early stage of this study in order to investigate the feasibility of this research. The first pilot study involved the recognition of characters in a neatly re-written version of a Bushman story (Williams, 2010). The second pilot study involved the recognition of a small subset of text lines of Bushman text using a Hidden Markov Model (HMM) (Williams and Suleman, 2011). In this chapter, these two pilot studies are briefly discussed.

3.1 Pilot Study I: Recognition of Neatly Rewritten Characters

The purpose of the first pilot study was to provide an initial investigation into the feasibility of the study. This pilot study focused on the automatic recognition of a limited set of characters from a |xam Bushman story. The recognition was performed using a Support Vector Machine (SVM) (Cortes and Vapnik, 1995) to classify the characters. The text that was recognised was a neatly rewritten version of the first page of *A Story of the Girl who made the Milky Way* (Figure 3.1), which appears in one of Lucy Lloyd's |xam notebooks. Two authors participated in this study with the purpose of evaluating the ability to recognise the handwriting of multiple authors. On this notebook page there are 39 different character classes, which are shown in Figure 3.2

Authors created both training and testing data. Author 1 provided 10 samples of each character in the story as training data and author 2 provided 3 samples of each character in the story as training data. Each author then neatly wrote out the story, which was then used as testing data for recognition. An example of the story in English, |xam and classified into character classes is shown in Figure 3.3.

3.1.1 Implementation

The implementation of this pilot study involved line, word and character segmentation, feature extraction and the training of the SVM. The pages of text were segmented into words, lines and characters using Connected Component (CC) analysis (Kong and Rosenfeld, 1996). In cases where there were segmentation errors, they were manually corrected

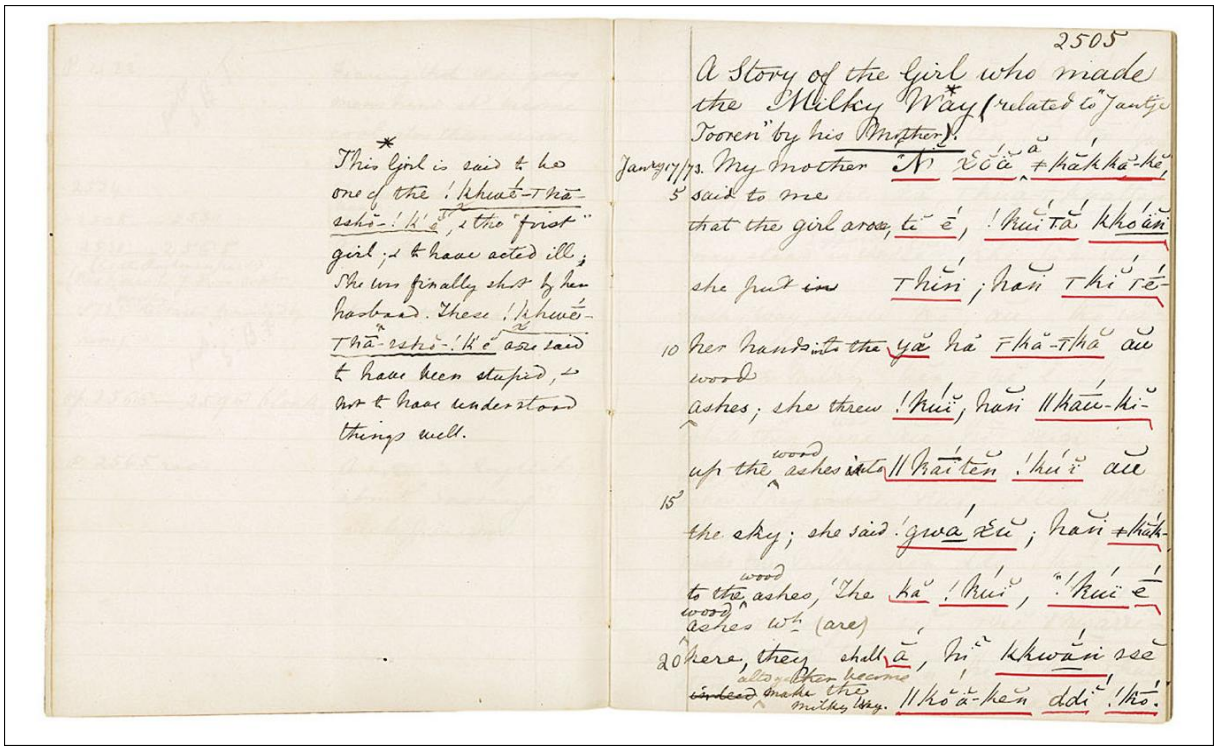


Figure 3.1: The notebook page showing *A Story of the Girl who made the Milky Way*

or the pages of text were digitally altered until they could be perfectly segmented using the automatic techniques.

Each character was normalised by resizing it to a 32x32 pixel block, which was then thresholded. A 4x4 pixel sliding window was used to calculate Undersampled Bitmaps (UBs) (see Section 5.2) and normalised over the range [0-1].

A SVM with a Radial Basis Function (RBF) kernel was trained using the features from each training sample provided by the authors. Three recognition models were trained: one for each author individually and one for the two authors combined. The three models had the following properties:

1. Model A1: Author 1 - 388 samples.
2. Model A2: Author 2 - 117 samples.
3. Model A1_2: Authors 1 & 2 - 505 samples.

3.1.2 Evaluation

The pilot recognition system was evaluated using the testing data, which was a neatly rewritten version of the story by each author and which had been excluded from the training set. Cross validation was not used in this pilot study since each author specifically created testing data for the experiment; however, cross validation was used in all of the other experiments described in this dissertation. The testing data was automatically segmented and features were extracted for each character. These characters were then recognised for each of the six (3×2) author/model combinations below:

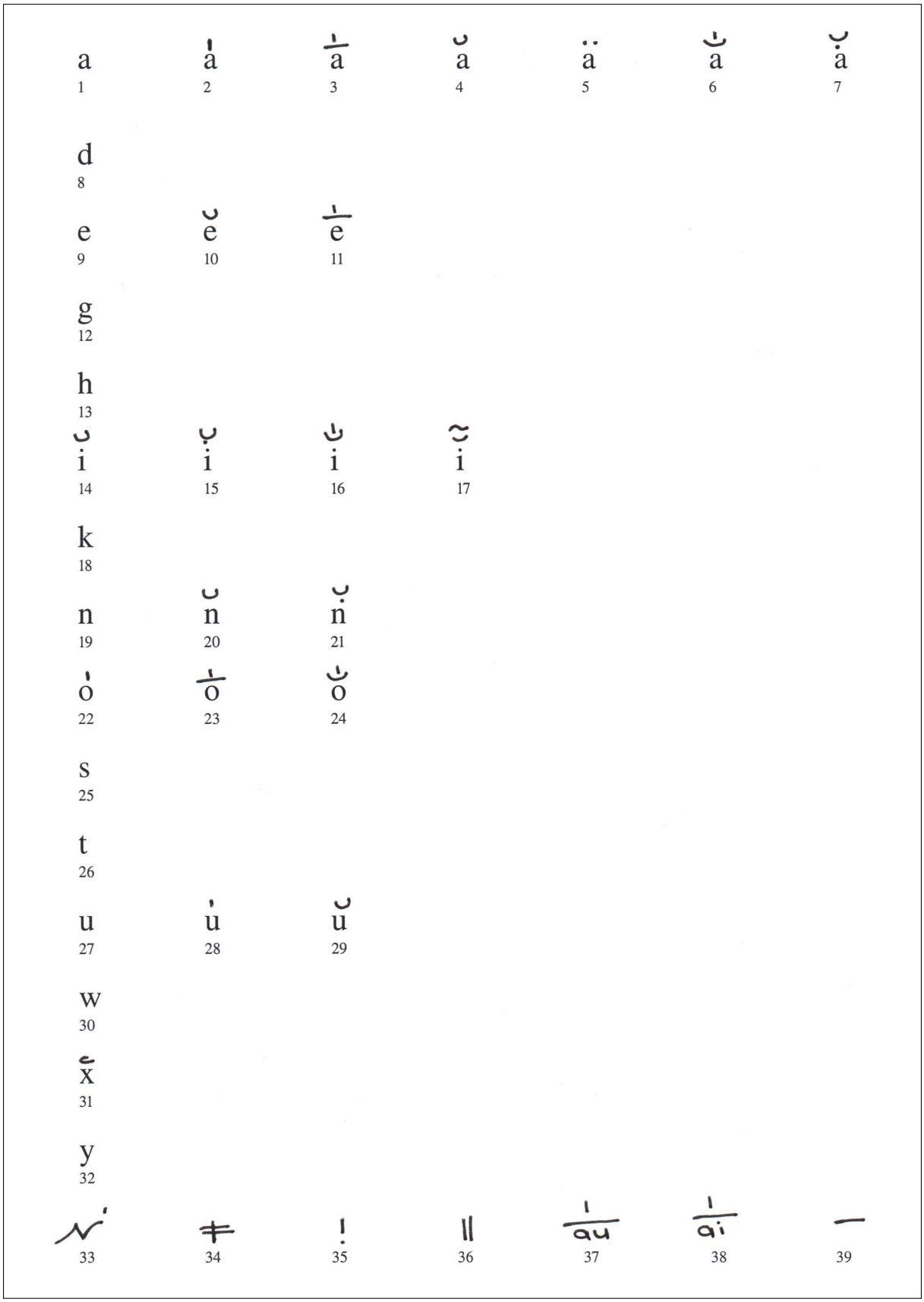


Figure 3.2: Characters that appear in *A Story of the Girl who made the Milky Way*

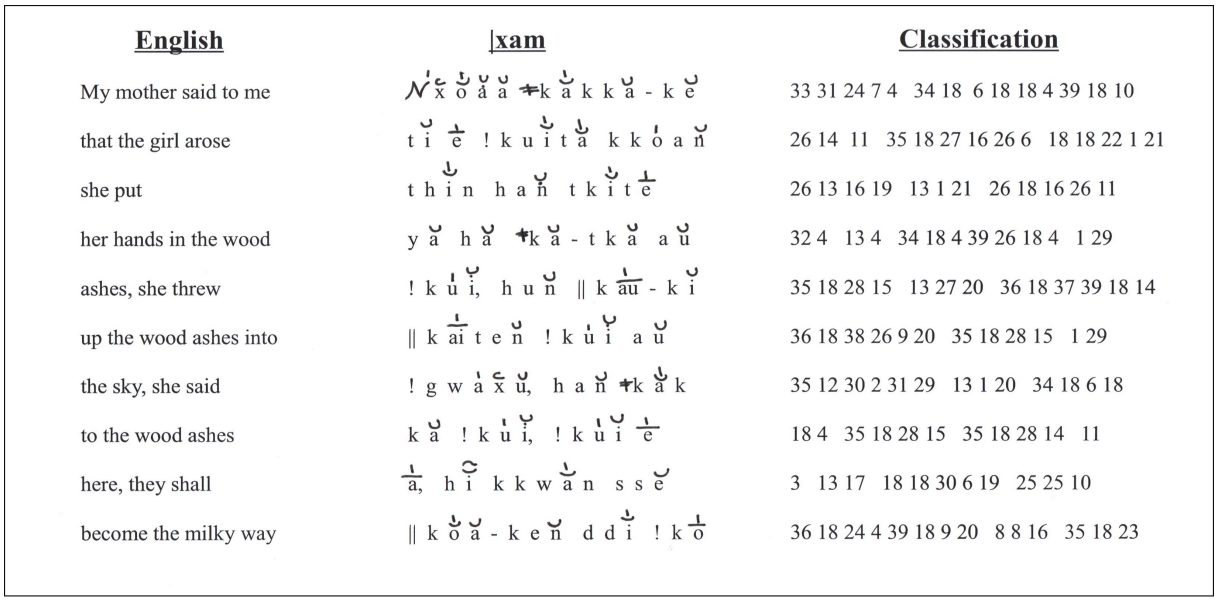


Figure 3.3: English, |xam, and transcribed/classified versions of *A Story of the Girl who made the Milky Way*

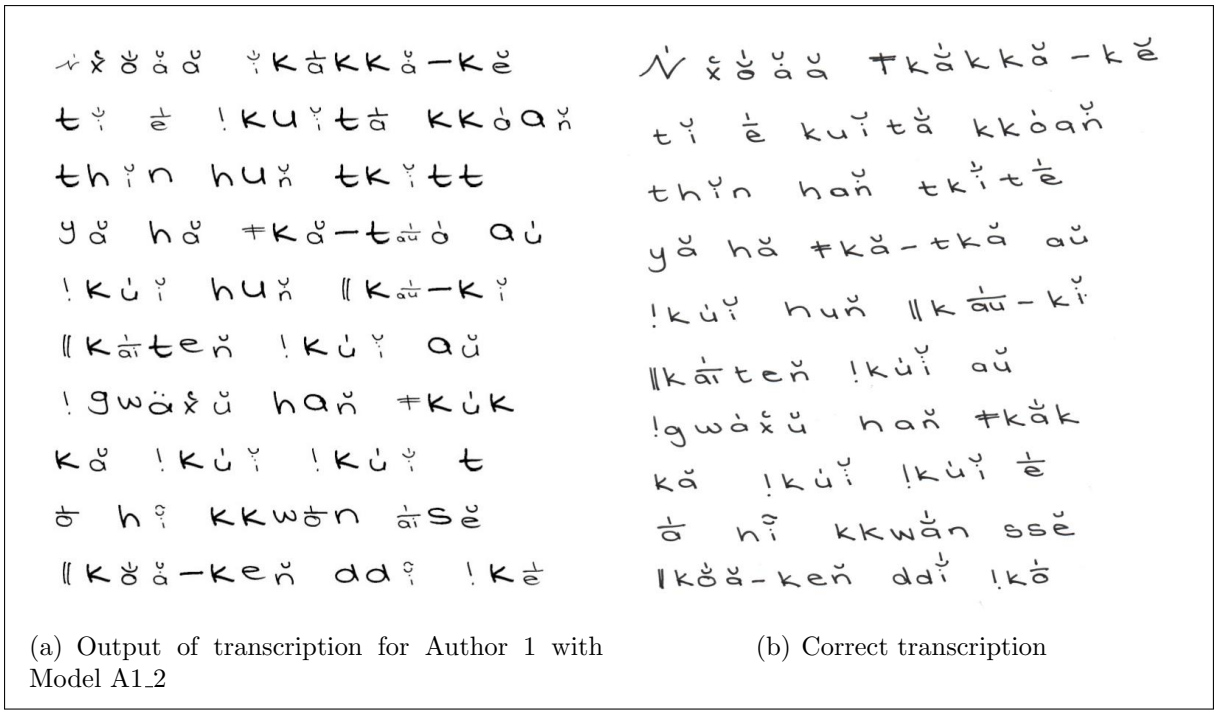
- Story by Author 1 and Model A1.
- Story by Author 2 and Model A1.
- Story by Author 1 and Model A2.
- Story by Author 2 and Model A2.
- Story by Author 1 and Model A1_2.
- Story by Author 2 and Model A1_2.

Table 3.1 shows the recognition accuracy as defined by Equation 6.1 and Figure 3.4 shows the recognition output of the page that achieved the highest accuracy, alongside the correct transcription of the same page. Figure 3.4 shows sample character images since this pilot study was conducted before the Bushman text encoding scheme described in Chapter 4 had been developed.

Table 3.1: Transcription accuracy for first pilot study

	Model A1	Model A2	Model A1_2
Author 1	77.17%	28.35%	79.53%
Author 2	36.22%	62.99%	71.65%

The results show that using an author's training set to transcribe that author's handwriting can be done with a satisfactory level of accuracy - approximately 77% and 63% for authors 1 and 2 respectively. Similarly, the results show that using one author's training set to transcribe another author's handwriting results in poor levels of accuracy. Transcribing the handwriting of author 2 using author 1's training set only led to approximately 36% accuracy. However, by augmenting the relatively large training set of author 1 with the relatively small training set of author 2, the accuracy increased to



(a) Output of transcription for Author 1 with Model A1_2

(b) Correct transcription

Figure 3.4: Comparison of most accurate transcription with correct transcription for first pilot study

approximately 72%. This finding is in line with the findings of Indermühle et al. (2008), in which HMMs were used to transcribe handwritten historical documents and where it was shown that augmenting a general training set with author specific training samples improved accuracy. The highest accuracy - approximately 80% - was achieved using the augmented training set of both authors. A possible reason for this is that the augmented training set accounts for more variability due to the different writing styles of the authors.

This pilot study showed the feasibility of the recognition of Bushman characters using a SVM with maximum recognition accuracies of approximately 80% and 72% for authors 1 and 2 respectively. This pilot study, however, made use of a neatly re-written version of a Bushman story with a limited character set, thereby limiting the generalisation of these findings. Nonetheless, the findings still suggested that it was possible to automatically distinguish among the different character classes, even with their complex diacritics.

3.2 Pilot Study II: Text Line Recognition Using a Hidden Markov Model

The first pilot study recognised characters in a neatly rewritten version of one of the Bushman stories and achieved a best recognition accuracy of approximately 80%. The shortcoming in the initial pilot study was that it made use of user created data and only used a limited character set. This second pilot study was an attempt to overcome some of these shortcomings by investigating the feasibility of text line recognition of real data using a HMM.

3.2.1 Implementation

The system included a preprocessing step in which the pages in the Bleek and Lloyd Collection were binarised and the lines of text in the pages segmented using a horizontal projection profile based on the number of foreground-background transitions (Marti and Bunke, 2001). The segmented text lines were transcribed using the custom tool built for capturing Bushman text and encoded using the custom LaTeX-based encoding (see Chapter 4).

A recognition system based on a left-to-right continuous HMM with 12 emitting states and 16 Gaussian components was built. HMMs were trained for each of the character/diacritic combinations and for the characters ignoring diacritics and then combined to create word level models. Features were extracted using a sliding window approach where 9 geometrical features (Marti and Bunke, 2001) (see section 5.3) and the first 3 coefficients (excluding the zero-frequency component) of the Discrete Cosine Transform (DCT) (Ahmed et al., 1974) (see Section 5.7) were extracted for every window. The system was tested using each of these features independently as well as in combination.

3.2.2 Evaluation

74 pages from one of the notebooks in the Bleek and Lloyd Collection were used in this pilot study. The pages were automatically thresholded and then segmented into 995 separate text lines that contained a total of 336 different character/diacritic combination classes. Of these 995 lines, 296 were excluded from the training and testing phases due to noise or bad segmentation and one sample was excluded because it could not be fully represented using the LaTeX encoding used in this study. The system was evaluated using 10-fold cross validation (see Section 6.1.3) using the remaining 698 samples, which were randomly allocated to 10 folds. The results reported are the averages of the results of the 10 folds.

Three experiments were conducted as part of the evaluation. In the first experiment, HMMs were trained for each character/diacritic combination and character/diacritic combinations were recognised. In the second experiment HMMs were trained for each character/diacritic combination and diacritics were ignored during recognition, i.e. if the base characters were correctly recognised then it was marked as correct regardless of any diacritics. In the third experiment, HMMs were trained using transcriptions that ignored diacritics and recognition was also performed ignoring diacritics.

The recognition accuracies, as defined by Equation 6.2 (which is not a percentage), are shown in Figure 3.5. As can be seen from the figure, the recognition accuracy for the character/diacritic combinations in Experiment 1 was the lowest, averaging at 43.84. It is speculated that the main reason for this is the large number of combinations of characters and diacritics that are possible and the lack of training samples for many of the models. The results from Experiment 2 showed that by ignoring the diacritics in the recognition process, the recognition accuracy increased by about 10. Lastly, the recognition accuracy for Experiment 3 was 49.69, thereby demonstrating that ignoring diacritics during training and recognition only leads to a small improvement in recognition accuracy. This finding suggests that the use of a multilayer classifier may not be viable since, even if the first layer ignored diacritics during recognition, the input to the second layer, which would be responsible for differentiating among diacritics, would be of poor quality.

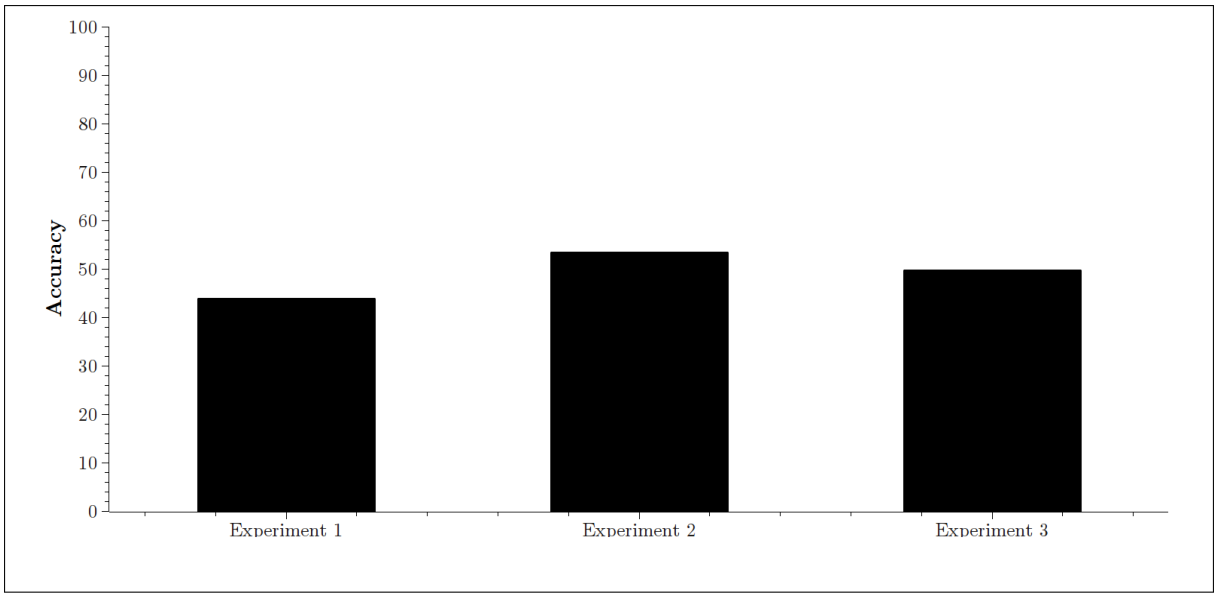


Figure 3.5: Results of the second pilot study for Bushman text line recognition

This second pilot study attempted to improve on the first by working with data from the notebooks instead of user created data. These texts contain more complex diacritics than the first pilot study, which makes the recognition process difficult due to the number of character/diacritic combinations that exist. The effect of this complex representation is that, for the majority of cases, there are fewer than 3 training samples. It was shown that an improvement could be achieved when diacritics were ignored. However, the diacritics play an important role in the Bushman languages, thus making this an infeasible approach.

The findings of these pilot studies showed that the recognition of Bushman texts is possible. However, they also suggested that additional techniques could be employed in order to improve recognition accuracy.

Chapter 4

Corpus Creation

In Chapter 2 the background to this study was discussed and, included in this discussion, was a description of some corpora that exist for handwriting recognition. A corpus for handwriting recognition usually contains a number of handwriting samples and their associated transcription and is used both for supervised learning, in which models are trained using the handwriting samples and their associated transcriptions, as well as for evaluating accuracy in which the recognition output is compared to the known transcriptions of text. Corpora already exist for many well-understood and well-studied languages, some of which have already been discussed in Chapter 2. However, in the case of historical texts, it is often necessary to create new corpora since suitable corpora for recognition usually do not exist.

This chapter describes the creation of a handwriting recognition corpus for Bushman languages. The corpus created represents a gold standard of transcriptions of the Bushman texts that appear in the Bleek and Lloyd Collection. In this context, a gold standard refers to the best available set of transcriptions, which, in an absolute sense, may or may not be a true representation of the text. This chapter begins by describing the custom method that was developed as part of this study to allow for the Bushman text to be represented electronically. This is followed by the description of a tool that was created to allow for pages of Bushman text to be segmented and transcribed and that was used in a series of workshops that were run as part of this study in which the corpus was created.

4.1 Bushman Text Representation

The Bushman text is complex due to the diacritics that appear above, below, and above and below characters and that can span multiple characters. Since the script used to represent the Bushman text cannot be represented using Unicode, it was necessary to create a custom solution. This custom solution involved the use of LaTeX and the TIPA package (Rei, 1996), which allows for the processing of International Phonetic Alphabet (IPA) symbols in LaTeX. The TIPA package does not support many of the diacritics that appear in the Bushman text by default but does allow for the creation of custom macros that can be used to create nested and stacked diacritics that are similar to those found in the Bushman text. Thus an encoding was developed that took advantage of this fact.

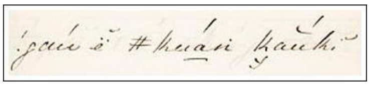
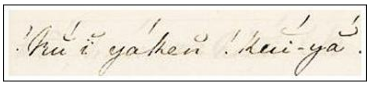
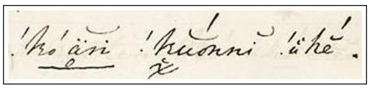
In the encoding that was developed, each Bushman diacritic is represented using a backslash (\) followed by a command that specifies the diacritic. For example, `\uline{}`

represents a U-shape with a line above it. In order to add diacritics to characters, the characters are enclosed within the braces of the diacritic command. For example, `\uline{a}` would produce ă̍. It is possible to attach a diacritic to multiple characters by enclosing multiple characters within the braces of the diacritic command, as in `\uline{ba}`. The encoding also allows for diacritics so be stacked and nested, as can be seen in `\cover{\onedot{\twodots{a}}}` which produces ä.

Using the custom macros the Bushman text can be encoded and an approximate visual representation, which is limited by the drawing capabilities of the TIPA package, can be created. Future work includes the possibility of creating a custom font for Bushman script. These encodings are not commutative and the meaning and visualisation depends on the order of the encoding operators, i.e. `\dialine{\onedot{a}}` produces ȧ whereas `\onedot{\dialine{a}}` produces ȧ. This non-commutative nature is important since it affects the order in which the diacritics appear, which is desirable from a preservation perspective. It is possible that the same diacritics appear in the text in different stacking orders, thus suggesting that the stack order potentially affects the meanings of the words themselves.

Table 4.1 shows how Bushman text can be encoded using LaTeX and the custom TIPA macros and also shows the approximate visual representation. The full set of the encodings of all supported Bushman diacritics and their approximate visualisations are provided in Appendix A.

Table 4.1: Encoding and visual representation of Bushman text using custom TIPA macros

Text Line	Encoding	Visual Representation
	!ga\dialine{u} \twodotsu{e} \texthash{}ku\barblinet{a}n \ybelow{k}a\uline{u}k\u{i}	!gaủ ĕ̈ #kua̲n ḳaủkĭ
	!k\uline{u} \twodotsu{i} y\dialine{a}ke\u{n} !ku\uline{i}-y\uline{a} .	!kŭ ĭ̈ yȧkeň !kuĭ-yă .
	!k\barbelow{\dialine{o} \circbtwodotsa{a}\onedot{n}} \xcbelow{k}\uline{uo}nn\u{i} !\uu{u}h\uline{e} .	!kȯăṅ ḳ ŭonnĭ !ŭhĕ .

4.2 A Specialised Tool for Creating the Corpus

A specialised tool called xȯä'xȯä, which means "to write" in the |xam Bushman language, was developed to create a corpus of Bushman texts. xȯä'xȯä is an AJAX (Mahemoff, 2006) Web-based tool that allows multiple users to segment pages and transcribe text by making use of automatic algorithms as well as manual user interaction. The xȯä'xȯä workflow begins with text selection in which Bushman text is selected and this is followed by preprocessing of the Bushman text. Segmentation of lines and words then takes place and, finally, text lines are transcribed. An overview of the steps involved when using xȯä'xȯä is shown in Figure 4.1 and described in the sections below.

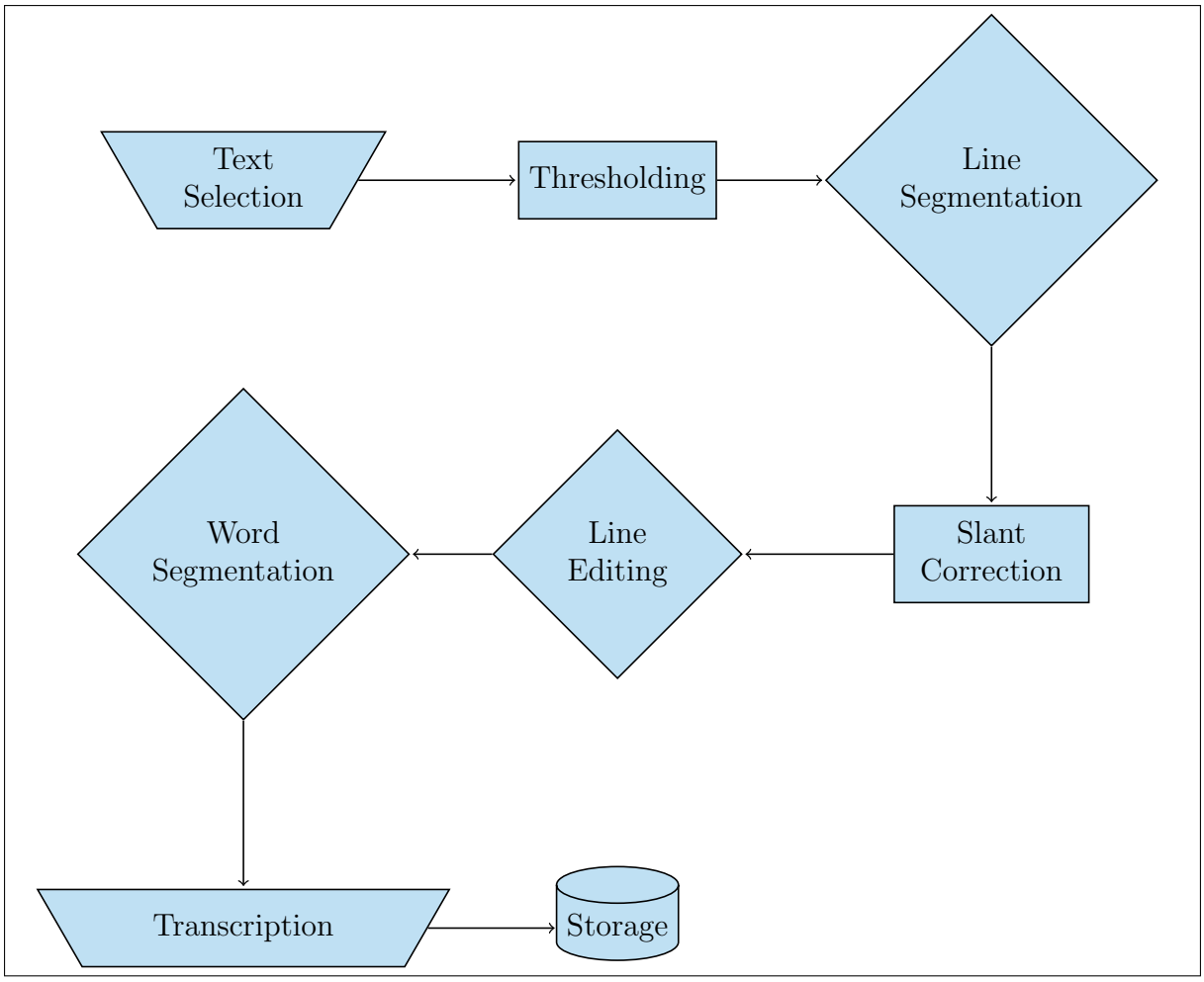


Figure 4.1: Steps involved during corpus creation

In addition to the core functionality of segmenting and transcribing text, xö̧a̧'xö̧a̧ also acts as a job management and monitoring tool. Each user has an account on the system and xö̧a̧'xö̧a̧ randomly assigns segmentation and transcription jobs to users and keeps track of the jobs completed by each user. By keeping track of the jobs completed by each user, the system makes it possible to evaluate each user in terms of their efficiency and the quality of the data they produce. Furthermore, users are provided with feedback on the amount of work that they have done relative to other users, which could possibly act as a motivating factor.

xö̧a̧'xö̧a̧ will be discussed in detail in this section, starting with a description of how the text is selected and the preprocessing steps that take place and then a description of the line and word segmentation processes. Thereafter, the component that allows for Bushman text to be transcribed is discussed.

4.2.1 Text Selection

In many cases, notebook pages in the Bleek and Lloyd Collection contain both English text and Bushman text alongside each other. Since the focus of this study is on the creation of a corpus of Bushman text, the first step is to select the Bushman text using a cropping tool. As can be seen in Figure 4.2, there is a clear separation between the

English text, which appears on the left side of the page and the Bushman text, which appears on the right side of the page. This arrangement of English and Bushman text is generally consistent across the collection and the writers used in this study, though there are some exceptions. The user is able to use a rectangular selection tool to draw a box around the Bushman text, as can be seen in Figure 4.2, which is then automatically cropped. In cases where the page contains no Bushman text, the user is able to click the *No Segmentation* button and the page will be ignored.

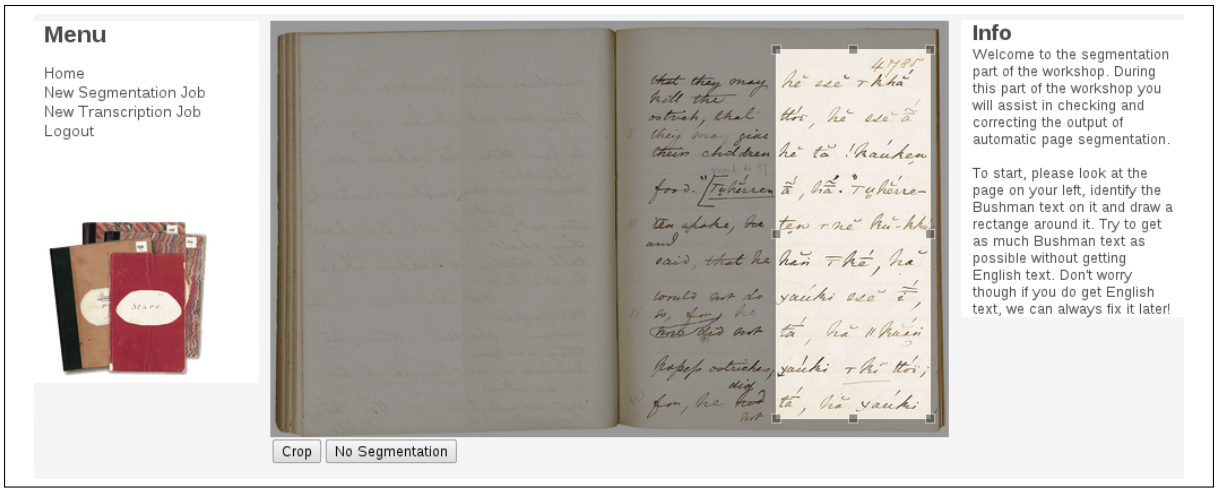


Figure 4.2: Interface showing how Bushman text is selected and separated from English text

4.2.2 Thresholding

Thresholding is an image processing technique in which a colour or greyscale image is converted to a binary image consisting of only black and white pixels. After the Bushman text has been selected and cropped, thresholding of the cropped area takes place. The thresholding used in this study is based on a local adaptive approach (Sezgin and Sankur, 2004) using a sliding window. In this approach, the centre pixel for each window placement is considered and compared to the mean for the window. If the centre pixel is greater than the mean by a bias value then it is considered to belong to the foreground, otherwise it is considered to belong to the background. Thresholding of the image requires no input from the user. Figure 4.3 shows an example of what thresholded text looks like.

4.2.3 Segmentation

Once a page of Bushman text has been preprocessed by cropping and thresholding it, the next step in the corpus creation workflow involves segmenting the page. Pages of Bushman text are first segmented into individual lines, which then have their slant corrected and can be edited by the user. Once this is completed the individual text lines are segmented into words. Both the line segmentation and the word segmentation operations are semi-automatic in that they make use of automatic algorithms to identify candidate segmentation points. These candidate segmentation points can either be approved by the user with no modification or the user can make changes by adding, removing and moving segmentation points.

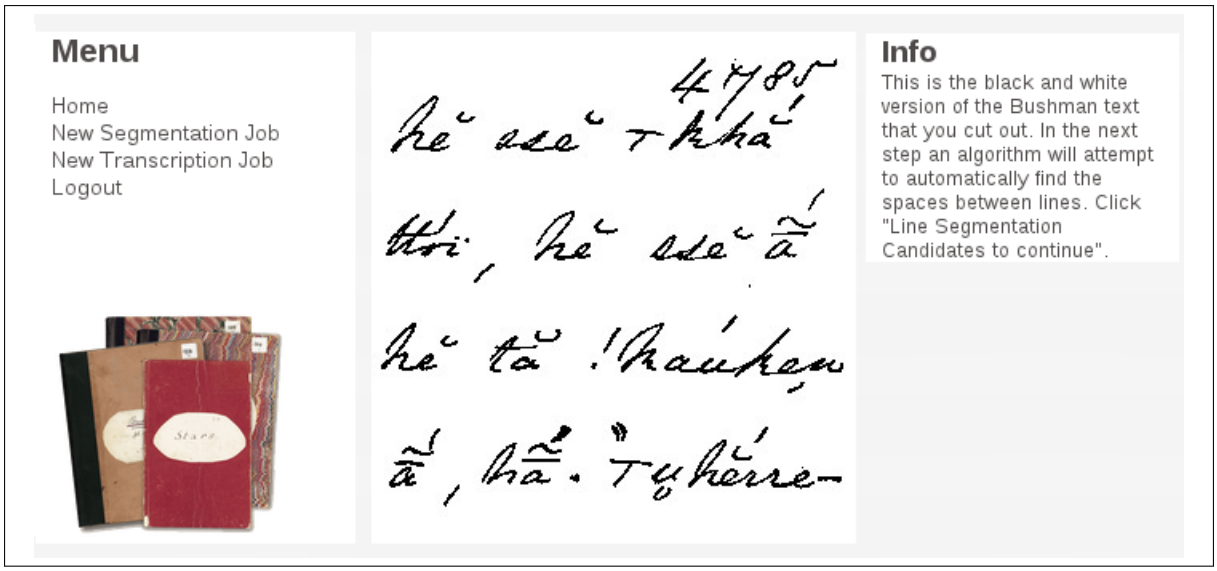


Figure 4.3: Interface showing thresholded Bushman text

4.2.3.1 Line Segmentation

Line segmentation involves segmenting a page into its individual text lines. To do this, an approach based on the horizontal projection profile is used (Marti and Bunke, 2001). In this approach, each horizontal pixel-row is analysed to determine the number of foreground-background transitions and a horizontal projection profile is created for the image. The projection profile models the distribution of pixels along each row of the image, where rows that contain text have relatively high values in the projection profile and those that contain no text have relatively low or zero values in the projection profile. In the projection profile, minima represent the spaces between individual lines; however, noise and the spaces between diacritics and their base characters can create false minima. For this reason, the projection profile is smoothed using a Gaussian filter. An example of a projection profile before and after Gaussian smoothing is shown in Figure 4.4, which clearly shows how smoothing eliminates false minima.

Minima in the smoothed projection profile represent the candidate spaces between lines. However, before the page is segmented into lines, the candidates are presented to the user, who is given the option to move the candidate line segmentation locations by dragging them with the mouse and add or delete candidate line segmentation locations. Figure 4.5 shows an example of the interface when the candidate line segmentation locations are automatically identified by the line segmentation algorithm.

4.2.3.2 Slant Correction

Different authors have different styles of writing and one source of variation among different authors comes from the angle of slant with which they write. Text normalisation can be used to minimise the variation among different authors and allows for a handwriting recognition system that is more robust when recognising the handwriting of multiple authors. One technique for text normalisation is slant correction, in which the slant of text is corrected so that the text appears upright. Slant correction is performed by calculating the slant of the text in a text line and then shearing the text line to correct the slant.

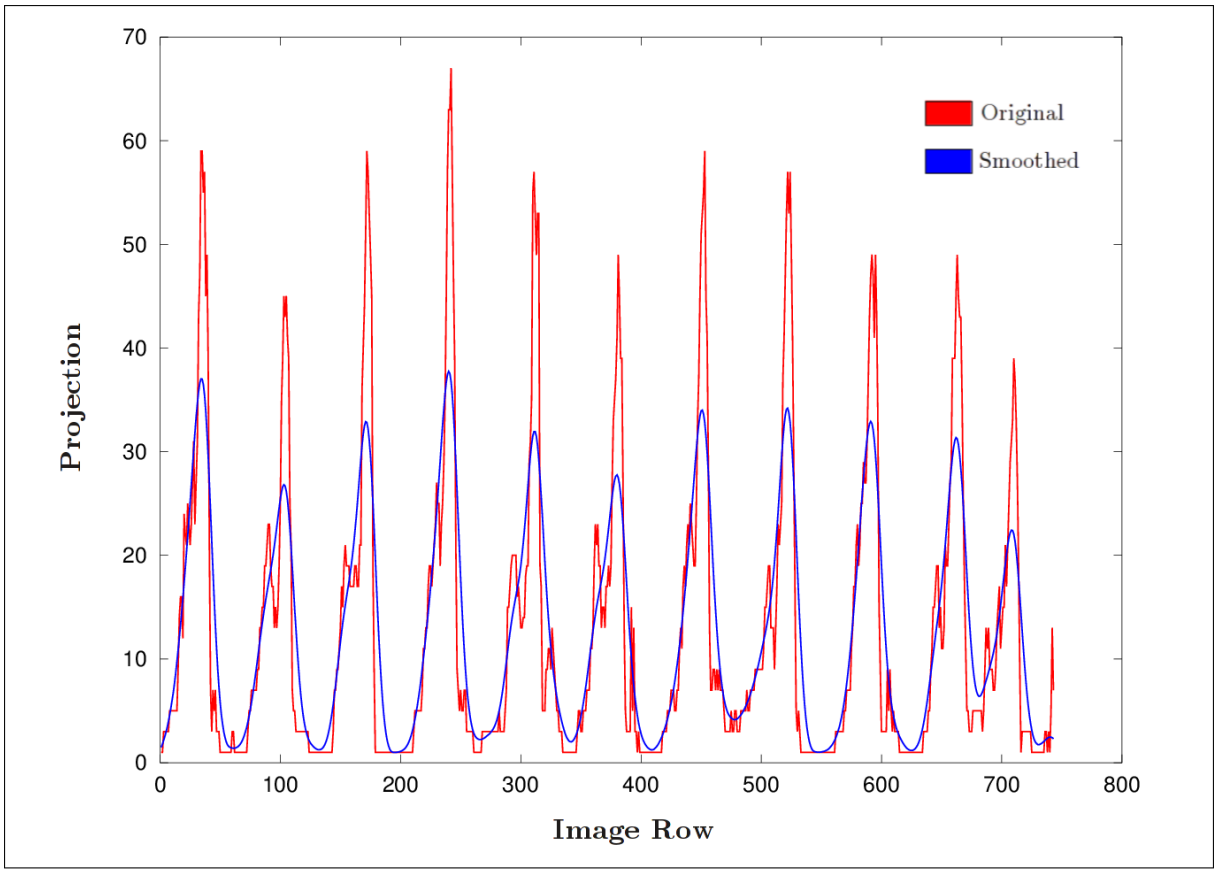


Figure 4.4: Projection profile for line segmentation before and after Gaussian smoothing

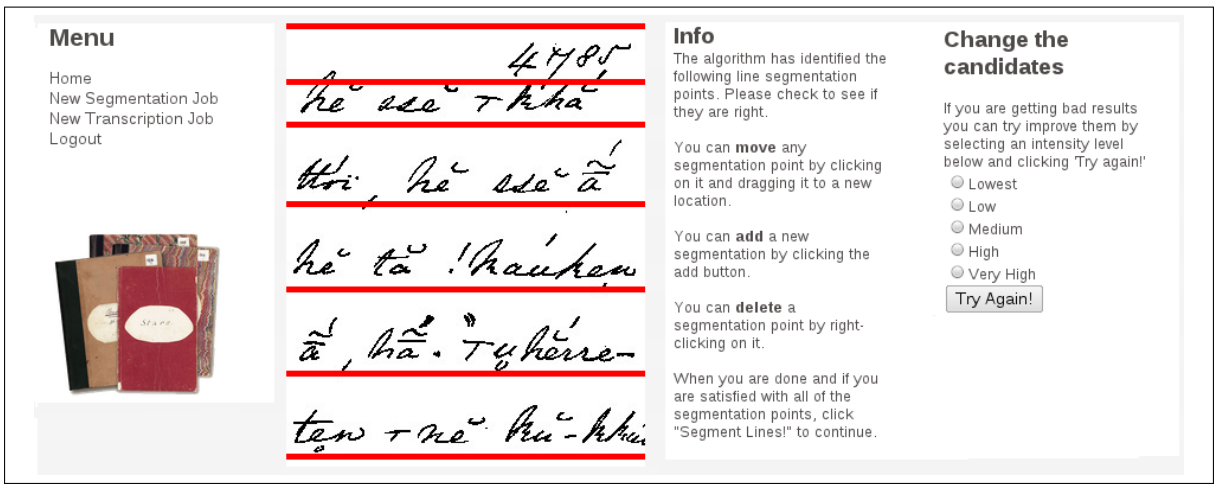


Figure 4.5: Interface allowing users to view line segmentation candidates and change them

The technique used in this study for slant correction is the same as that used by Pastor et al. (2004) and is based on a vertical projection profile technique.

In a vertical projection profile technique, the image is sheared at a number of discrete angles and the vertical projection profile at each sheared angle is calculated. The projection profile that gives the most variation is considered as being the straightened image since less slant in an image leads to more variation in the vertical projection profile, largely as a result of the effect of the ascenders and descenders (Pastor et al., 2004). Thus, once the angle at which the most variation in the projection profile has been calculated, the image

can be sheared and the slant corrected. Figure 4.6 shows an example of a text line before and after having its slant corrected.

(a) Original text line (b) Slant corrected text line

Figure 4.6: A text line and its slant corrected version

4.2.3.3 Line Editing

Once the individual text lines have been segmented and the slant has been corrected, they are presented to the user for further editing before word segmentation takes place. The user is given the option to manually edit each of the individual lines by either further cropping it, re-applying the slant correction operation if they feel that the result of the previous slant correction operation was unsatisfactory or even deleting the text line entirely.

4.2.4 Word Segmentation

Once the user is satisfied with the individual text lines, the next step involves word segmentation. Word segmentation is performed on each of the individual text lines segmented in the previous step and the word segmentation algorithm used in this study is based on the distances between CCs (Makridis et al., 2007). The first step in segmenting a text line into its individual words involves identifying and labelling the CCs (Shapiro and Stockman, 2001). CCs are then sorted by their x-coordinates and the horizontal distances between every pair of adjacent CCs is calculated. When CCs overlap vertically, the distance between them is set to 0. For instance, this can occur when diacritics appear above a character but do not form part of the same CC. If the distance D between two adjacent CCs C_i and C_{i+1} is greater than the threshold T then C_i and C_{i+1} are considered as belonging to two separate words and the central point between them is marked as a candidate word segmentation point; otherwise, C_i and C_{i+1} are considered as belonging to the same word. The threshold T is calculated as:

$$T = \frac{\sum_{i=1}^{n-1}(Distance(C_i, C_{i+1}))/2}{n}, \tag{4.1}$$

where n is the number of CCs in the image. This function for calculating the value of T was empirically found to produce good results. As was the case with line segmentation, the candidate segmentation points are presented to the user and the user has the option to move segmentation points as well as add or delete segmentation points. Once the user is satisfied with the segmentation points, each line will be segmented into its individual words. Figure 4.7 shows an example of the word segmentation interface.

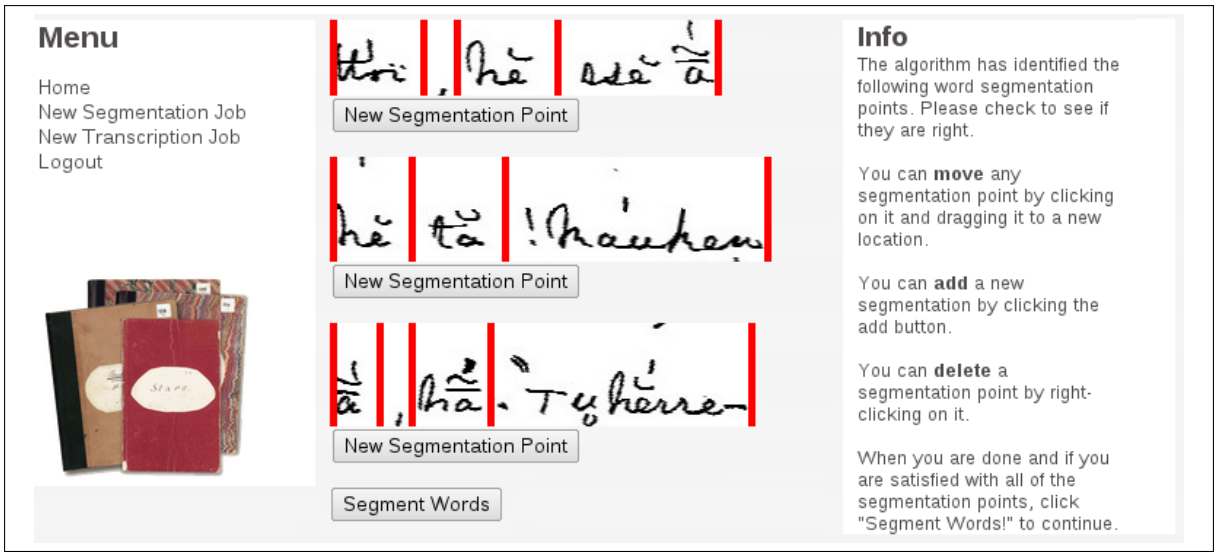


Figure 4.7: Interface allowing users to view word segmentation candidates and change them

4.2.5 Transcription

The text that appears in the notebooks needs to be captured in a machine readable format, which is described in Section 4.1, in order for future processing to take place. xǒä'xǒä supports transcription through a Web interface, which is shown in Figure 4.8. There are several components that make up this interface. The first of these is the text line that is to be transcribed, which appears at the top of the interface and is randomly selected from all the text lines in the corpus. Below the image of the line to be transcribed is a text input box that allows the user to type in the characters that appear in the text line. In order to add diacritics, the user highlights the text to which the diacritics are to be added in the text input box, and then clicks on the appropriate diacritic in the right panel of the interface. This has the effect of enclosing the highlighted text in the diacritic using the encoding method described in Section 4.1. The user is able to view the results of the encoding by clicking on the *Convert to Latex* button, which then shows the LaTeX rendering of the text at the bottom of the interface.

The interface caters for diacritics that appear above text, below text and above and below text. The diacritics that it supports were pre-determined by scanning through a subset of the notebooks in the Bleek and Lloyd Collection and noting the different diacritics encountered. However, new diacritics are constantly being discovered while working with the collection and thus it is possible that some diacritics might be encountered that the interface does not support. If this occurs, the user is able to click on the *No Representation* button and the line will be marked as not being supported by the interface. At a later stage these lines can be reviewed and the previously unseen diacritics that they contain can be added to the interface. The *No Representation* button can also be used when the user feels that a text line is not suitable for transcription, such as when the text line contains English text, when words have been crossed out or when large amounts of noise occurs in the image. When the user is satisfied with their transcription, they can click the *Save Transcription* button and the transcription will be saved.

The steps in the use of xǒä'xǒä are summarised in Table 4.2 and the level of user interaction is indicated. The use of this tool is, however, not limited to use with Bushman text but

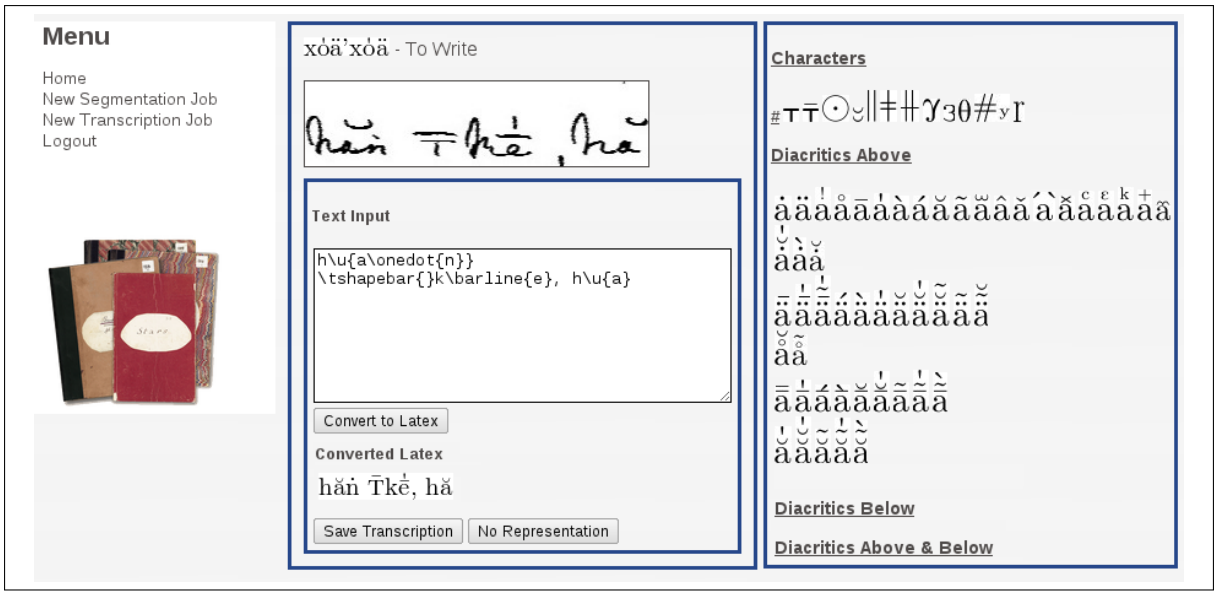


Figure 4.8: Interface for transcribing the Bushman text

could potentially be adapted to suit other historical texts that cannot be represented using Unicode. xǒäˈxǒä was used in this study to create a handwriting recognition corpus for Bushman languages and its use in a series of workshops is discussed in the next section.

Table 4.2: Steps involved in corpus creation

Step	Operation	User Interaction Required
1	Text Selection	Yes
2	Thresholding	No
3	Line Segmentation	Optional
4	Slant Correction	No
5	Line Editing	Optional
6	Word Segmentation	Optional
7	Transcription	Yes

4.3 Bushman Corpus Creation

A series of corpus creation workshops were held as part of this study in which students were recruited as data capturers and used xǒäˈxǒä to create a handwriting recognition corpus for Bushman languages. None of the data capturers spoke or understood Bushman languages and were primarily undergraduate students, though some post-graduate students also participated. For the workshops, 900 pages from the Bleek and Lloyd Collection were randomly sampled and added to xǒäˈxǒä as segmentation jobs. In this study, handwriting recognition was investigated for single and multiple authors and thus the pages that were sampled contained the writing of two authors. Three corpus creation workshops were held with groups of different sizes. The workshops and the data collected are briefly described below.

4.3.1 First Corpus Creation Workshop

The first corpus creation workshop took place on a Saturday and lasted for 7 hours. Twenty-nine data capturers, the majority of whom came from a computer science or electrical engineering background, participated in this workshop. The purpose of the first corpus creation workshop was to segment the pages of Bushman text and then transcribe the text. The workshop began with an introduction to the problem and a discussion of the goals of the workshop and the anticipated outcome and use of the data. Given the complexities of the Bushman script, the data capturers were encouraged to talk to one another and work collaboratively when segmenting and transcribing the text.

4.3.1.1 Segmentation

The segmentation part of the first corpus creation workshop began with a short demonstration of how segmentation was to be performed. Segmentation jobs were randomly allocated to users by the job management system, which kept track of who the jobs were allocated to and what changes each individual user made. Text lines were the standard input method for capturing the Bushman text and therefore the text lines that each page was segmented into were added to a list of transcription jobs.

In total there were 900 segmentation jobs, of which 729 were completed and segmented into 7950 text lines. There are a number of reasons why the other 171 jobs might not have been completed, such as them not containing Bushman text.

4.3.1.2 Transcription

Transcriptions were captured for some of the individual text lines segmented during the segmentation part of the workshop. The data capturers were given a demonstration of how to perform transcription and some known special cases that arise in the Bushman text were highlighted. The data capturers were encouraged to work collaboratively to determine what the characters were and the importance of consistency was stressed.

Of the 7950 transcription jobs created during the previous step, 1548 were completed and 459 were marked as having no representation. The text lines that were marked as having no representation were investigated in preparing for the second and third corpus creation workshops (discussed below) and, where appropriate, new diacritics were added to the xo̱ä'xo̱ä interface and the lines re-inserted as transcription jobs.

There are two reasons for 5951 transcription jobs not being completed. The first is that there was insufficient time to complete all of the jobs in one sitting and the other is that some users were more efficient than others. Since the goal of the workshop was to create a high quality corpus, the contributions of each user were evaluated in terms of accuracy and efficiency and the data capturers who produced the best quality data most efficiently were invited to participate in the second and third corpus creation workshops.

4.3.2 Second and Third Corpus Creation Workshops

Two additional corpus creation workshops were held after the first and each lasted approximately 6 hours and the data capturers who were the most efficient and accurate

during the first corpus creation workshop were recruited for these additional workshops. Since all of the segmentation jobs had been completed during the first workshop, this workshop focused solely on the transcription task.

A total of 1700 text lines were transcribed during these two workshops. In addition to this, approximately 10% of text lines were re-transcribed in order to allow for the quality and consistency of the corpus to be assessed. 10% of the text lines from the first workshop were also re-transcribed in order to gain an idea of the quality of the data created during the first workshop.

4.4 Corpus Preprocessing and Transformation

Even though the data capturers were encouraged to work collaboratively when transcribing the text, there were a number of complexities in the text that led to confusion. Some of these sources of confusion were noted during the workshops and others were discovered later during an analysis of the corpus. The result of this confusion is that it could lead to the same characters and diacritics being erroneously classified as being different by different data capturers. In addition to these sources of confusion, there are also a number of erroneous encodings of the text that make text processing difficult, such as double quotation marks appearing in the text. Therefore, preprocessing or "cleaning" of the corpus was used to remove the erroneous encoding as well as to minimise the effects of the confusion that arose during transcription. In this section, the term *symbol* is used to refer to any single character, character/diacritic combination or space while the term *class* is used to refer to the classification of a symbol or word.

The preprocessing and transformation of the corpus creates new versions of the corpus by removing conflicting transcriptions that arose as a result of confusion and by cleaning the corpus for easy text processing. For this reason, and for further experimentation, each version of the corpus was given a new name to make it identifiable. To start, the original, unedited corpus is referred to as Corpus-Original or Corpus-O and each version of the corpus that was created builds on the previous version.

In this section, the preprocessing of the corpus that results in the creation of the new versions of the corpus will be discussed. Beginning with the original corpus, Corpus-O, the transformations required to allow for text processing to take place and which resulted in the creation of Corpus-Functional, or Corpus-F, are described. As mentioned, there were sources of confusion about the data that arose and were highlighted during the workshops. Thus, the next section describes the transformations to address these sources of confusion and which resulted in the creation of Corpus-Workshop or Corpus-W. Thereafter, an automatic analysis of Corpus-W was performed in order to identify characters and diacritics that were commonly classified as being different by different data capturers. The transformations to account for these different classifications by different data capturers led to the creation of Corpus-Analysed or Corpus-A. The transformations that take place are summarised in Figure 4.9.

4.4.1 Corpus-Original

Corpus-Original or Corpus-O is the unedited corpus created by the data capturers during the corpus creation workshops. This corpus contains the errors that could potentially

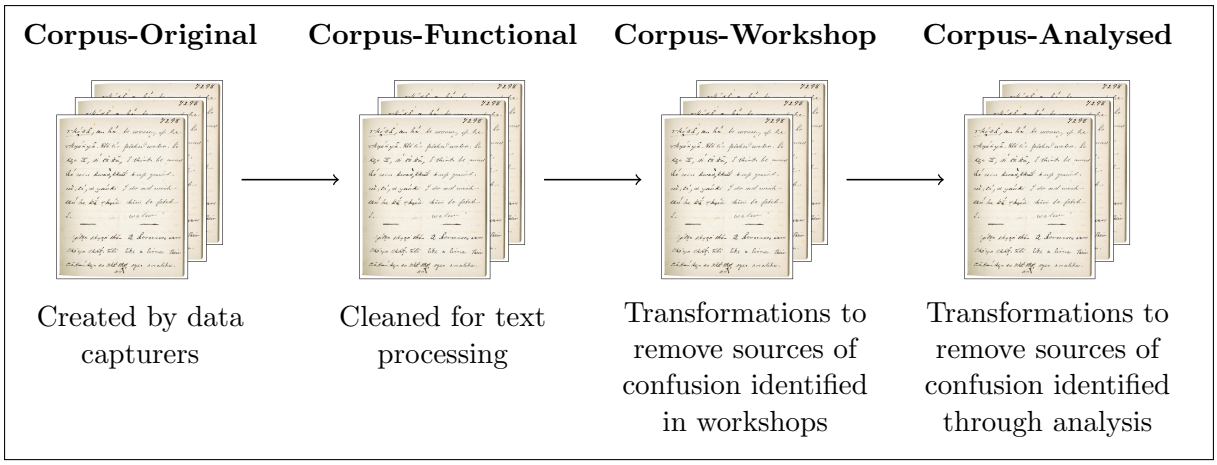


Figure 4.9: Corpus transformations that take place and result in the creation of new versions of the Bushman corpus

cause the software to function incorrectly as well as any other errors introduced by the data capturers.

4.4.2 Corpus-Functional

Corpus-Functional or Corpus-F builds on Corpus-O to create a corpus that allows for text processing to occur without error, for instance, by escaping special characters. The following transformations were performed to create Corpus-F: *!* to `\excl{}`; # to `\texthash{}`; | to `\textbar{}`; and `"` to `''`.

4.4.3 Corpus-Workshop

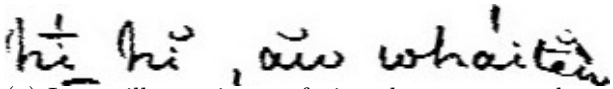
Corpus-F was created by performing transformations that allowed for text processing to take place. Corpus-F, however, does not account for any of the sources of confusion that arose from the data. There were five cases of confusion that became immediately obvious during the workshops and transformations were created to correct them. These sources of confusion were brought up several times by the data capturers. These five cases are discussed below and examples from the text are given. The outcome of these transformation rules is to create Corpus-Workshop or Corpus-W, which builds on Corpus-F by applying the transformations discussed below.

- **Comma and Period Placement**
  Figure 4.10 (a) shows an example of where confusion can arise when placing the comma during transcription. Intuition suggests that the comma should follow the first word, which in Figure 4.10 (a) would result in a transcription that appears as *h̲i̲ hi̲, ău whȧitěǹ*. However, in the figure, the comma is clearly closer to the second word, which would result in a transcription that appears as *h̲i̲ hi̲ ,ău whȧitěǹ*. This confusion about placement of a comma occurred frequently during transcription and is corrected during the preprocessing, where all commas are set to appear directly after a word, with no additional spacing. The same correction is made for period symbols in the text.

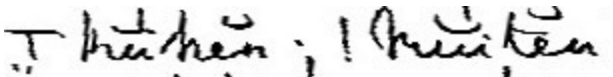
- **Semi-colons**
  Figure 4.10 (b) shows an example of a text line that appears to be a semi-colon. However, this could also be confused as being a comma with noise appearing above it. This was brought up by multiple data capturers several times during the workshop and it can be assumed that it was a legitimate source of confusion and that data capturers differed in their transcription of the character. Thus, to eliminate this confusion, all semi-colons that appear in the transcriptions are converted to commas.

- **Additional Spaces**
  Leading and trailing spaces appear in text lines because of the way that they are segmented and due to them not always being perfectly cropped. In addition to this, sometimes the spaces between words in a single text line differ enough to suggest that there may be double spaces separating some of the words. Figure 4.10 (c) shows an example of a leading space and large and small spaces between adjacent sets of words. Because data capturers may have chosen to represent these additional spaces in different ways, preprocessing is used to remove any additional spaces by removing leading and trailing spaces in transcriptions and by replacing two or more adjacent spaces with a single space.

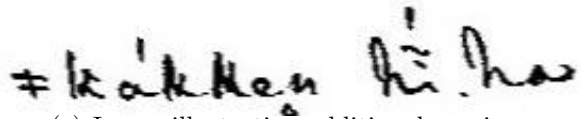
- **T vs τ**
  The first character in Figure 4.10 (b) shows an example of the τ symbol, which can easily be confused with a uppercase T, which rarely appears in the text. Though this was brought to the data capturers' attention, it is likely that τ was mistaken for an uppercase T on multiple occasions. Given the rare frequency of an uppercase T appearing relative to the common appearance of τ, all uppercase Ts in the transcriptions are converted to τs.



(a) Image illustrating confusion about comma placement



(b) Image illustrating semicolon and confusion between T and τ



(c) Image illustrating additional spacing

Figure 4.10: Examples of common sources of confusion from corpus creation workshops

The output of the above transformations is Corpus-W, which is based on the sources of confusion that were raised by the data capturers during the corpus creation workshops. An example of a text line transformation from Corpus-O to Corpus-W is shown in Figure 4.11.

Corpus-O:

```
" Th\uline{i} ; tk\u{u} ,hi    #ki |ka "
```

Corpus-W:

```
"\tshape{}h\uline{i}, tk\u{u} , hi \texthash{}ki \textbar{}ka"
```

Figure 4.11: An example of a text line transformation from Corpus-O to Corpus-W

4.4.4 Corpus-Analysed

Beyond the obvious sources of confusion from the workshop, there was also the possibility that cases existed where the data capturers largely disagreed about what certain symbols were. To identify these sources of confusion, the 10% of the collection that was re-transcribed was compared to the original transcriptions of the same text lines. The differences between the two transcriptions were analysed using the Levenshtein distance (Levenshtein, 1966) (see Section 4.5.1.1) and all substitutions that took place were noted. In the calculation of the Levenshtein distance, character and diacritic combinations were considered as being a single symbol. For this reason, the substitutions that took place were analysed both with and without the base characters. For instance, when analysing with base characters, the substitutions from `\diabar{a}` → `\dbar{a}` and `\diabar{b}` → `\dbar{b}` would be considered as substitutions for two different symbols due to the different base characters. However, when the substitutions were analysed without the base characters, `\diabar{a}` →`\dbar{a}` and `\diabar{b}` → `\dbar{b}` would be considered as two occurrences of the same substitution and suggest that the confusion arose due to the diacritics and not the base characters.

The frequency of each substitution was counted and the average number of substitutions was calculated. This was done for substitutions both with and without base characters. Substitutions that occurred with an above average frequency were then further analysed in order to investigate what proportion of the character's occurrences were substituted for the other. For instance, if `\ubar{}` occurred 10 times in the text and was substituted by `\diabar{}` 5 times, then the substitution proportion would be $\frac{5}{10}$ or 50%. If a character's substitution proportion with another character was above some level D then it was marked as a candidate for transformation, where $D = 100\%, 75\%, 50\%, 25\%$. Each transformation candidate was then individually analysed in order to determine if it was suitable for transformation to occur and, where applicable, the transformation was applied.

Separate corpora were created for each substitution proportion level D resulting in the creation of Corpus-Analysed-D, where $D = 100\%, 75\%, 50\%, 25\%$. For instance, Corpus-Analysed-50 or Corpus-A-50 is built on Corpus-W and transforms symbols where the substitution proportion from the original transcriptions to the re-transcribed versions of the text was at least 50%. The same occurs for other values of D.

Table 4.3 gives the transformations that occurred in order to create the various versions of Corpus-Analysed-D. In the table, when the transformation occurs regardless of base character, the transformation is shown as applied to a placeholder X; otherwise, when the transformation is character-specific, the base character is shown.

Table 4.3: Corpus-Analysed-*D* transformations

		Original		Transformed	
Corpus	Symbol	Encoding	Visual	Encoding	Visual
		`\onedotdbar{X}`	$\dot{\bar{X}}$	`\dialine{X}`	$\overset{\shortmid}{X}$
A-100		`\xbelow{\ubelow{X}}`	$\underset{\underset{x}{\smile}}{X}$	`\xbelow{X}`	$\underset{x}{X}$
		`\barbubart{X}`	$\acute{\underline{X}}$	`\barblinet{X}`	$\overset{\shortmid}{\underline{X}}$
A-75		`\ukappie{X}`	$\check{X}$	`\u{X}`	$\breve{X}$
		`\ubar{X}`	$\acute{X}$	`\dialine{X}`	$\overset{\shortmid}{X}$
A-50		`\dialine{k}`	$\overset{\shortmid}{k}$	`k`	k
		`\cover{i}`	$\overset{c}{i}$	`\u{i}`	$\breve{i}$
		`K`	K	`k`	k
		`\wover{X}`	$\overset{\smile}{X}$	`\u{X}`	$\breve{X}$
		`/`	/	`\textdoublepipe{}`	‖
A-25		`\hcirclebelow{X}`	$\underset{\smile}{X}$	`\circlebelow{X}`	$\underset{\circ}{X}$
		`\dialine{n}`	$\overset{\shortmid}{n}$	`\onedot{n}`	$\dot{n}$
		`\dbar{n}`	$\dot{n}$	`\onedot{n}`	$\dot{n}$
		`j`	j	`J`	J

4.5 Corpus Analysis

Table 4.4 summarises the quantity of data collected during the corpus creation workshops. In total, 7950 text lines from 900 pages in the Bleek and Lloyd Collection were segmented. Of these 7950 text lines a total of 3158 lines or 39.7% were transcribed. Of these 3158 lines, 302 or 9.6% were re-transcribed in order to allow for the quality of the data created to be assessed. In this section, the consistency within the corpus, as measured by the Levenshtein distance (Levenshtein, 1966) between the original transcriptions and the re-transcribed data, will be evaluated, followed by an analysis to determine the frequency count of symbols and words in the corpus.

Table 4.4: Summary of data collected during corpus creation workshops

	Lines Segmented	Transcriptions	Overlap
Workshop 1	7950	1458	137
Workshop 2	0	1000	98
Workshop 3	0	700	67
Total	**7950**	**3158**	**302**

4.5.1 Corpus Consistency

In addition to the quality of the corpus, the consistency between the different data capturers is important. Consistency is important for the training of recognition models since variation between interpretations of symbols negatively affects the accuracy with which these models can correctly recognise unseen data. Since the interpretation of the more complex symbols in the text is, to a certain extent, subjective, the consistency among transcriptions among the different data capturers was evaluated in order to gain insight into the extent to which the data capturers were consistent with their interpretations. Consistency in the corpus was measured based on the normalised Levenshtein distance (Yujian and Bo, 2007) between the original transcriptions and the re-transcribed versions of the same text lines. This section begins with a description of the Levenshtein Distance (Levenshtein, 1966), followed by an experiment in which the consistency of the corpus was evaluated.

4.5.1.1 The Levenshtein Distance

The edit distance between two strings is the number of edit operations required to transform one string into another (Wagner and Fischer, 1974). The Levenshtein distance is one of the most common measures used to calculate the edit distance between strings and is used in this study to measure the consistency between the original text lines that were transcribed and the re-transcriptions of 9.6% of those text lines. The edit operations that the Levenshtein distance takes into consideration are substitutions, deletions and insertions (Levenshtein, 1966).

Formally, for an alphabet Σ, Σ^* is the set of finite strings in Σ and $\lambda \notin \Sigma$ is the null string. The string $X \in \Sigma^*$ is given by $X = x_1, x_2, ..., x_n$, where x_i is the ith symbol in X and n is the length of X given by $|X|$. $a \rightarrow b$, $a \rightarrow \lambda$ and $\lambda \rightarrow b$ represent the substitution,

deletion and insertion edit operations respectively. $O_{X,Y} = O_1, O_2, ..., O_k$ represents a set of edit operations to transform X into Y. A weight function γ is the cost of an edit operation where $\gamma(O_{X,Y}) = \sum_{i=1}^{k} \gamma(O_i)$ produces a non-negative real number that is the cost of the transformation. The Levenshtein distance (LD) is then given by:

$$LD(X, Y) = min\{\gamma(O_{X,Y})\} \tag{4.2}$$

The Levenshtein distance is a number that represents the minimum cost or number of edit operations that are required in order to transform one string into another. This number is normalised as the Normalised Levenshtein distance (NLD) presented by Yujian and Bo (2007), which gives a measure of similarity in the range [0-1] and is based on the following equation:

$$NLD(X, Y) = \frac{2 \cdot LD(X, Y)}{\alpha \cdot (|X| + |Y|) + LD(X, Y)}, \tag{4.3}$$

where $\alpha = max\{\gamma(a \to \lambda), \gamma(\lambda \to b)\}$.

4.5.1.2 Measuring Levenshtein Distance in Bushman Text

The encoding of the Bushman text, as described in Section 4.1, uses a multiple character encoding to represent a single symbol. For instance, `\ubar{a}` uses 8 characters to represent the single á Bushman symbol. To account for this, the encoding of character/diacritic combinations are considered as a single symbol. For example, $\backslash uline\{a\}$, $\backslash uline\{\backslash dialine\{a\}\}$ and a are all considered as single symbols and a difference between them and any other symbol will increase the Levenshtein distance by the weight function γ.

In the next section, an experiment that was conducted to investigate the corpus consistency is presented.

4.5.1.3 Experiment: Corpus Consistency

- **Purpose**
  - To gain insight into the consistency of the transcriptions of the text in the corpus.
- **Procedure**
  - The Levenshtein distance is used to measure the similarity between the original transcriptions and the re-transcribed versions using the measure of similarity in Equation 4.3.
  - The average similarity for the data gathered during each workshop is calculated as well as the average for the entire corpus based on the relative weightings of the amount of data captured during each workshop, where the relative weightings are 0.46, 0.32 and 0.22 for workshops 1, 2 and 3 respectively.
  - The above procedure is repeated for Corpus-O, Corpus-F, Corpus-W, Corpus-A-100, Corpus-A-50, Corpus-A-75 and Corpus-A-25.

Table 4.5: Consistency in the corpus

Corpus	Workshop 1	Workshop 2	Workshop 3	**Weighted Average**
Corpus-O	69.05%	76.27%	81.46%	**74.19**%
Corpus-F	69.05%	76.27%	81.46%	**74.19**%
Corpus-W	70.14%	77.33%	83.02%	**75.38**%
Corpus-A-100	70.18%	77.30%	83.78%	**75.56**%
Corpus-A-75	70.37%	77.30%	84.04%	**75.71**%
Corpus-A-50	70.97%	78.44%	84.21%	**76.38**%
Corpus-A-25	71.66%	79.05%	84.61%	**76.98**%

- **Results** The similarity between the original transcriptions and the re-transcriptions is shown in Table 4.5.
- **Discussion**
  - Table 4.5 shows that, on average, there was more consistency among transcriptions in the second and third workshops. This is to be expected since the data capturers who were recruited for the second and third workshops were the most efficient and accurate data capturers. Another reason for the lower consistency in the first workshop could be that the data capturers who did the re-transcription of the data from the first workshop were only a subset of the data capturers involved in the first workshop. This is different to the case for the second and third workshops, in which the same group of data capturers did the re-transcriptions.

    To test for differences in consistency between the different corpora, an Analysis of Variance (ANOVA) test was conducted to test for differences between means of the unweighted Levenshtein distances between all original transcriptions and re-transcriptions. The ANOVA test was insignificant at $p = 0.1$, thereby indicating that there was no statistical difference in the mean consistency between corpora.

    The large inconsistencies within the corpus are likely to have a negative effect on the robustness of learning models trained using this corpus and the effect of this is discussed in Section 7.3.5.

4.5.2 Corpus Distribution

It is well known that there is a positive relationship between the amount of training data used in handwriting recognition and the recognition accuracy that can be achieved. Corpus distribution, in this context, refers to the frequency with which symbols and words occur within the corpus. Insight into the corpus distribution is valuable as it can assist in explaining recognition results. In this section, two experiments are described in which the distribution of symbol classes and words classes within the corpus are investigated.

4.5.2.1 Experiment: Symbol Distribution

- **Purpose**
  - To identify the frequency with which the different symbols occur in the corpus.
- **Procedure**
  - The number of occurrences of each symbol class in all transcriptions was counted.
  - The above procedure is repeated for Corpus-O, Corpus-F, Corpus-W, Corpus-A-100, Corpus-A-50, Corpus-A-75 and Corpus-A-25.
- **Results** The symbol distribution is shown in Figure 4.12 and the dominant frequency range of 1-10 is shown in greater detail in Figure 4.13.

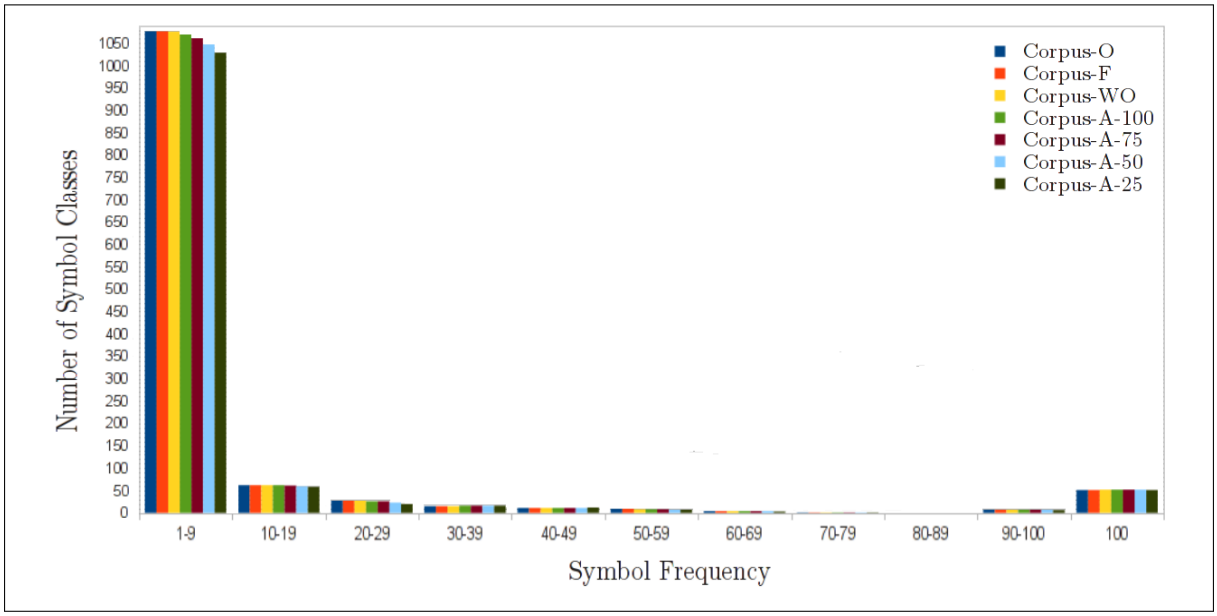


Figure 4.12: Symbol frequency count in corpus

- **Discussion**
  - As can be seen in Figure 4.12, a large number of symbol classes appear 1-9 times in the corpus. This is further investigated in Figure 4.13, which shows that a large number of symbol classes have only a single sample. In fact, 56.27% of symbol classes only have one sample and 85.30% of symbol classes contain fewer than 9 samples. Since the evaluation of the recognition models used in this study is performed using cross validation, this suggests that none of the symbols with only one sample can be correctly recognised since the single sample can only be in one of the training or testing sets. This, however, needs to be considered within the context of the size of the corpus. There is a total of 33480 individual symbols in the corpus and approximately 700 symbol classes that only have one sample. These 700 symbol classes with one sample only represent 2.10% of all of the symbols in the corpus and those with fewer than 10 samples represent 3.17% of the corpus.

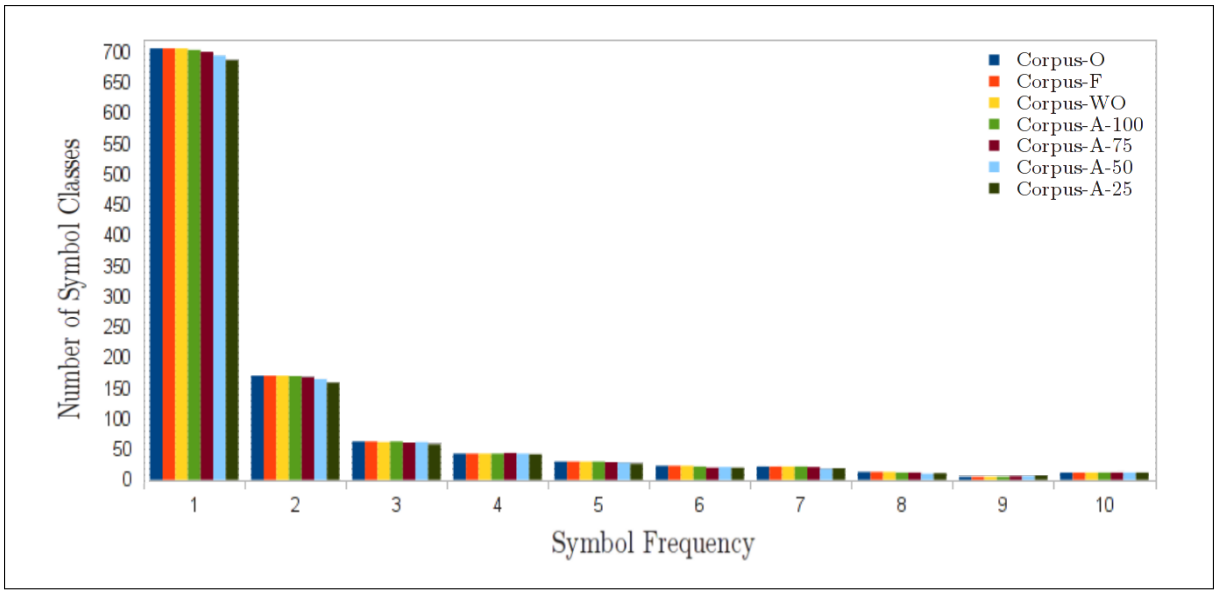


Figure 4.13: Symbol frequency count in corpus for frequency range 1-10

Table 4.6 shows the average number of times that each symbol class occurs in the text for the various corpora, with an average of 25.87. Table 4.6, along with Figures 4.12 and 4.13, highlights an important characteristic of the Bushman script and the corpus in general - that a large proportion of symbol classes occur only once or very infrequently in the text. These symbol classes, however, only make up a small number of the symbols in the text, which seems to be dominated by the symbols that occur more frequently.

Table 4.6: Average number of samples for each symbol class

O	F	WO	A-100	A-75	A-50	A-25
23.36	23.36	26.38	26.48	26.70	27.13	27.69

4.5.2.2 Experiment: Word Distribution

- **Purpose**
  - To identify the frequency count of words in the corpus.
- **Procedure**
  - The number of occurrences of each word class in all transcriptions was counted.
  - Punctuation marks (comma, period, exclamation mark, semi-colon) were removed before calculating word frequencies.
  - The above procedure is repeated for Corpus-O, Corpus-F, Corpus-W, Corpus-A-100, Corpus-A-50, Corpus-A-75 and Corpus-A-25.
- **Results** The word frequency count is shown in Figure 4.14 and the dominant frequency range of 1-10 is shown in greater detail in Figure 4.15.

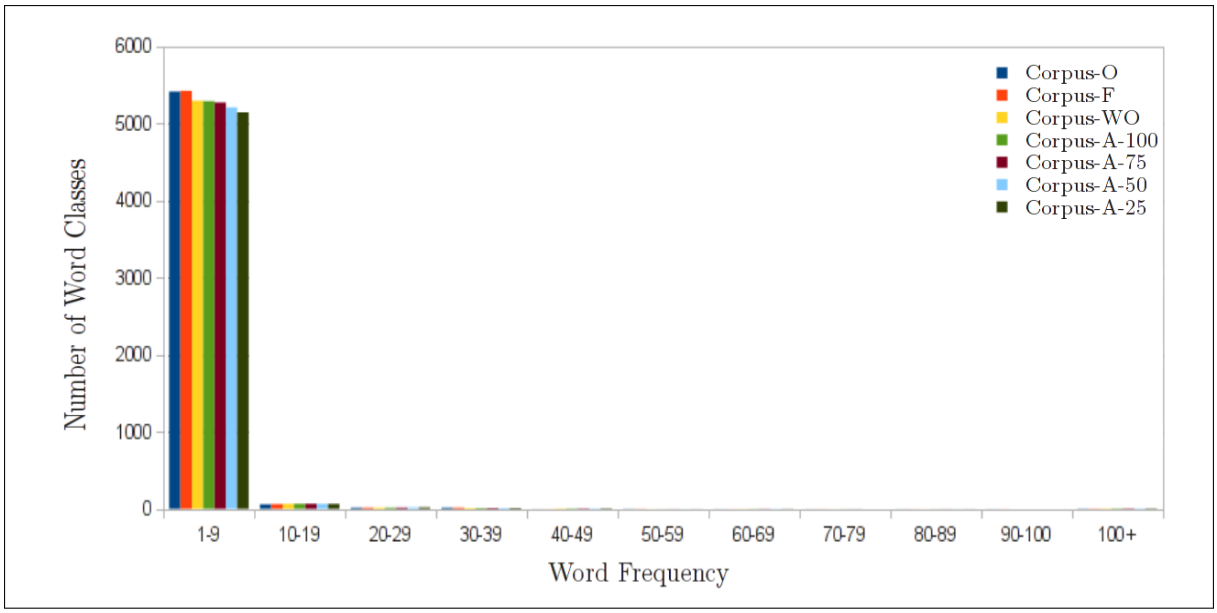


Figure 4.14: Word frequency count in corpus

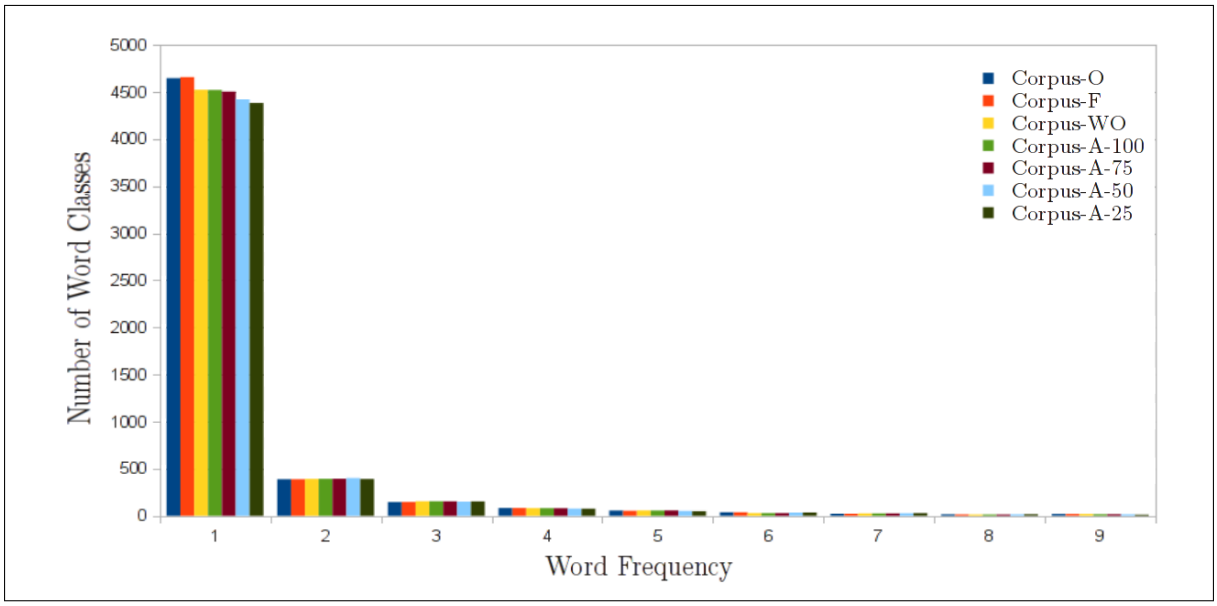


Figure 4.15: Word frequency count in corpus for frequency range 1-10

- **Discussion**
  - As can be seen in Figure 4.14, a case similar to that for the symbol class frequency distribution occurs with a large number of words occurring fewer than 10 times in the corpus. This is further investigated in Figure 4.15, which shows that a large number of words in this frequency range only contain a single sample. 83.82% of words only occur once in the text and 97.99% of words occur fewer than 10 times. It is highly likely that the distribution of words in this corpus follows Zipf's Law (Zipf, 1949), which states that the frequency with which a word occurs in a corpus of a natural language is inversely proportional to its frequency rank. For instance, the most frequent word occurs twice as often as the second most frequent, three times more often than the third most

frequent, etc.

Recognition takes place in two ways in this study. In the first way, referred to as word recognition, whole words are recognised as a single pattern. In this sense, an image containing a single word is provided for training and recognition. The word itself represents the class and the word is recognised as a whole rather than have the symbols that it is made up of individually recognised and then concatenated to form a word. Thus, in the case of word recognition, a word will not be recognisable when it occurs only once as the single occurrence can only be in one of the training and testing sets. Thus, when performing whole word recognition, the corpus is sampled to account for this. The way in which the corpus is sampled is discussed in Section 6.3.1.

The second way in which words are recognised in this study is when they form part of text line, which is referred to as text line recognition. In text line recognition, the symbols that appear in a text line are recognised and thus a word that occurs only once can still be recognised based on the symbols that it is made up of. Thus, for text line recognition, any word can be recognised.

The average number of samples per word class for each corpus is shown in Table 4.7 and the average number of samples across all of the corpora is 1.87.

Table 4.7: Average number of samples for each word class

O	F	WO	A-100	A-75	A-50	A-25
1.81	1.81	1.87	1.87	1.88	1.90	1.93

4.6 Discussion

The corpus described in this chapter is used for experimentation in the rest of this study and, since it was found that there were very few differences between the different versions of the corpus, Corpus-W is used since it at least accounts for the common sources of confusion from the corpus creation workshops. The lack of consistency among the different data capturers is of concern since it has a direct effect on the robustness of the recognition models. For instance, the 75% consistency suggests that, for any transcription output, the probability of a false negative or positive is 25%. Furthermore, the fact that there is a lack of consistency also suggests that recognition models will suffer from high variance due to having samples of the same symbols being classified as belonging to multiple classes. Thus, all evaluation results presented in Chapter 7 should be considered within this context. However, since the focus of this study is on evaluating different techniques for Bushman handwriting recognition, the corpus still allows for a robust comparison of the different techniques considered since all evaluation is based on the same data.

Chapter 5

Features

Descriptive features from the literature are used in this study in order to investigate their ability, when coupled with machine learning algorithms, to correctly classify handwritten Bushman texts. Chapter 2 introduced some of the desired properties of descriptive features as described by Nixon and Aguado (2008), which are repeated here for completeness.

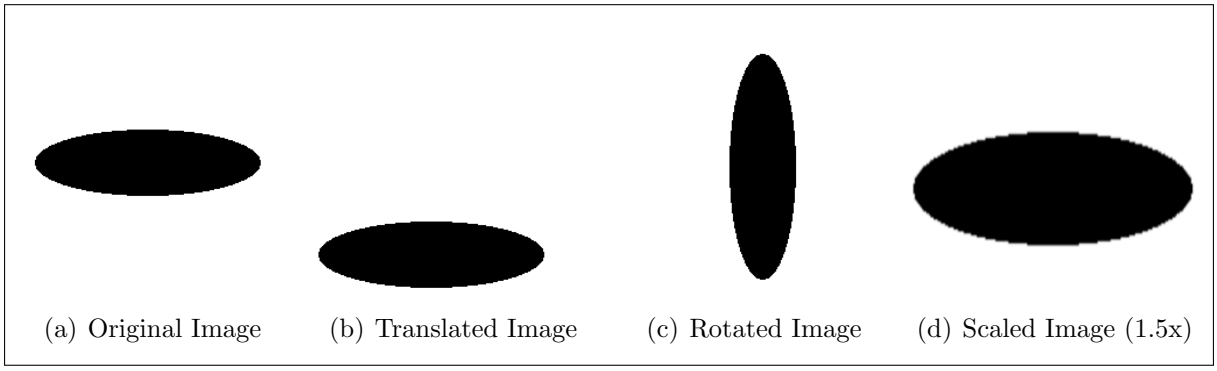
- Two objects should only have the same descriptors if they are the same.
- Descriptors should be congruent such that two similar objects will have similar descriptors.
- It is convenient for descriptors to be invariant to scale, rotation and translation.
- Descriptors should be compact so that objects are described in an efficient way.

It is, of course, debatable as to whether it is in fact always desirable for features to be invariant to scale, rotation and translation. For instance, if a feature had all three invariant properties it would result in the same descriptors for p and d, thereby invalidating the first desired property that two objects should only have the same descriptors if they are the same. Thus, as part of this study, the use of both invariant and non-invariant features is investigated.

It has been speculated that there are hundreds of different features that can be used for handwriting recognition (Arica, 1998), making implementing and evaluating every possible feature set an impossible task. Therefore this study has instead focused on a small subset of features used in the literature. The features used in this study have been chosen due to their varying complexity as well as their popularity in literature. The features describe different properties of the images, such as the distribution of pixels, the directions of strokes and the statistical properties of the image. Therefore, while not a complete evaluation of all possible features, this study still provides a valuable comparison of different types of features.

This chapter discusses the descriptive features used in this study. This discussion begins with a description of the feature extraction procedure for the different machine learning algorithms. Then, for each of the descriptive features used in this study, the derivation of the feature is discussed, followed by an evaluation of its invariant properties and, lastly, the way in which the feature was extracted is described.

The invariant properties of each feature described in this chapter is analysed by extracting features from a shape as well as translated, rotated and scaled versions of it. The images from which the features are extracted are shown in Figure 5.1. Unlike handwriting, the images in Figure 5.1 are generic and symmetric; however, they still provide a good illustration of the invariant properties of the different features discussed in this chapter and show how they differ.



(a) Original Image (b) Translated Image (c) Rotated Image (d) Scaled Image (1.5x)

Figure 5.1: Images for analysing invariant properties of features

.

5.1 Feature Extraction Procedure

The features used in this study require that parameters for feature extraction be specified. The values for these parameters could potentially have an impact on the performance of the recognisers. Thus, all experiments involving features were designed to investigate the effect of varying parameter values during feature extraction. In this section, a general description of the way in which features are extracted for the different machine learning algorithms used in this study is described. Then, when each of the features is discussed in the later sections of this chapter, the specific method for extracting each feature is described.

5.1.1 Support Vector Machine and Artificial Neural Network Feature Extraction

Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs) are used in this study for Bushman word recognition, which is discussed in Section 6.3. In this study, Bushman words are recognised as a whole rather than having the individual symbols that make up a word recognised. Thus, for SVMs and ANNs, feature extraction is performed on the word image as a whole, which may or may not be partitioned into cells. When the word image is partitioned into cells, it is portioned into C equally sizes cells where $C = 2^y, y \in Z$ and features are extracted from each cell. Formally, given a feature extraction function ϕ and a word image I that has been partitioned into C cells, the feature vector F is constructed as follows:

$$F = \phi(I_1) + \phi(I_2) + ... + \phi(I_n), n = 1, 2, ...C, \tag{5.1}$$

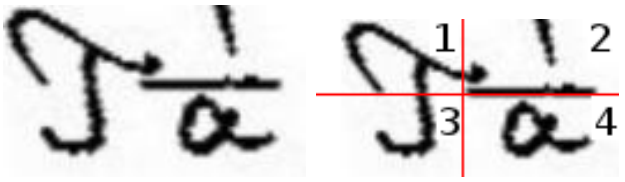
where n is the nth cell that the word image I is partitioned into.

Figure 5.2 is used to demonstrate the feature extraction procedure for SVMs and ANNs. For the unpartitioned word in Figure 5.2, the feature vector F is given by

$$F = \phi(I),$$

whereas for the image that has been partitioned into 4 cells the feature vector F is given by

$$F = \phi(I_1) + \phi(I_2) + \phi(I_3) + \phi(I_4).$$



(a) Whole word image

(b) Word image partitioned into 2 cells along each dimension

Figure 5.2: Feature extraction on partitioned word image and whole word image

5.1.2 Hidden Markov Model Feature Extraction

Hidden Markov Models (HMMs) are used in this research for Bushman word and text line recognition. Features for HMM-based recognition are extracted using a left-to-right sliding window that models the time domain. When using HMMs for Bushman word recognition, the classes that are recognised are whole words whereas, when recognising text lines, the classes that are recognised are individual symbols that are then concatenated to form words. When extracting features, the width of the window (or image column) needs to be specified as well as any overlap between adjacent windows. Similar to the case for SVM and ANN-based feature extraction, the window can be partitioned into cells and features are extracted from each cell. However, in the case of HMM-based feature extraction, the window is only divided into C cells along the vertical axis where $C = 2^y, y \in Z$. Formally, given a feature extraction function ϕ, a image I, a sliding window with width W with overlap O between adjacent windows and windows that have been vertically partitioned into C cells, the feature vector F at time step t is given by:

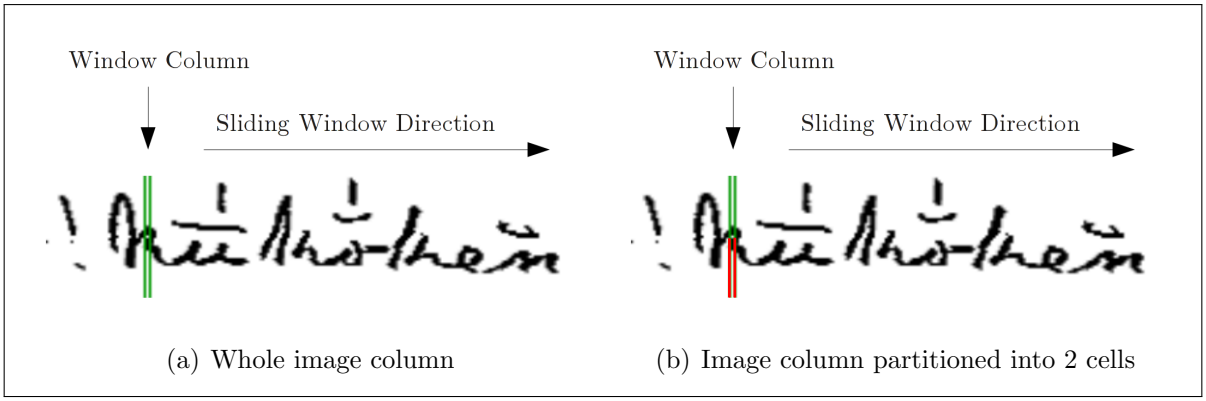
$$F^t = \phi(I_1^t) + \phi(I_2^t) + ... + \phi(I_n^t), n = 1, 2, ...C, \tag{5.2}$$

where I_n^t is the nth cell in the window at time step t. The number of time steps T is given by:

$$T = \frac{\text{Width of whole image}}{W - O}.$$

Figure 5.3 demonstrates how features are extracted using a sliding window. For the unpartitioned highlighted image column at time t, the feature vector F^t is given by:

$$F^t = \phi(I^t),$$



(a) Whole image column (b) Image column partitioned into 2 cells

Figure 5.3: Feature extraction using a sliding window

whereas for the partitioned image column at time t the feature vector F^t is given by:

$$F^t = \phi(I_1^t) + \phi(I_2^t).$$

In most cases, the column width $W = 1$ and overlap $O = 0$. However, in some cases, $W > 1$ and $O > 0$.

The general methods for extracting features for SVMs, ANNs and HMM were discussed above. However, for each of the features used in this study, the specifics of the feature extraction procedure depends on the parameters that need to be specified for each feature. In the next sections, each of the specific features is described. Then, its invariant properties are analysed followed by a description of the specifics of its extraction.

5.2 Undersampled Bitmaps

5.2.1 Description

Undersampled Bitmaps (UBs) are based on the density of the normalised number of black pixels within a cell in an image (Oliveira et al., 2006). An image is divided into a number of equally sized cells and, for each cell, the number of black pixels is counted and then normalised over the range [0-1] (Vamvakas et al., 2008) to create a feature vector F with C features, where C is the number of cells that the image is partitioned into. Cells that have a high pixel density will have high values and cells that have a low pixel density will have low values. These features are useful for showing the distribution of pixels in an image and are also invariant to small differences in pixel distribution (Oliveira et al., 2006). An example of the various densities for a Bushman word separated into 36 cells is shown in Figure 5.4.

5.2.2 Invariance

Table 5.1 shows the feature values for UBs when the images in Figure 5.1 are partitioned into 4 cells.

As can be seen from the table, UBs are invariant to scale, but not to translation or rotation. This is to be expected since the values of the UBs are dependent on the distribution of pixels in the Cartesian plane.

(a) Bushman word (b) UB densities for word

Figure 5.4: A Bushman word and a visual representation of its UBs

Table 5.1: Feature values for investigating invariant properties of UBs

Original Image	Translated Image	Rotated Image	Scaled Image
$UB_1 = 1$	$UB_1 = 0$	$UB_1 = 0.46$	$UB_1 = 1$
$UB_2 = 0.93$	$UB_2 = 0$	$UB_2 = 1$	$UB_2 = 0.92$
$UB_3 = 0.51$	$UB_3 = 1$	$UB_3 = 10.41$	$UB_3 = 0.51$
$UB_4 = 0.46$	$UB_4 = 0.62$	$UB_4 = 0.91$	$UB_4 = .46$

5.2.3 Extraction

For SVMs and ANNs, UBs are extracted by partitioning the word image into $C = 2^y, y \in 1, 2, 3, 4$ cells and calculating the UB for each cell. For HMMs, the width of the sliding window is given by $W = 1$ and the overlap is given by $O = 0$. UBs are extracted from each cell in each column in the sliding window by partitioning the column into $C = 2^y, y \in 1, 2, 3, 4$ cells.

5.3 Marti & Bunke Features

5.3.1 Description

Marti and Bunke (2002) proposed nine geometric features that are extracted using a sliding window, from here on referred to as Marti & Bunke (M&B) features. The first three of these features characterise the window relative to the global point of view, while the other six are used to give more details about the writing. The first three features are the weight of the window (given by the number of foreground pixels), the centre of gravity of the window and the second order moment of the window. These three features, $F_1(x), F_2(x)$ and $F_3(x)$, are given by the following equations:

$$F_1(x) = \frac{1}{m} \sum_{y=1}^{m} p(x, y) \tag{5.3}$$

$$F_2(x) = \frac{1}{m} \sum_{y=1}^{m} yp(x, y) \tag{5.4}$$

$$F_3(x) = \frac{1}{m^2} \sum_{y=1}^{m} y^2 p(x, y), \tag{5.5}$$

where m is the height of the window and $p(x, y)$ is the pixel value at column x and row y. Features four and five are the lower and upper positions of the contours of the window. Features six and seven are the gradients of the upper and lower contours of the window. Feature eight is the number of black to white transitions and feature nine is the number of black pixels between the lower and upper contours of the window. Features $F_4(x), F_5(x), F_6(x), F_7(x), F_8(x), F_9(x)$, are given by the following equations:

$$F_4(x) = argmin_y(p(x, y) = 1) \tag{5.6}$$

$$F_5(x) = argmax_y(p(x, y) = 1) \tag{5.7}$$

$$F_6(x) = \frac{d}{dx} F_4(x) \tag{5.8}$$

$$F_7(x) = \frac{d}{dx} F_5(x) \tag{5.9}$$

$$F_8(x) = \text{Number of black/white transitions} \tag{5.10}$$

$$F_9(x) = \frac{\sum_{y=F_4(x)}^{F_5(x)} p(x, y)}{F_5(x) - F_4(x)}, \tag{5.11}$$

where $p(x, y)$ is the pixel value at column x and row y.

Figure 5.5 shows an example of some of the M&B features being extracted from an image column.

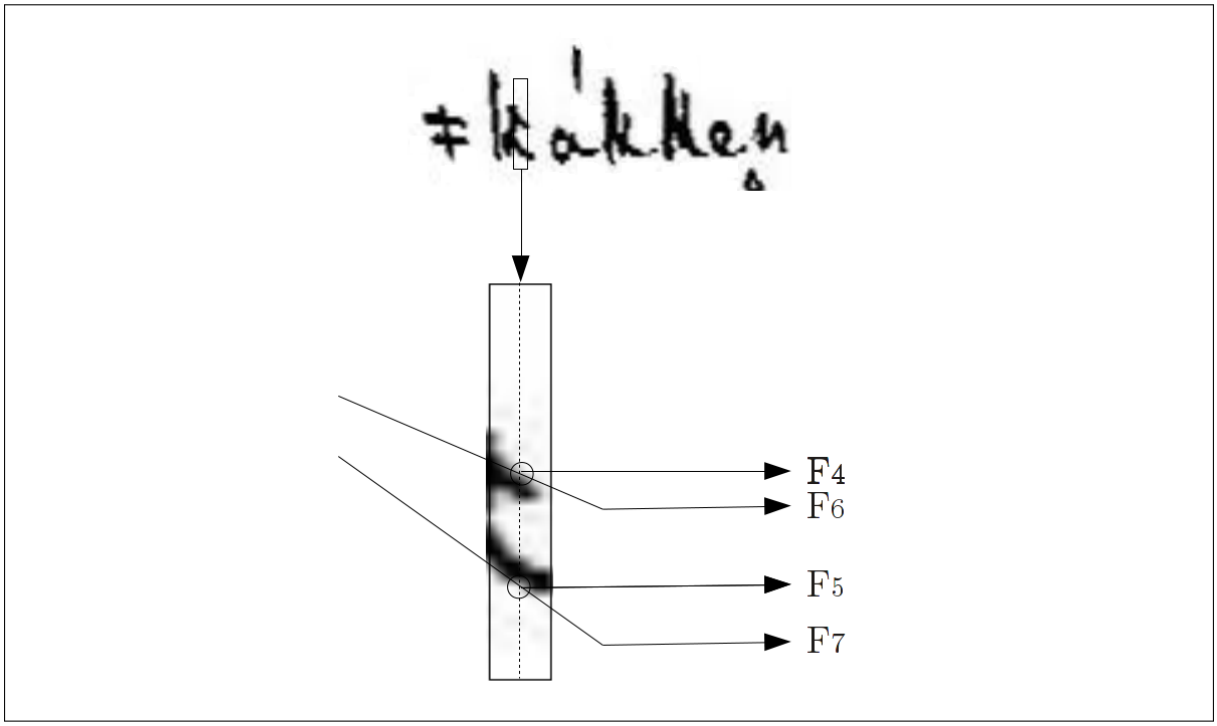


Figure 5.5: Examples of M&B features for an image column

Marti and Bunke (2002) note the importance of careful preprocessing in order to make these features robust when it comes to different writing styles.

5.3.2 Invariance

Table 5.2 shows the feature values when M&B features were extracted from the 100th column of the images in Figure 5.1.

Table 5.2: Feature values for investigating invariant properties of M&B features

Original Image	Translated Image	Rotated Image	Scaled Image
$M\&B_1^{100} = 0.22$	$M\&B_1^{100} = 0.23$	$M\&B_1^{100} = 0$	$M\&B_1^{100} = 0.20$
$M\&B_2^{100} = 25.74$	$M\&B_2^{100} = 44.67$	$M\&B_2^{100} = 0$	$M\&B_2^{100} = 32.07$
$M\&B_3^{100} = 12.27$	$M\&B_3^{100} = 35.28$	$M\&B_3^{100} = 0$	$M\&B_3^{100} = 15.25$
$M\&B_4^{100} = 90$	$M\&B_4^{100} = 168$	$M\&B_4^{100} = 250$	$M\&B_4^{100} = 130$
$M\&B_5^{100} = 144$	$M\&B_5^{100} = 224$	$M\&B_5^{100} = 0$	$M\&B_5^{100} = 198$
$M\&B_6^{100} = 0$	$M\&B_6^{100} = 0$	$M\&B_6^{100} = 0$	$M\&B_6^{100} = 0$
$M\&B_7^{100} = 0$	$M\&B_7^{100} = 0$	$M\&B_7^{100} = 0$	$M\&B_7^{100} = 5$
$M\&B_8^{100} = 2$	$M\&B_8^{100} = 2$	$M\&B_8^{100} = 0$	$M\&B_8^{100} = 0$
$M\&B_9^{100} = 1$	$M\&B_9^{100} = 1$	$M\&B_9^{100} = 0$	$M\&B_9^{100} = 0.96$

As can be seen from the table, M&B features are not invariant in any way.

5.3.3 Extraction

For M&B features, the feature extraction for SVMs and ANNs is the same as that for HMMs. M&B features are extracted from each column in the image using a sliding window with width $W = 1$ and overlap $O = 0$.

5.4 Geometric Moments

5.4.1 Description

Moments, which are statistical global descriptors of the shape of an image, have been used as features in a number of handwriting recognition studies. The fact that they provide a compact global description of an object, have a built-in ability to discern and filter noise and have invariant properties has made them popular and successful in a number of applications (Nixon and Aguado, 2008). Moments were first used for image analysis in the 1960s by Hu (1962) and have since been used in a number of applications, such as facial recognition (Nabatchian et al., 2008) and object recognition in road scenes (Apatean et al., 2008).

The calculation of geometric moments will be briefly discussed here. For more details, see Hu (1962).

5.4.1.1 Basic Properties

For a 2-D continuous function, the $(p+q)th$ order moment of a function $I(x, y)$ is defined as:

$$m_{pq} = \int_{-\infty}^{\infty} x^p y^q I(x, y). \tag{5.12}$$

For a discrete 2-D function, Equation 5.12 can be approximated by:

$$m_{pq} = \sum_x \sum_y x^p y^q I(x,y) \triangle A, \tag{5.13}$$

where $\triangle A$ is the area of a pixel.

The zero-order moment, m_{00} represents the total mass (intensity) of an image. In the case of a binary image, it represents the area of the foreground object Flusser et al. (2009) as is given by:

$$m_{00} = \sum_x \sum_y I(x,y) \triangle A, \tag{5.14}$$

The two first-order moments - m_{10} and m_{01} - are given by:

$$m_{10} = \sum_x \sum_y xI(x,y) \triangle A \quad m_{10} = \sum_x \sum_y yI(x,y) \triangle A. \tag{5.15}$$

Using these moments, the centre of mass (centroid) - $(\bar{x}, \bar{y})$ - of the image can then be calculated as the ratio of the first and zero order components:

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \bar{y} = \frac{m_{01}}{m_{00}}. \tag{5.16}$$

In order to make the moments invariant to translation, they need to be normalised with respect to the centre of mass. Given the centre of mass, the centralised moments, μ_{pq}, which are invariant to translation, are given by:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x,y) \triangle A. \tag{5.17}$$

The second-order moments of an image describe the variance of the image intensity about the origin, and are analogous to the moments of inertia in mechanics (Flusser et al., 2009). The second-order central moments, μ_{20} and μ_{02}, give the variance about the centre of mass, and μ_{11} describes the covariance between them.

Figure 5.6 shows the second-order centralised moments of a shape as well as its second-order centralised moments when translated and when rotated. As a result of the centralisation, the centralised moments are invariant to translation, but not to rotation and scale.

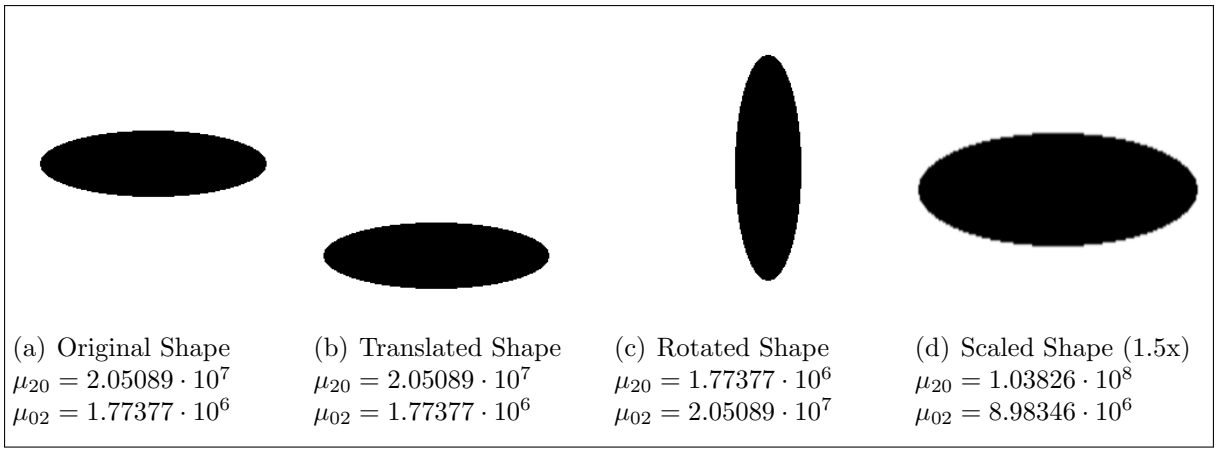
5.4.1.2 Invariant Moments

In order for the centralised moments to be rotation and scale invariant, they need to be normalised. The normalised central moment, η_{pq}, is defined as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \tag{5.18}$$

where

$$\gamma = \frac{p+q}{2} + 1, \quad \forall p + q \geq 2. \tag{5.19}$$



(a) Original Shape
$\mu_{20} = 2.05089 \cdot 10^7$
$\mu_{02} = 1.77377 \cdot 10^6$

(b) Translated Shape
$\mu_{20} = 2.05089 \cdot 10^7$
$\mu_{02} = 1.77377 \cdot 10^6$

(c) Rotated Shape
$\mu_{20} = 1.77377 \cdot 10^6$
$\mu_{02} = 2.05089 \cdot 10^7$

(d) Scaled Shape (1.5x)
$\mu_{20} = 1.03826 \cdot 10^8$
$\mu_{02} = 8.98346 \cdot 10^6$

Figure 5.6: Second order centralised moments of original shape, translated shape, rotated shape and scaled shape

Hu (1962) described a set of 7 invariant moments $(M_1, M_2, ..., M_7)$, which can be calculated using Equation 5.18 as follows:

$$\begin{aligned}
M_1 =& \eta_{20} + \eta_{02} \\
M_2 =& (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
M_3 =& (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
M_4 =& (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M_5 =& (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
& (3\eta_{12} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
M_6 =& (\eta_{20} - \eta_{02})[(\eta_{30} - \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] + \\
& 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
M_7 =& (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
& (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].
\end{aligned} \tag{5.20}$$

Hu moments M_1 and M_2 are second-order moments that describe the variance of the image. M_3, M_4, M_5 and M_6 are third-order moments that describe the skew of the image, and M_7 allows for distinguishing between mirror images. In this study, Hu moments, referred to as Geometric Moments (GMs), are added to the feature vector F and used as features in this study.

5.4.2 Invariance

Table 5.3 shows the first two GMs when extracted from the images in Figure 5.1.

Table 5.3: Feature values for investigating invariant properties of GMs

Original Image	Translated Image	Rotated Image	Scaled Image
$GM_1 = 0.29$	$GM_1 = 0.29$	$GM_1 = 0.29$	$GM_1 = 0.29$
$GM_2 = 0.06$	$GM_2 = 0.06$	$GM_2 = 0.06$	$GM_2 = 0.06$

As can be seen from the table, GMs are invariant to translation, rotation and scale.

5.4.3 Extraction

For SVMs and ANNs, GMs are extracted from each cell in the word image by partitioning the word image into $C = 2^y, y \in 0, 1, 2, 3$ cells. For HMMs, the width of the sliding window is given by $W = 1$ and the overlap is given by $O = 0$. GMs are extracted from each each cell in each column in the sliding window by partitioning the column into $C = 2^y, y \in 0, 1, 2, 3$ cells. The following GMs are extracted from each cell [M1], [M1,M2], [M1,M2,M3] and [M1,M2,M3,M4].

5.5 Histograms of Oriented Gradients

5.5.1 Description

Histograms of Oriented Gradients (HoGs) were first introduced by Dalal and Triggs (2005) as a set of descriptors for human detection, though they were predated by Gradient, Structural, Concavity (GSC) features, which contained HoG-like features (Favata and Srikantan, 1996). The basic thought behind HoGs is that local object shape and appearance in an image can be characterised by the distribution of local gradients. This distribution of local gradients is found by dividing an image into a series of equally-sized cells and then, for each cell, compiling a list of the gradients of each pixel in the cell and creating a histogram of gradients for that cell. The calculation of HoGs essentially involves four steps: gradient computation; orientation binning; separation into descriptor blocks; and block normalisation.

5.5.1.1 Gradient Calculation

Dalal and Triggs (2005) found that the best filter for gradient calculation was a simple 1-D mask along both the x-axis and the y-axis in the form of:

$$D_x = [-1\ 0\ 1] \quad D_y = [-1\ 0\ 1]^T, \tag{5.21}$$

such that for an Image I

$$I_x = I \times D_x \quad I_y = I \times D_y. \tag{5.22}$$

The magnitude of the gradient of a pixel at location $I(x, y)$ in image I is then given by:

$$|G| = \sqrt{{I_x}^2 + {I_y}^2}. \tag{5.23}$$

The orientation of the gradient of a pixel at location (x, y) in image I is given by:

$$\theta = \arctan\left(\frac{I_x}{I_y}\right). \tag{5.24}$$

Transforming the gradient to degrees involves the following equation which gives values in the range $[-180°; 180°]$:

$$\alpha = \theta \times \frac{180}{\pi}. \tag{5.25}$$

These gradients are then used in the orientation binning stage to create the HoGs.

5.5.1.2 Orientation Binning

In the orientation binning stage, each pixel casts a weighted vote for an orientation-based histogram channel, based on its orientation found in the gradient calculation stage. Dalal and Triggs (2005) found that 9 evenly spaced bins produced the best results. The orientation bins in the histogram are evenly spaced over the range $[0; 180°]$ for an "unsigned" gradient or $[0; 360°]$ for a "signed" gradient. Given that the gradients are in the range $[-180°; 180°]$, they need to be translated to the range of the bins. For unsigned gradients:

$$\alpha_{unsigned} = \begin{cases} \alpha, \text{ if } \alpha \geq 0 \\ \alpha + 180, \text{ if } \alpha < 0, \end{cases}, \tag{5.26}$$

and for signed gradients:

$$\alpha_{signed} = \begin{cases} \alpha, \text{ if } \alpha \geq 0 \\ \alpha + 360, \text{ if } \alpha < 0, \end{cases} \tag{5.27}$$

The weight of a pixel's vote can be the pixel's magnitude itself, its square, its square root or a clipped form of the magnitude, which is based on the presence or absence of an edge at the pixel (Dalal and Triggs, 2005). In this study, the magnitude of each pixel is calculated according to Equation 5.28

$$M_{I(x,y)} = \sqrt{I_x^2 + I_y^2} \tag{5.28}$$

The results of the orientation binning stage is a histogram of oriented gradients for each cell of pixels.

5.5.1.3 Descriptor Blocks

Variations in illumination and foreground-background transitions in local regions of an image can result in gradients that vary widely over a wide range (Dalal and Triggs, 2005) and it was found by Dalal and Triggs that local contrast normalisation was essential for good performance. In order to perform this normalisation, it is necessary to group the cells into descriptor blocks and the final descriptor is then the normalised vector of all the components of the cells. These blocks often overlap so that each cell contributes several times to the final descriptor but normalised differently in each case (Dalal and Triggs, 2005).

5.5.1.4 Block Normalisation

Dalal and Triggs proposed four methods for block normalisation. Given the unnormalised vector v, let $||v||_k$ be its k-norm for $k = 1, 2$ and ϵ be some small constant. Then, the following normalisation schemes can be applied:

L2-norm $f = \dfrac{v}{\sqrt{||v||_2^2 + \epsilon^2}}$

L2-Hys L2-norm with clipping

L1-norm $f = \dfrac{v}{||v||_1 + \epsilon}$

L1-sqrt $f = \sqrt{\dfrac{v}{||v||_1 + \epsilon}}$.

It was shown by Dalal and Triggs that L2-norm, L2-Hys and L1-sqrt improved recognition performance equally, L1-norm reduced performance by 5% and no normalisation reduced performance by 27%. In this study, the L2-norm without overlapping is used to normalise blocks.

5.5.1.5 HoGs at Multiple Scales

Howe et al. (2009) used HoGs at multiple resolutions as features. In this approach, the different resolutions represent different cell sizes and allow for descriptive features to be captured at multiple scales. In this study, HoGs have been extracted at three different resolutions, referred to as $R1, R2$ and $R3$. The feature vectors for each resolution are then appended onto each other in order to form a single feature vector.

5.5.2 Invariance

Table 5.4 shows values of the HoGs when extracted from the images in Figure 5.1.

Table 5.4: Feature values for investigating invariant properties of HoGs

Original Image	Translated Image	Rotated Image	Scaled Image
$HoG_1 = 16$	$HoG_1 = 16$	$HoG_1 = 290$	$HoG_1 = 74$
$HoG_2 = 0$	$HoG_2 = 0$	$HoG_2 = 0$	$HoG_2 = 0$
$HoG_3 = 98$	$HoG_3 = 98$	$HoG_3 = 98$	$HoG_3 = 98$
$HoG_4 = 0$	$HoG_4 = 0$	$HoG_4 = 0$	$HoG_4 = 0$
$HoG_5 = 580$	$HoG_5 = 580$	$HoG_5 = 32$	$HoG_5 = 972$
$HoG_6 = 0$	$HoG_6 = 0$	$HoG_6 = 0$	$HoG_6 = 0$
$HoG_7 = 98$	$HoG_7 = 98$	$HoG_7 = 98$	$HoG_7 = 98$
$HoG_8 = 0$	$HoG_8 = 0$	$HoG_8 = 0$	$HoG_8 = 0$
$HoG_9 = 16$	$HoG_9 = 16$	$HoG_9 = 290$	$HoG_9 = 74$

As can be seen from the table, HoGs are invariant to translation, but not to rotation or scale. The HoGs for the rotated shape suggests some form of correlation with the original shape since where they differ, they differ by a fixed factor of 18.25.

5.5.3 Extraction

For SVMs and ANNs, the image is partitioned into cells and HoGs are extracted at 3 resolutions R_1, R_2, R_3. That is, the size of the cells are $R_1 \times R_1$, $R_2 \times R_2$ and $R_3 \times R_3$. For HMMs, HoGs are extracted from each column in a sliding window with window width $W = 3$ and overlap $O = 2$ at 3 resolutions R_1, R_2, R_3, such that the size of the cells are $R_1 \times W$, $R_2 \times W$ and $R_3 \times W$. For SVMs, ANNs and HMMs unsigned and unnormalised

HoGs are extracted where R_1, R_2, R_3 = [64,32,16], [32,16,8], [16,8,4], [64,32,0], [32,16,0], [16,8,0], [64,0,0], [32,0,0], [16,0,0].

In investigating the effect of signing and normalising the gradients, the assumption is made that the best performing unsigned values for R_1,R_2 and R_3 will remain the best for signed gradients and when the gradients are normalised. In order to determine the effect of signing the gradients, signed HoGs without normalisation are extracted for the best R_1, R_2 and R_3. Then, in order to determine the effect of normalising the HoGs, normalised HoGs are extracted for the best R_1, R_2 and R_3 where the block normalisation size is 2. Whether or not the gradients are signed depends on the findings of the previous step.

5.6 Gabor Filter-based Features

5.6.1 Description

A Gabor Filter (GF) is a Gaussian function modulated by a complex sinusoid in both the spatial domain and frequency domain and contains both a real and imaginary part (Chen et al., 2010). Multi-directional GFs have been used in character recognition to extract stroke information from characters (Wang et al., 2005). This is possible due to GFs being orientation specific (Chen et al., 2010). GF-based features have been shown to work well for automatic handwriting recognition since they operate directly on greyscale images rather than requiring the images to be binarised (Chen et al., 2010). This is beneficial since binarisation is usually a difficult task when it comes to historical documents due to poor quality, document degradation and other factors such as ink-bleed.

In the spatial domain, the GF is given by:

$$h(x, y, \lambda, \theta, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\{-\frac{1}{2}(\frac{R_1^2}{\sigma_x^2} + \frac{R_2^2}{\sigma_y^2})\} exp\{i\frac{2\pi R_1}{\lambda}\}, \tag{5.29}$$

where $R_1 = x\cos\theta + y\sin\theta$, $R_2 = y\cos\theta - x\sin\theta$, λ is the spatial wavelength, θ is the orientation and σ_x and σ_y are the variances along the x and y axes respectively.

In the frequency domain, the GF is given by:

$$H(u, v, \lambda, \theta, \sigma_x, \sigma_y) = Kexp\{-2\pi^2\sigma_x^2(F_1 - \frac{1}{\lambda})^2 + \sigma_y^2 F_2^2\}, \tag{5.30}$$

where K is a constant, $F_1 = u\cos\theta + v\sin\theta$ and $F_2 = u\cos\theta - v\sin\theta$.

The approach used in this study to create GF-based features is the same as that proposed by Chen et al. (2010) and is described briefly here. For more details, see Chen et al. (2010).

5.6.1.1 Feature Calculation

Calculate Spatial Frequencies According to Wang et al. (2005), the GF outputs the maximum response when

$$\lambda = 2w, \tag{5.31}$$

where w is the width of a handwritten stroke in an image. The value for w can be estimated by measuring the stoke widths in an image and then averaging them. The spatial frequency can then be calculated as:

$$f = \cos(\frac{1}{\lambda}) \tag{5.32}$$

In this study, $\lambda \in \{2, 4\}$, was empirically found to perform well and is used for experimentation.

Choose Orientations Wang et al. (2005) found the orientations for Chinese characters to cluster around the four angles $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$. The same orientations were used by Chen et al. (2010) for recognising Arabic characters. Due their successful use in previous studies, the same orientations are used in this study.

Construct and Apply Gabor Filters A total of $n \times m$ GFs are constructed and then applied to the image for the n frequencies and m orientations.

Determine Mean and Count Pixels Above Mean In their approach, Chen et al. chose to emphasise salient pixels and that is the approach followed here. For the filtered image, the mean magnitude of the response of the GF is computed and the number of pixels for which the magnitude exceeds the mean and are considered salient are counted.

Split Image Into Blocks and Compute Features The filtered image is divided into C cells. For each cell, the number of salient responses is calculated and divided by the total number of salient responses in the whole image and appended to the feature vector as a GF-based feature.

As a result of this feature calculation, a feature vector of length $n \times m \times C$ is created.

5.6.2 Invariance

Table 5.5 shows the feature values for GF-based features for the images in Figure 5.1, which were calculated based on the proportion of salient responses for 4 blocks. The values in Table 5.5 are the responses for the first (top-left) block for each frequency/orientation pair.

As can be seen from the table. GF-based features are invariant to scale but not to translation or rotation.

5.6.3 Extraction

For SVMs and ANNs, GFs are applied to each word image and the relative number of salient points for each of C equally sized cells is calculated where $C = 2^y, y \in 1, 2, 3, 4$. For HMMs, the width of the sliding window is given by $W = 4$ and the overlap is given by $O = 3$. GFs are applied to each image column and the relative number of salient points for each of C equally sized cells per column is calculated where $C = 2^y, y \in 1, 2, 3, 4$.

Table 5.5: Feature values for investigating invariant properties of GF-based features

Original Image	Translated Image	Rotated Image	Scaled Image
$GF_1 = 0.33$	$GF_1 = 0$	$GF_1 = 0.18$	$GF_1 = 0.33$
$GF_2 = 0.32$	$GF_2 = 0$	$GF_2 = 0.15$	$GF_2 = 0.33$
$GF_3 = 0.33$	$GF_3 = 0$	$GF_3 = 0.16$	$GF_3 = 0.34$
$GF_4 = 0.36$	$GF_4 = 0$	$GF_4 = 0.16$	$GF_4 = 0.36$
$GF_5 = 0.33$	$GF_5 = 0$	$GF_5 = 0.18$	$GF_5 = 0.33$
$GF_6 = 0.32$	$GF_6 = 0$	$GF_6 = 0.15$	$GF_6 = 0.33$
$GF_7 = 0.33$	$GF_7 = 0$	$GF_7 = 0.16$	$GF_7 = 0.34$
$GF_8 = 0.36$	$GF_8 = 0$	$GF_8 = 0.16$	$GF_8 = 0.36$

5.7 Discrete Cosine Transform Coefficients

5.7.1 Description

The Discrete Cosine Transform (DCT), related to the Fourier transform, was originally proposed by Ahmed et al. (1974) and expresses a function or signal in terms of a sum of different cosine functions at different frequencies (Nguyen and Bui, 2008). The DCT has been widely used in video and image processing, such as in the JPEG standard where it is used for compression. Image transforms are largely based on the idea that correlation exists between neighbouring pixels and that the correlation can be used to predict the value of a pixel based on its neighbouring pixels (Khayam, 2003). In this sense, a transform maps correlated data to uncorrelated data (Khayam, 2003). In this research, the Fastest Fourier Transform in the West (FFTW3) (Frigo and Johnson, 2005) implementation of the DCT is used.

There are eight kinds of DCT, though in practice only four of them are commonly used (Nguyen and Bui, 2008). The Type-II DCT is the most common and is the original one proposed by Ahmed et al. (1974) and is the one used in this research.

5.7.1.1 2-D Discrete Cosine Transform

For use as features for image recognition, the 2-D DCT is defined as Khayam (2003):

$$G_x(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right] cos\left[\frac{\pi(2x+1)v}{2N}\right], \tag{5.33}$$

where $u, v = 0, 1, 2, ..., N-1$ and $\alpha(u)$ and $\alpha(v)$ are given by Equation 5.34.

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } u = 0 \\ \sqrt{\frac{2}{N}}, & \text{if } u \neq 0, \end{cases} \tag{5.34}$$

The inverse transform of the 2-D DCT is given by Khayam (2003):

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)G_x(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right] cos\left[\frac{\pi(2x+1)v}{2N}\right], \tag{5.35}$$

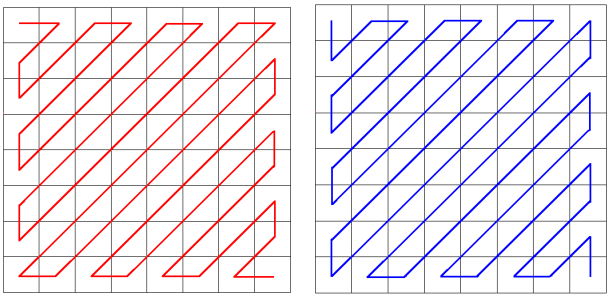
where $x, y = 0, 1, 2, ..., N - 1$.

5.7.1.2 DCT Coefficients as Features

Low frequency regions of the DCT contain most of the energy from the transform and encode most of the variance of an image (Khayam, 2003). For this reason, the low frequency coefficients of the DCT transform can be used as features, while the high frequency components can be discarded through a process called quantization (Khayam, 2003). The 2-D coefficients of the DCT are converted to a 1-D vector using a technique called zig-zagging (AlKhateeb et al., 2008), the purpose of which is to extract the low frequency coefficients from the DCT first. An example of an image of a Bushman word and its DCT transform, which shows most of the variance in the low frequency components, is shown in Figure 5.7 and two approaches for zig-zagging are shown in Figure 5.8.

(a) Bushman word (b) DCT Transform

Figure 5.7: An example of an image and its DCT



(a) (b)

Figure 5.8: Two different zig-zagging approaches for extracting low frequency coefficients from the DCT

Once the features have been extracted using zig-zagging, a linear feature vector F can be created.

5.7.2 Invariance

Table 5.6 shows the first five DCT coefficients when extracted from the images in Figure 5.1. The coefficients have been normalised by dividing all coefficients by the largest coefficient produced by the DCT.

As can be seen from the table, DCT coefficients are invariant to scale, but not to translation or rotation. Correlation also appears to exist between the coefficients of the original

Table 5.6: Feature values for investigating invariant properties of DCT coefficients

Original Image	Translated Image	Rotated Image	Scaled Image
$DCT_1 = 1$	$DCT_1 = 1$	$DCT_1 = 1$	$DCT_1 = 1$
$DCT_2 = 0.041$	$DCT_2 = 0.20$	$DCT_2 = -0.09$	$DCT_2 = 0.04$
$DCT_3 = 0.09$	$DCT_3 = -0.77$	$DCT_3 = 0.04$	$DCT_3 = 0.09$
$DCT_4 = -0.92$	$DCT_4 = 0.21$	$DCT_4 = -0.42$	$DCT_4 = -0.92$
$DCT_5 = 0.004$	$DCT_5 = -0.150$	$DCT_5 = 0.004$	$DCT_5 = 0.004$

shape and the rotated shape, i.e. coefficients C_2 and C_3 are swopped, and coefficient C_5 is the same for both shapes.

5.7.3 Extraction

For SVMs and ANNs, DCT coefficients are extracted by partitioning the word image into $C = 2^y, y \in 0, 1, 2, 3$ cells and applying the DCT to each cell. For HMMs, the width of the sliding window is given by $W = 1$ and the overlap is given by $O = 0$. DCT coefficients are extracted from each column in the sliding window by partitioning the column into $C = 2^y, y \in 0, 1, 2, 3$ cells and applying the DCT to each cell. For each cell, S DCT coefficients are extracted using zig-zagging where $S = 1, 10, 20, 30, 40, 50$.

5.8 Discussion

The role of features in this study is to act as compact descriptors, which allow for the machine learning algorithms to distinguish among different words, characters and symbols. This section has discussed the features used in this study, beginning with a discussion of their derivation and implementation, then a description of the parameters for feature extraction and a discussion of their invariant properties. The features used in this study were chosen based on their popularity and successful use in the literature, their varying levels of invariance, their varying levels of difficulty in terms of derivation and implementation and their descriptive focus, i.e. distribution of pixels, orientation of lines and pixel gradients. These features are used for experimentation for Bushman handwriting recognition and, in the next chapter, the experimental design used in this study is described.

Chapter 6

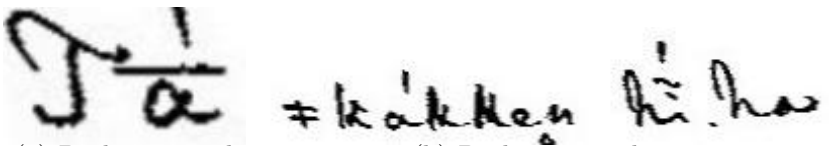
Experimental Design

The main purpose of this study was to investigate different techniques for the recognition of handwritten Bushman texts. These techniques are influenced by a number of factors including the following:

- The choice of machine learning algorithm and model parameters.
- The choice of descriptive features and feature extraction parameters.
- The recognition unit, i.e. character, word, text line.
- The amount of training data that is available and the use of synthetic training data.
- The number of authors that the recognition engine is trained for.
- The use of additional information such as a dictionary and statistical information about the language.

In this study, the effects of the above factors are investigated through a series of experiments designed to investigate each factor individually, as well as combinations of the different factors. There is a potentially infinite number of combinations of techniques for handwriting recognition and it would be infeasible to consider all of them. Therefore, a selection of combinations are investigated in order to gain insight into those which produce the best results. The general goal of this study is to come up with a best approach for automatic Bushman handwriting recognition with potential application to other historical texts. Once this best approach has been determined, future research could focus on fine-tuning the findings in order to improve results and potentially build production-ready handwriting recognition systems.

Two approaches to Bushman handwriting recognition are investigated in this study. In the first approach, referred to as word recognition, Bushman words are recognised where images of Bushman words are used for training and recognition, each word represents a single class and the word is recognised as a single pattern, rather than recognising the individual symbols that it is made up of. In the second approach, referred to as text line recognition, images of Bushman text lines are used for training and recognition, the Bushman symbols that appear in text lines are recognised individually and a dictionary is used to form words from the symbols. Figure 6.1 shows examples of the two recognition units that are are used in this study.



(a) Bushman word (b) Bushman text line

Figure 6.1: The recognition units used in this study

In this chapter, the experimental design used in this study is described. This description begins by discussing the common methodology applied to all experiments in this study. Then, the factors that are investigated in the experiments are discussed, followed by a description of the experimental design for the recognition of Bushman words using Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Hidden Markov Models (HMMs) and then the recognition of Bushman text lines using HMMs. This chapter only gives an overview of the experimental design, while the actual experiments and results are presented in Chapter 7.

6.1 General Approach

The specific methodologies for the individual experiments in this study vary. However, there are some aspects of the approach that remain constant across all experiments. In this section, these aspects that remain constant are discussed, while the specific aspects of the methodologies for each experiment are discussed in the relevant sections. In this section, the general experimental methodology is discussed, followed by a discussion of steps taken to minimise variation in experimental design. Cross validation is then described, followed by a description of the performance metrics used in this study. Lastly, the creation of synthetic training data that is used in some experiments in this study is described.

6.1.1 General Experimental Methodology

In general, the following experimental methodology is used for all experiments in this study.

1. A machine learning algorithm is selected.
2. The machine learning algorithm parameters are set.
3. Descriptive features are selected.
4. The feature extraction parameters are set.
5. Preprocessing takes place and features are extracted.
6. Using 10-fold cross validation, training and recognition takes place.
7. Postprocessing is performed.

6.1.2 Experiment Variation Reduction

A comparative study, such as this, can easily be influenced by variation in experimental design. Therefore, the following steps have been taken in an attempt to minimise variation between experiments and thus reduce bias.

- Each word and text line sample is randomly partitioned for cross validation (see Section 6.1.3) and the partitions are kept constant for all experiments in order to minimise selection bias.
- Features are extracted with the same parameters for the different machine learning algorithms, unless the parameters need to differ due to the fundamental nature of the machine learning algorithms.
- The same set of experiments are conducted for each machine learning algorithm at each segmentation level.

6.1.3 Cross Validation

Cross validation is a method for estimating prediction error when an independent test sample is used for the estimation (Hastie et al., 2001). A separate validation set can be used for cross validation. However, this requires that, in addition to the data required for the training set, enough data is available to form a reliable validation set. An alternative to having an independent validation set is to perform K-fold cross validation. In K-fold cross validation, the full data set is randomly split into K independent folds, where a fold is a partition of the data (Hastie et al., 2001). $K-1$ folds are then used for training prediction models and the remaining fold is used for testing. This process is then repeated, each time selecting a different fold for testing and training with the remaining $K-1$ folds. Figure 6.2 shows an example of how data can be separated into 10 folds and how the testing data can be rotated. The outcome of cross validation is an independent recognition accuracy for each of the K folds that was used for testing. The overall recognition accuracy can then be calculated as the average recognition accuracy of the K folds

10-fold cross validation is used in this study as it has been recommended as a good compromise in terms of bias, variance and over-estimating the true prediction error (Hastie et al., 2001). In this study, all text line and word samples are randomly separated into 10 folds. These randomly created folds are then kept constant for all experiments, thereby allowing for a direct comparison of techniques with fixed data points. All results reported for the experiments described in this study are based on the averages for the 10 folds.

6.1.4 Performance Metrics

There are two performance metrics used during evaluation in this study. The first of these metrics is used for word recognition and is a binary function for whether the whole word was correctly recognised or not and the recognition accuracy is given by:

$$Accuracy = \frac{\text{Number of words correctly recognised}}{\text{Total number of words}} \times 100\%. \tag{6.1}$$

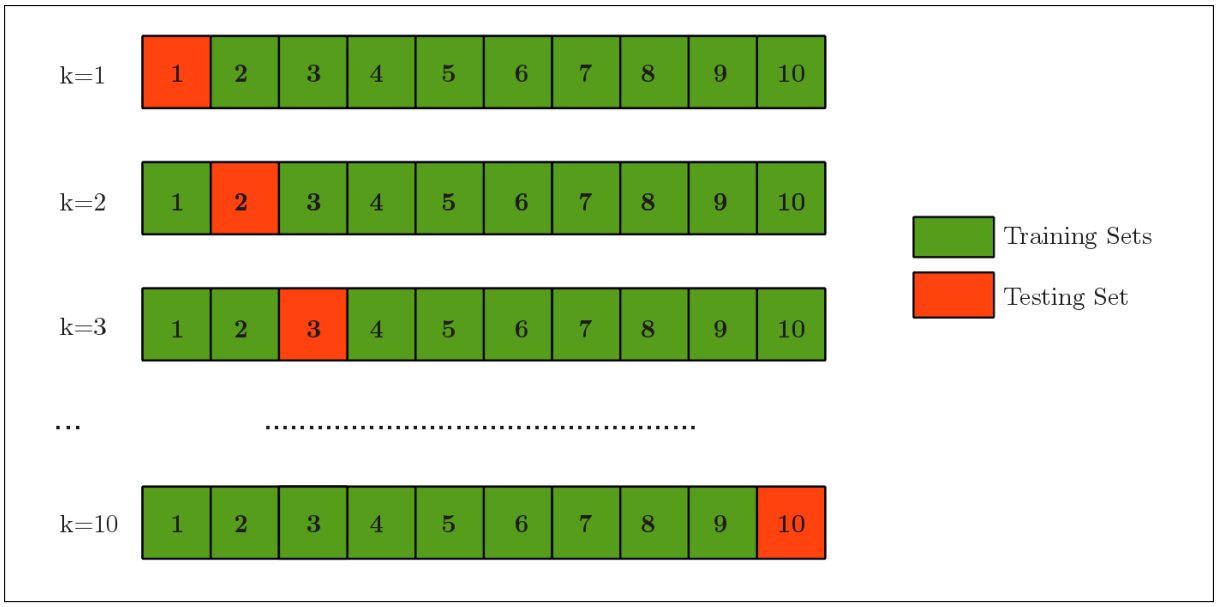


Figure 6.2: Example of k-fold cross validation, $k = 10$

The metric used for text line recognition is based on how many symbols in the transcription were correctly recognised, where a symbol refers to a single character, character/diacritic combination or a space. The metric is a function of the number of substitutions, deletions and insertions required to transform the recognised text into the correct transcription and is given by:

$$\text{Accuracy} = \frac{N - S - D - I}{N} \times 100, \tag{6.2}$$

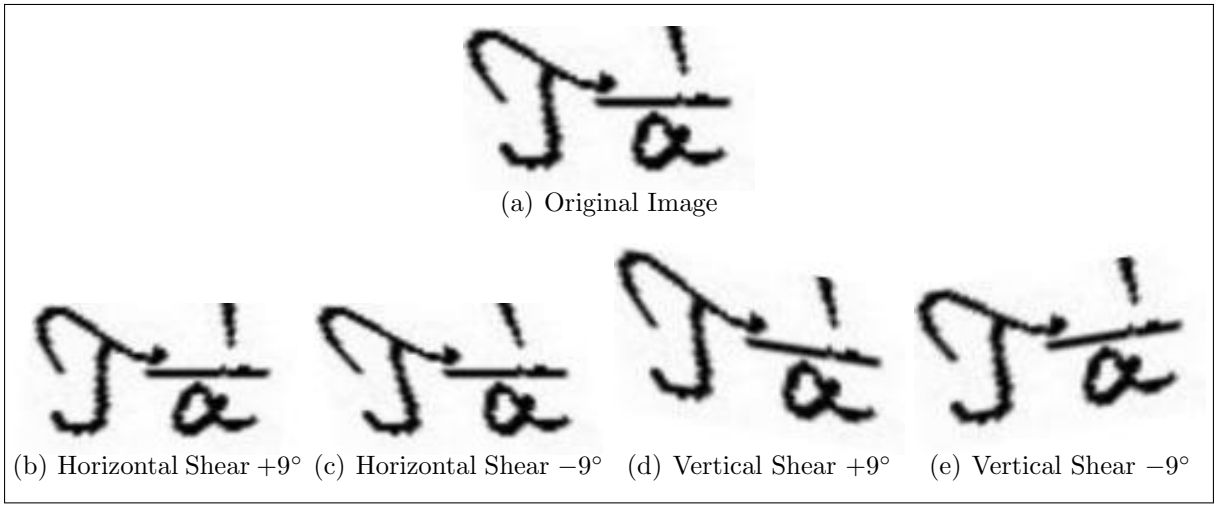
where N is the length of the correct transcription, S is the number of substitution errors, D is the number of deletion errors and I is the number of insertion errors. A substitution error occurs when a symbol is misclassified, an insertion error occurs when the recognition output contains a symbol that does not occur in the original transcription and a deletion error occurs when a symbol that occurs in the original transcription does not appear in the recognition output (Vinciarelli et al., 2004). This recognition accuracy is not actually a percentage as it can be below 0, which could occur, for instance, if there are a large number of insertion errors. The maximum value for the recognition accuracy is 100 and occurs when there are no substitution, deletion or insertion errors. This measure of recognition accuracy is more indicative of the quality of an automatic transcription compared to say, for instance, a measure of the number of symbols correctly recognised. This is because it is possible to correctly recognise most symbols, but still have a low quality transcription as a result of the recogniser outputting a large number of additional symbols, i.e. more than actually appear in the text, which a simple measure of the number of symbols correctly recognised does not take into account but that the measure of accuracy used in this study does.

6.1.5 Synthetic Data

A well-established rule of thumb in handwriting recognition is that more training data results in better the recognition accuracy (Varga and Bunke, 2003). For instance, Rowley

et al. (2002) showed how an increase from 10 training samples per character class to 1000 samples per character class was able to reduce recognition error rates by more than 10%. However, it is not always the case that there are large amounts of training data available and many researchers are often limited in this respect. One way to overcome the lack of real training data is to supplement the real training data with synthetic training data. Cano et al. (2002) used some simple image transformations to create synthetic training data. The transformations Cano et al. used included slant and shrink transformation to account for geometric distortions and erosion and dilation transformation to account for different writing instruments and acquisition conditions. Cano et al. showed that, while all of the transformations reduced recognition error rates, the slant transformation had the largest effect. In this study, the effect that synthetic training data has on performance is investigated. The synthetic data was created by performing a $\pm 9°$ shear operation along both the horizontal and vertical axes.

A number of experiments make use of these transformations in order to investigate the effect that supplementing the training set with synthetic data has on performance. Figure 6.3 shows an example of an image with its synthetic variants after the transformations have been applied.



(a) Original Image

(b) Horizontal Shear $+9°$ (c) Horizontal Shear $-9°$ (d) Vertical Shear $+9°$ (e) Vertical Shear $-9°$

Figure 6.3: Bushman word image and its synthetic variants

While the methodology for each experiment varies depending on the factors being investigated, there are some elements of the approaches that remain constant for all experiments and which have been discussed in this section.

6.2 Factors Investigated

This is a comparative study where a number of features and other factors that affect handwriting recognition are compared. The focus is on the use of each of these factors with different machine learning algorithms. These factors relate to, among others, the use of different features, the use of synthetic data, the effect of the amount of training data available and the effect of multiple authors. The actual experiments investigating each of these factors are presented in the next chapter. The purpose of this section, however, is to provide motivation for why each of the factors is investigated.

6.2.1 Different Features

One of the considerations when creating a handwriting recognition system involves the choice of descriptive features and this study investigates a number of descriptive features from the literature and that were described in Chapters 5. The features that are used and the parameters used for feature extraction could potentially play a significant role in the performance of the different recognisers. For this reason, different features are evaluated when used with the different machine learning algorithms. Most of the features investigated in this study require that parameters be specified for feature extraction and the values for these parameters could potentially have an impact on the performance of the recogniser. Therefore, all feature-based experiments are designed to investigate two aspects related to the use of different features for Bushman handwriting recognition:

1. The effect of varying parameter values during feature extraction.
2. The performance of the different features when used with each recogniser.

In investigating the former of the above, for each machine learning algorithm, each feature is evaluated with different parameters for feature extraction and, in investigating the latter, the best performing parameter values for each feature are compared in order to gain insight into the best overall performing features. The features investigated in this study are:

- Undersampled Bitmaps (UBs)
- Marti & Bunke (M&B) features
- Geometric Moments (GMs)
- Histograms of Oriented Gradients (HoGs)
- Gabor Filter (GF)-based features
- Discrete Cosine Transform (DCT) coefficients

6.2.2 Hybrid Features

Each of the individual features used in this study has descriptive properties. For instance, the GF-based features describe the orientations of lines and the UBs describe the distribution of pixels in the Cartesian plane. It is possible to combine these different features to form hybrid features that incorporate the descriptive properties of the individual features that they are composed of. It is possible that these hybrid features could improve the performance of the recognisers as a result of their potentially increased descriptive capabilities. In investigating the use of hybrid features, the best performing individual features for each recogniser are combined and the combinations are evaluated.

6.2.3 Synthetic Training Data

It is well known that the amount of training data available has an effect on the performance of handwriting recognition systems (Rowley et al., 2002). However, in many cases, large amounts of training data are not available due to the cost of its acquisition. Therefore, some studies have investigated the effect that synthetic training data has on the performance of handwriting recognition systems (Cano et al., 2002). Section 6.1.5 described the creation of synthetic training data and a number of experiments are conducted where the use of synthetic training data for each recogniser is evaluated.

6.2.4 Number of Samples Available

As already mentioned, the performance of handwriting recognition systems is often largely dependent on the number of training samples available. Thus, the performance of the different machine learning algorithms for varying amounts of training data is investigated.

6.2.5 Multiple Authors

It is often desirable for handwriting recognition systems to be writer independent and robust to the differences in the handwriting of multiple authors. The notebooks in the Bleek and Lloyd Collection were written by multiple authors and the corpus described in Section 4 includes the writing of two authors: Lucy Lloyd and Wilhelm Bleek. The effect of having multiple authors is to increase the variation within classes since different authors have different handwriting styles. In this study, the way in which the different machine learning algorithms react to the increase in variation within the training data due to the introduction of a second author is investigated.

6.2.6 Statistical Language Models

Statistical language models, which were discussed in Section 2.6, contain information about the distribution of words or characters in a language. Language models can be used to improve the performance of an automatic recogniser by constraining the recognition output. A language model is built by gathering information about the distribution of words and characters in a language from samples of texts in that language. Thus, the language model actually models the distribution of words and characters in the sample texts and acts as an approximation of the actual distribution of words and characters in the natural language. In this study, the effect of a statistical language model is investigated for a single author. A bigram word language model was created using the SRILM Language Modelling Toolkit (Stolcke, 2002) from the transcriptions of the text lines written by a single author and is used in an experiment in order to investigate the effect that it has on recognition accuracy. It may be possible to build higher order language models; however, for the purpose of this study, a bigram language model was chosen since it is capable of demonstrating the effect that a statistical language model can have on recognition accuracy.

All of the above factors, except for the use of statistical language models, are investigated in various combinations for the different machine learning algorithms. The use of statis-

tical language models is investigated for Hidden Markov Model (HMM)-based text line recognition only.

6.3 Word Recognition

In this study, automatic recognition occurs for individual Bushman words and for Bushman text lines. In this section, Bushman word recognition is discussed and Bushman text line recognition is discussed in the next section. For Bushman word recognition, the machine learning algorithms are provided with an image of a single Bushman word at a time for training and a single Bushman word is recognised at a time. Rather than recognising the individual symbols that make up a word, each Bushman word is recognised as a whole. Thus, each word represents a single pattern that needs to be classified.

A number of machine learning algorithms and descriptive features are used for word recognition. Each of the machine learning algorithms is tested with each feature in order to evaluate its performance. The features are extracted from the words with different parameters in order to gain insight into the effect that feature extraction parameters have on performance. For each machine learning algorithm, the best performing feature is used for further investigation into other aspects affecting word recognition, such as the effect of multiple authors, the use of synthetic data and the number of training samples. Features for the Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs) are extracted for the whole image using the same parameters while features for the HMMs are extracted using a sliding window.

Section 6.1.3 described how cross validation is used for estimating the recognition accuracy. For word recognition, the data set was randomly separated into 10 folds, which were then kept constant for all experiments in order to eliminate differences in recognition accuracy due to sampling.

The metric used to evaluate techniques for word recognition is the word recognition accuracy as defined by Equation 6.1.

This section begins by describing how the corpus described in Chapter 4 was sampled in order to create sub-corpora for word recognition. This is followed by a description of the word normalisation process in Section 6.3.2. The experimental design for the use of SVMs, ANNs and HMMs for Bushman word recognition is then presented.

6.3.1 Corpus Sampling

Chapter 4 described the creation of a corpus of Bushman texts. For the creation of this corpus, pages of text were segmented into their individual text lines, which were then segmented into words and the text lines were then manually transcribed. A shortcoming of this approach is that the transcriptions correspond to the text lines and can not necessarily be mapped directly to the words. For instance, a transcription of a text line may contain 4 words[1] while the image of the text line was only segmented into 3 words. This is an unfortunate shortcoming of having different users do the transcription and segmentation of a text line and of having text lines transcribed rather than words. For this reason,

[1]Where words in a text line are defined as white-space-delimited sets of characters.

only text lines that had the same number of segmented words as the number of words in a transcription were used for word recognition.

For most of the experiments in this study, recognition models were trained for a single author and the experiments were conducted to recognise the texts of that author. However, there are a series of experiments where the handwriting of multiple authors was used for training and recognition. In this case, the recognition models were trained with the handwriting of both authors. For the main author, whose handwriting was used for most of the experiments in this study, a total of 2171 words were found to correspond perfectly to the text line transcriptions. However, for many of the word classes, there was only a single sample, making training models to recognise the word infeasible since the words are recognised as a single object rather than recognising the symbols that they are made up of. For this reason, only word classes that contained 2 or more samples were sampled for word recognition, resulting in a total of 369 word classes. For text line recognition, where individual symbols are recognised, all words were included for recognition. The corpus of 369 word classes was then manually cleaned, resulting in a final corpus containing 257 word classes and a total of 1704 word samples.

6.3.1.1 Minimum Number of Samples Per Class

Some of the experiments described in this chapter relate to the effect that the number of samples per class has on the performance of the recognisers. Therefore, in order to investigate this effect, the corpus containing 257 word classes and 1704 word samples was sampled to create sub-corpora where each sub-corpus had a minimum of n samples per word, where $n \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and the number of word classes also varied among the different sub-corpora. Table 6.1 summarises the properties of these sub-corpora.

Table 6.1: Properties of sub-corpora for minimum number of samples per class for word recognition

Min. Samples Per Class	Number of Classes	Number of Samples
2	257	1704
3	158	1506
4	108	1356
5	81	1248
6	69	1188
7	58	1122
8	52	1080
9	45	1024
10	39	970

6.3.1.2 Fixed Number of Samples and Fixed Classes

The sub-corpora described above all have varying numbers of classes and it is possible that, when investigating how the different machine learning algorithms react to the amount of training data, the number of classes could be the cause of variation in recogniser performance, rather than the number of samples. Thus, another set of sub-corpora was created where the number of classes and number of samples per class were fixed. In these

sub-corpora, the number of classes was fixed at 39 and, for each sub-corpus, the number of samples per word was n, where $n \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

6.3.1.3 Default Word Corpus

A default word corpus is used for most experiments in this study, except for those where the effect of the amount of training data is investigated or where the use of multiple authors is investigated. The default sub-corpus used for recognition is the one where each word class has a minimum of 5 samples. The reason for this is that, in order to get recognition results that accurately represent the descriptive capabilities of the features, it is necessary to have enough samples for training and a minimum of 5 sample was chosen as it was felt that it would be sufficient. The default corpus, which contains a minimum number of 5 samples per word has a total of 81 different word classes and a total of 1248 samples.

6.3.1.4 Multiple Authors Corpus

A number of experiments described in this section deal with the effect that multiple authors have on the performance of the recognisers investigated. The default word corpus, mentioned above, was supplemented with the handwriting of a second author in order to investigate the effect of multiple authors. In supplementing the default word corpus, a total of 247 word samples were added to the 81 classes that already existed. In addition to this, an additional 13 word classes and 109 word samples belonging to these new classes were added. Table 6.2 summarises the differences between the single and multiple author corpora.

Table 6.2: Properties of corpora for single and multiple authors for word recognition

Number of Authors	Number of Classes	Number of Samples
1	81	1248
2	94	1604

6.3.2 Word Normalisation

Normalisation of words refers to the process of preprocessing each word image before training and recognition in order to minimise variation within classes. For each word used for training and recognition, the following normalisation process takes place:

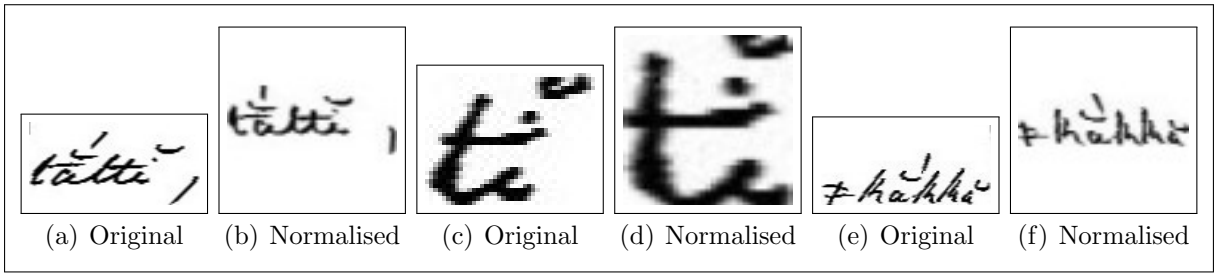
1. A median filter is used to blur the image and then the word boundary is identified and the original image is cropped at the word boundary. This has the effect of removing surrounding noise in the image.

2. Any remaining slant in the word is corrected using the technique described in Section 4.2.3.2.

3. The same process as in Step 1 is repeated in order to remove any noise introduced by the slant correction process.

4. The image is resized to a fixed block of size 64x64. This is done without distorting the proportions by scaling the image and then padding white space to the smaller side. This was necessary as some of the libraries used to implement the machine learning algorithms required that the feature vectors for different samples were the same size. During initial experiments it was found that padding the word images had a negligible effect on the performance of the recognisers.

Examples of normalised words are shown in Figure 6.4.



(a) Original (b) Normalised (c) Original (d) Normalised (e) Original (f) Normalised

Figure 6.4: Bushman words and their normalised variants

6.3.3 Support Vector Machines for Word Recognition

SVMs are one of the machine learning algorithms used in this study and are described in Chapter 2. For word recognition in this study, a SVMs with a Radial Basis Function (RBF) kernel was used. A SVM with a RBF kernel has two parameters - C and λ - which need to be determined before training can begin (Hsu et al., 2003). In order to determine optimal values for these parameters, a grid search is used in which pairs of values for C and λ are tested for the training set in order to see which pair produces the best results. In order to prevent over-fitting, cross validation of the training data is performed in selecting values for C and λ (Hsu et al., 2003).

In this study the LIBSVM (Chang and Lin, 2001) implementation of SVMs is used for scaling the data, performing the grid search, training the models and recognising unseen data and one-vs-one classification is used (see Section 2.5.3).

6.3.3.1 Experiment Procedure

For every SVM-based experiment the following steps take place:

1. The full data set is separated into 10 folds of equal size.
2. The data is scaled.
3. Cross validation is used where 9 folds are used for training and 1 fold is used for testing.
4. The optimal values for C and λ for the training data are found using grid search and an SVM model is trained.
5. The accuracy is recorded for the testing fold using the SVM model created in the previous step.

6. The average accuracy for the 10-folds, as described by Equation 6.1, is reported as the final accuracy.

6.3.4 Artificial Neural Networks for Word Recognition

ANNs were described in Chapter 2 and are one of the machine learning algorithms used in this study. In this study a multilayer perceptron is used for word recognition and is discussed in this section.

6.3.4.1 Multilayer Percepton Implementation

The topology of the multilayer perceptron used in this study has an input layer, a hidden layer and an output layer, thereby making it a 3-layer ANN. The number of nodes in the input layer is equal to the number of descriptors in the feature vector and the number of neurons in the output layer is equal to the number of classes that need to be recognised. The selection of the number of neurons in the hidden layer is described in the next section. Figure 6.5 shows an example of the type of ANN architecture used in this study.

The ANNs used in this study are implemented using the Fast Artificial Neural Network (FANN) library (Nissen, 2003).

Network Parameters When training ANNs, there are a number of parameters that need to be specified, including: the number of epochs, the learning rate, the learning momentum, the activation function, the error function and the training algorithm. The parameters in this study are fixed as follows:

- Epochs: 1000
- Activation function: sigmoid symmetric (tanh)
- Error function: tanh
- Training algorithm: iRprop (Christian Igel, 2000) - does not require a learning rate (an improved implementation of the Rprop algorithm (Riedmiller, 1994))

The only ANN parameter that is varied in this study is the number of neurons in the hidden layer. The best number of units in the hidden layer depends on several factors such as the number of units in the input and output layers, the amount of noise that exists in the the data, the training algorithm used and the activation function (Sarle, 1997). Sarle (1997) notes that there is no way to easily determine the best number of neurons in a hidden layer, where too few hidden units will result in underfitting and high generalisation and training errors, whereas too many units, while having a low training error, will result in overfitting. There are a few "rules of thumb" for choosing the number of units in the hidden layer, which Sarle argues are flawed because they ignore some of the aspects that the number of hidden units depend on and which were mentioned above. These "rules of thumb" include the following listed by Sarle from various sources:

- The number of hidden units should be between the number of units in the input layer and the output layer.

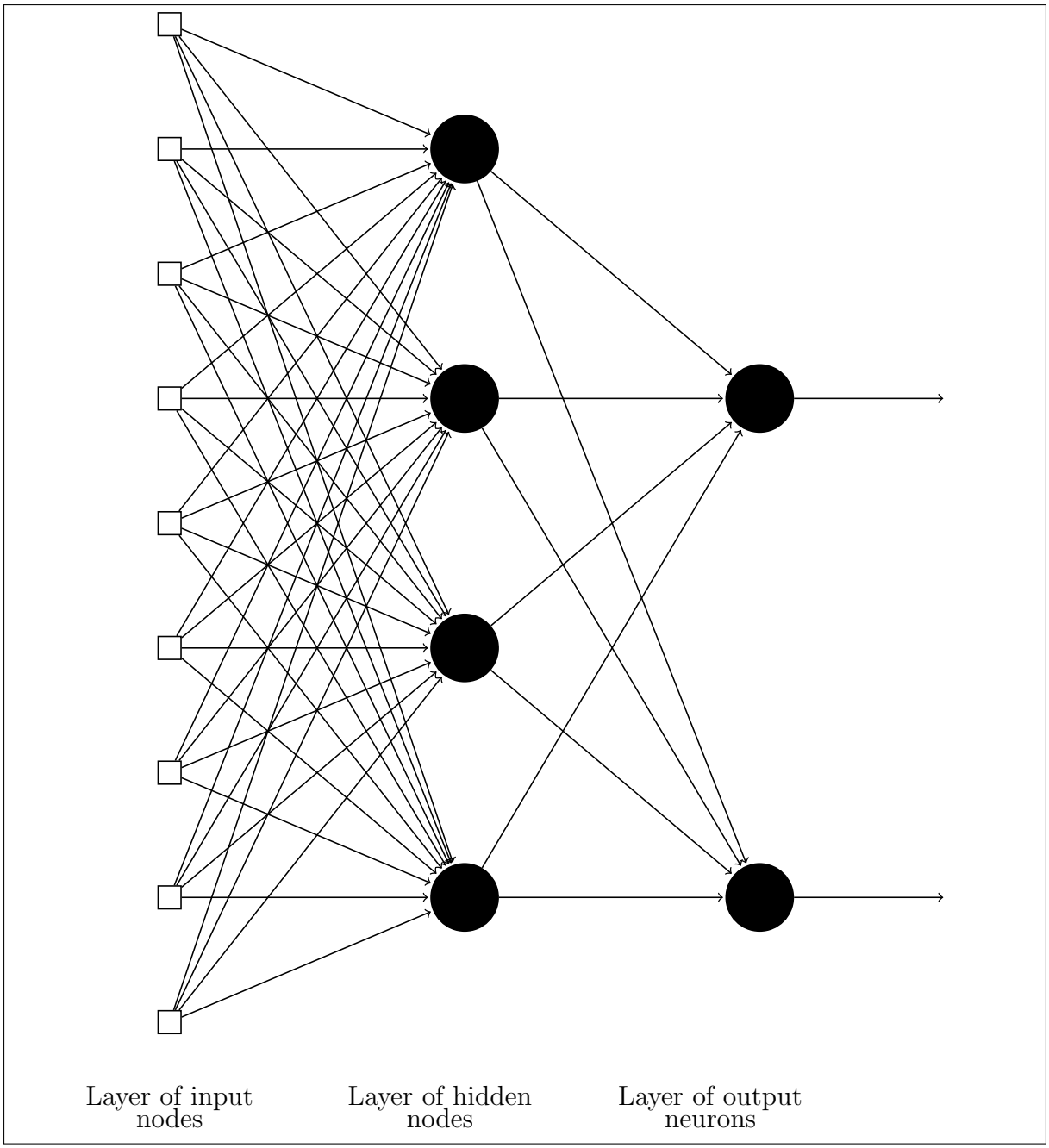


Figure 6.5: A multi-layered feed-forward network

- The number of hidden units can be calculated as:

$$(\text{size of input layer} + \text{size of output layer}) \times \frac{2}{3}.$$

- The size of the hidden layer should never be more than twice the size of the input layer.

Given the dependence of the performance of a ANN on the number of hidden units as well as the varying opinions on how many hidden units there should be, an approach to finding the best number of hidden units is just to train the ANN with different numbers of hidden units, estimate the error for each and use the best performing architecture. This is the

approach used in this study. For each ANN experiment, ANNs are built with N neurons in the hidden layers, where $N = \{S_I, \frac{S_I}{2}, S_O, \frac{S_O}{2}, \frac{S_I + S_O}{2}\}$, where S_I is the size of the input layer and S_O is the size of the output layer. The experiment is then conducted for each architecture and the accuracy of the best performing architecture is reported.

Neural Network Ensembles When recognising unseen data, the output at each neuron in the output layer is the strength of the signal in the range [-1;1] due to the sigmoid symmetric activation function on the output layer. A high positive value implies a strong correlation with the class represented by that neuron, while a low signal implies little correlation. However, since the ANNs have been initialised by random weights, these random values can have an impact on the performance of the ANN, i.e. "good" initial random weight can result in good performance while "bad" initial weights might lead to poor performance. An approach to overcoming this random effect is to use an ensemble of redundant ANNs. In an ensemble of ANNs a finite number of ANNs are trained for the same task and then their output is combined to create a single output (Chang Shu, 2004). The effect of this is to improve the generalisation of the ensemble as a whole. There are a number of different methods for creating ensembles, such as having networks with different architectures, varying the training algorithm and data set, and varying the initial random weights (Chang Shu, 2004).

Once the ensemble of networks has been trained, their output is combined to create a single output. Common techniques for doing this include linear methods, such as averaging or weighted averaging or stacking, in which an additional model learns how to combine the networks in the ensemble (Chang Shu, 2004).

For each experiment in this study, an ensemble of 10 ANNs is created. The ANNs in the ensemble differ by the random initialisation weights and are combined by averaging the weights at each output neuron for the 10 networks in the ensemble.

6.3.4.2 Experiment Procedure

For every ANN-based experiment for word recognition the following steps take place:

1. The same 10 folds of data as used in the experiments for SVM-based word recognition are used.
2. Cross validation is used, where 9 folds are used for training and 1 fold is used for testing.
3. An ensemble of 10 ANNs is trained for each ANN architecture, where each architecture is defined by N, the number of neurons in the hidden layer, where $N = \{S_I, \frac{S_I}{2}, S_O, \frac{S_O}{2}, \frac{S_I + S_O}{2}\}$.
4. The average output of the ensemble is calculated and used for classification.
5. The word accuracy, as described by Equation 6.1, is recorded and averaged for the 10 folds for the best performing ANN architecture.

6.3.5 Hidden Markov Models for Word Recognition

HMMs were described in Chapter 2 and are one of the machine learning algorithms used in this study for word recognition. In this study, HMMs are used for word recognition and a single HMM is built for each word. This study makes use of the Hidden Markov Model Toolkit (HTK) (Young et al., 2006) for HMM-based recognition. Even though HTK was originally designed for speech recognition, it can still be used for general purpose HMM-based recognition. All HMMs is this study are left-to-right continuous models and are defined by two main parameters that need to be determined before training and recognition can begin: the number of states in the model and the number of Gaussians. Section 7.2.1.1 describes an experiment for determining these parameters for text-line recognition and, since the words that are recognised in this study are extracted from the lines recognised in this study, it is assumed that these parameters are also suitable HMM-based word recognition.

6.3.5.1 Experiment Procedure

For every HMM-based experiment for word recognition, the following steps take place:

1. The same 10 folds of data as used in the experiments for SVM and ANN-based word recognition are used.
2. Cross validation is used, where 9 folds are used for training and 1 fold is used for testing.
3. HMMs with 12 emitting states are created for each word in the training set (see Section 7.2.1.1).
4. Beginning with 2 Gaussians, training takes place on the training set over four iterations.
5. The number of Gaussians is increased by 2 using the results of the previous training step as the base for the next until HMMs with 16 Gaussians have been trained.
6. The accuracy is recorded for the testing fold using the HMMs created in the previous step .
7. The word accuracy, as described by Equation 6.1, is recorded and averaged for the 10 folds.

This section discussed the experimental design for the recognition of Bushman words. SVMs, ANNs and HMMs are used for Bushman word recognition where the task is to recognise whole Bushman words as a single pattern rather than recognise the individual symbols that they are made up of. One of the disadvantages to word recognition is that text lines need to be segmented into words and, in this study, this was done semi-automatically. However, this process could significantly increase the difficulty in creating a recogniser. Thus, in the next section, the avoidance of word segmentation by using HMMs for text line recognition is described.

6.4 Text Line Recognition

The previous section discussed the research design of experiment for Bushman word recognition, where each of the words was recognised as a single pattern. This section describes the research design for text line recognition where each of the symbols in a text line is recognised individually and then combined to form words and sentences. The term "text line recognition" is used since the input to the recogniser is a line of text; however, the actual atomic units being recognised are the individual symbols.

A feed forward continuous HMM is used for text line recognition with the features described in Chapter 5, all of which have been extracted multiple times with different parameters. The best feature set is selected for further investigation into other factors that impact on text line recognition, such as the use of synthetic data, the effect of multiple authors and the amount of training data available.

As was the case with word-based recognition, cross validation as discussed in Section 6.1.3 is used in order to minimise selection bias.

All text lines are normalised to a height of 64 and the recognition accuracy, as described by Equation 6.2, is used as a performance metric for text line recognition.

6.4.1 Corpus Sampling

The corpus of text lines from Chapter 4 was sampled to create a set of sub-corpora for Bushman text line recognition. 3083 of the transcribed text lines from the Bushman corpus were used in this study, 1928 of which were written by Lucy Lloyd and the remaining 1155 written by Wilhelm Bleek. In this study, this corpus of 3083 text lines was sampled in various ways for the different experiments involving text lines. The ways in which this corpus was sampled are described below.

6.4.1.1 Default Corpus

The default corpus, which is used in most text line recognition experiments, was based on the writings of a single author. In this default corpus, the 1928 Lucy Lloyd text line samples were used for training and recognition. In this corpus, there are a total of 963 symbols, for which models are trained in the experiments. These symbols are used to form 3925 different words in the text line samples.

6.4.1.2 Amount of Training Data

One of the experiments for text line recognition relates to the amount of training data that is available for training. In investigating this, the default corpus mentioned above was randomly sampled such that P percent of the corpus was used for training and testing where $P \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

6.4.1.3 Minimum Number of Sample Per Class

Sub-corpora were created to investigate the effect that the minimum number of samples per class has on the performance of the recogniser where only text lines where all of the symbols in the text line had a minimum of n samples were used, where $n \in \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. In order to increase accuracy when determining the minimum number of samples per symbol, only text lines where the number of words in the transcription was equal to the number of words that text line was segmented into were used. In doing this, the probability of the transcription being accurate was increased since two data capturers would need to agree on the number of words when segmenting and transcribing a text line. Table 6.3 summarises the properties of each of these sub-corpora.

Table 6.3: Properties of sub-corpora for minimum number of samples per class for text line recognition

Min. Samples Per Class	Number of Classes	Number of Samples
2	367	1418
3	260	1222
4	208	1100
5	172	981
6	154	910
7	132	801
8	120	736
9	116	714
10	113	692

6.4.1.4 Multiple Authors

One of the experiments in this study describes an investigation into the effect that multiple authors has on the performance of the text line recogniser. In order to investigate this, the default corpus described above is supplemented with the writings of a second author. Table 6.4 summarises the differences between the single and multiple author corpora.

Table 6.4: Properties of corpora for single and multiple authors for text line recognition

Number of Authors	Number of Classes	Number of Samples
1	963	1928
2	1231	3083

6.4.2 Hidden Markov Models for Text Line Recognition

HMMs were described in Chapter 2. In this study, HMMs are used for text line recognition in a similar fashion to that of word recognition. However, a difference exists in the way that models are built and recognised. For word recognition, models were built for every word in the text and then individual words were recognised while a network constrained the recognition output to a single word.

A slightly different approach is taken for text line recognition where models are built for each symbol that appears in the text. These models are then concatenated to form words through the use of a dictionary. Furthermore, instead of only allowing a single model as output, as was the case with word recognition, the network allows for any number of words to be output.

Features for HMM-based text line recognition are extracted in the same manner as for word recognition, i.e. using a sliding window of width W with possible overlap O (see Section 5.1.2).

6.4.2.1 Experiment Procedure

For every HMM-based experiment for text line recognition (except the one when the parameters for the HMMs are determined where Step 3 is omitted), the following steps take place:

1. The data is randomly separated into 10 folds that are then kept constant for all experiments.
2. Cross validation is used where 9 folds are used for training and 1 fold is used for testing.
3. HMMs with 12 emitting states are created for each symbol in the training set.
4. Beginning with 2 Gaussians, training takes place on the training set over four iterations.
5. The number of Gaussians is increased by 2 while building on the previous training phase until HMMs with 16 Gaussians have been trained.
6. The recognition accuracy, as described by Equation 6.2, is recorded and averaged for the 10 folds.

6.5 Discussion

There are many choices that can be made when designing a handwriting recognition system. These choices relate to the choice of machine learning algorithms, descriptive features, segmentation levels and amount of training data, among others. All of these factors could potentially have an effect on the performance of a handwriting recognition system and thus it is beneficial to use an approach that results in the best performance. In this study, two approaches to Bushman handwriting recognition are investigated. In the first approach, whole Bushman words are recognised as a single pattern and, in the second approach, the individual symbols in Bushman text lines are recognised. This chapter has described the experimental design used in this study, which allows for a direct comparison of different techniques to be made. In the next chapter, the experiments in which these different techniques for handwriting recognition were investigated are presented and their results analysed.

Chapter 7

Experiments

Chapter 6 described the research design used in this study, including a discussion of how Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Hidden Markov Models (HMMs) are investigated for word recognition and how HMMs are investigated for text line recognition. The factors that are investigated as part of this study include:

- The use of different features.
- The use of hybrid features.
- The use of synthetic data.
- The effect of the amount of training data.
- The effect of multiple authors.
- The use of a statistical language model.

In this chapter, the experiments conducted to investigate each of these factors is presented and their results discussed. For each factor, the findings are presented within the context of the machine learning algorithm being investigated. However, a meta-analysis is provided later in the chapter based on the findings for all of the machine learning algorithms. This chapter begins with a description of experiments for word recognition, then a description of the experiments for text line recognition and, lastly, an analysis of the overall findings and implications.

7.1 Word Recognition

Words were one of the units of recognition considered in this study and this section presents the results of using SVMs, ANNs and HMMs for word recognition.

7.1.1 Support Vector Machines for Word Recognition

In this section, experiments investigating the use of SVMs for Bushman word recognition for the following are presented:

- The use of different descriptive features in Section 7.1.1.1.
- The use of hybrid features in Section 7.1.1.2.
- The effect of synthetic training data in Section 7.1.1.3.
- The effect of the number of training samples in Section 7.1.1.4.
- The effect of multiple authors in Section 7.1.1.5.

The SVM experimental procedure described in Section 6.3.3.1 is used for all SVM-based word recognition experiments. For all experiments, except the one investigating the effect of multiple authors, the default corpus containing the handwriting of a single author, as described in Section 6.3.1, is used. The recognition accuracy reported is based on Equation 6.1 and is a measure of the proportion of words that were correctly classified.

7.1.1.1 Experiment: Support Vector Machines with Different Features

Purpose To investigate the performance of a SVM-based word recogniser with various descriptive features.

Procedure

1. Features are extracted for each of the features described in Chapter 5 using varying parameters.
2. The best performing parameter values for each feature are then used to provide a comparison of the different features for SVM-based word recognition.

Results

Undersampled Bitmaps

Figure 7.1 shows the recognition accuracy for SVM-based word recognition using Undersampled Bitmaps (UBs) as features. The recognition accuracy is shown when the word images are partitioned into different sized cells. As can be seen from Figure 7.1, decreasing the cell size, and thereby increasing the size of the feature vector, has a positive effect on the recognition accuracy, which is highest at 50.73% when UBs are extracted from 8x8 cells. The findings suggest that the descriptive power of UBs increases when the cell size decreases. However, when the cell size becomes too small, as is the case for the 4x4 cells, UBs appear to lose some of their descriptive power.

Marti & Bunke Features

The average word recognition accuracy when using the Marti & Bunke (M&B) set of features for SVM-based Bushman word recognition was 47.28%.

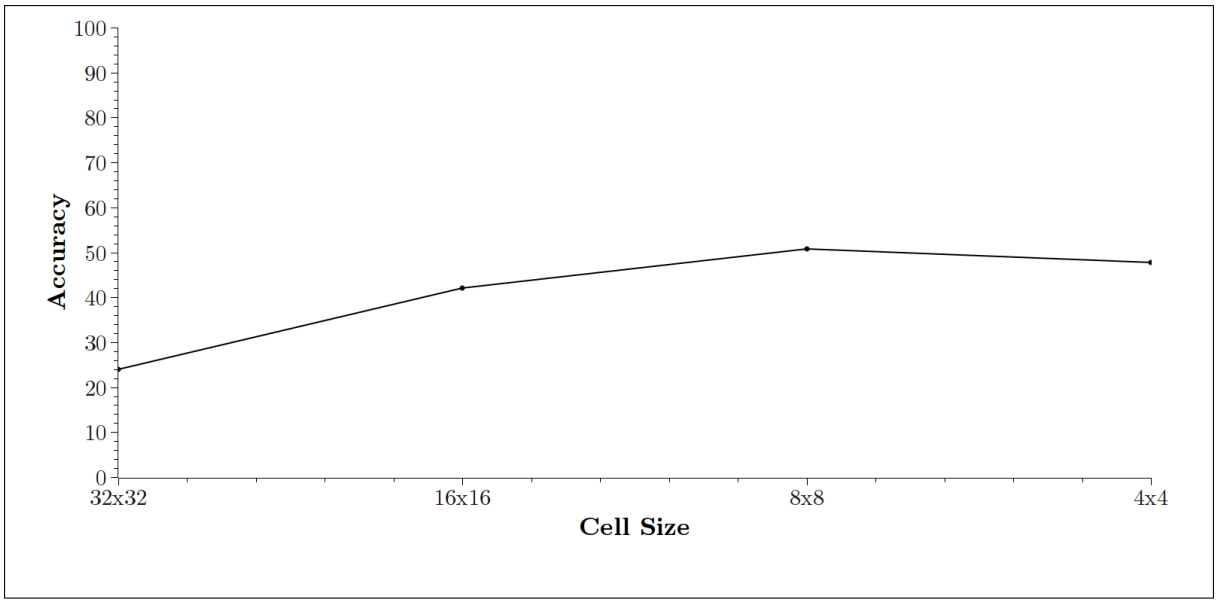


Figure 7.1: Recognition accuracy for SVM-based word recognition using UBs as features

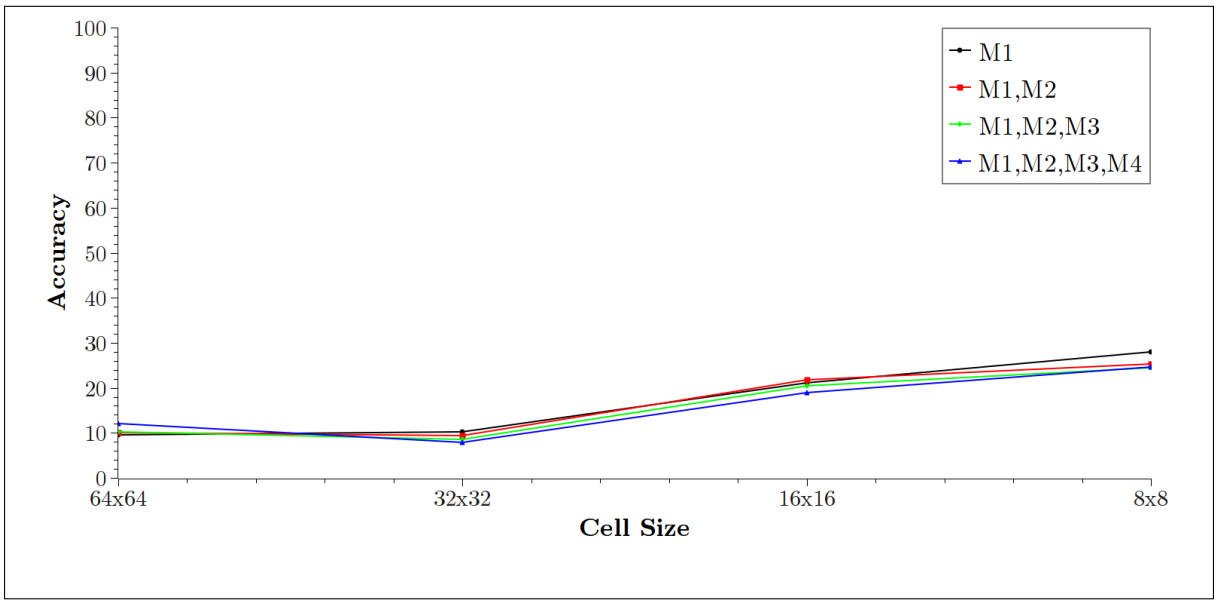


Figure 7.2: Recognition accuracy for SVM-based word recognition using GMs as features

Geometric Moments

Figure 7.2 shows the recognition accuracy for SVM-based word recognition using Geometric Moments (GMs) as features. The first thing to notice from the figure is that the GMs that are selected for use as features appear to have little impact on the recognition accuracy achieved. Instead, it appears that the size of the cells that the images are partitioned into has a larger impact. For instance, extracting GMs from whole images or from images partitioned into 32x32 cells results in poor performance, while decreasing the size of the cells that the images are partitioned into improves performance. These findings are in line with the findings when UBs were used as features where decreasing the size of the cells led to an improvement in performance. The best recognition accuracy of 27.96% occurs when 8x8 cells are used and when only the first GM (M1) is used as a feature.

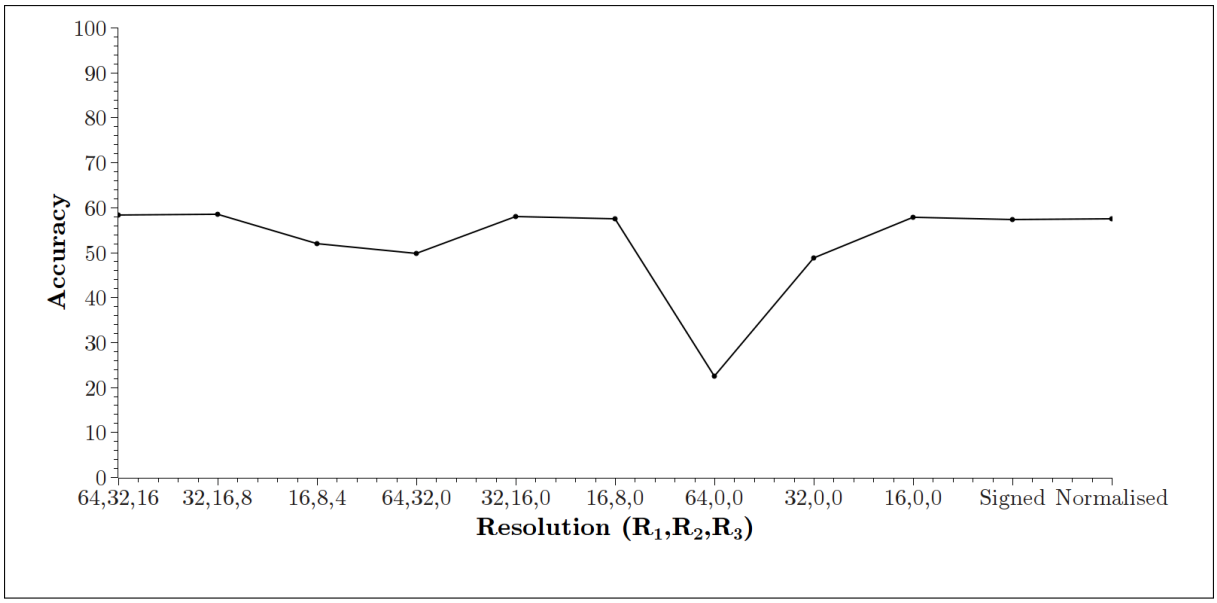


Figure 7.3: Recognition accuracy for SVM-based word recognition using HoGs as features

Histogram of Oriented Gradients

Figure 7.3 shows the word recognition accuracy for Histograms of Oriented Gradients (HoGs) extracted with various parameters. The first labels along the x-axis are the resolutions at which the features were extracted and, as can be seen from the figure, the best word recognition accuracy of 58.49% was achieved when resolutions R_1, R_2 and R_3 were equal to $32, 16$ and 8 respectively. However, all resolutions tested appear to have performed relatively well except when $R1, R2$ and $R3$ were equal to $64, 0$ and 0. In this case HoGs were extracted from whole images without them being partitioned into cells and it appears that, at this high resolution, the features lack the descriptive power that occurs at lower resolutions. The fact that, in most cases, the resolutions at which the HoGs were extracted has little effect on the accuracy achieved suggests that the same information is summarised by the HoGs at all resolutions. As can be seen from the figure, signing and normalising the gradients led to a slight decrease in accuracy.

Gabor Filter-based Features

Figure 7.4 shows the word recognition accuracy for Gabor Filter (GF)-based features when the images are partitioned into varying numbers of cells for salient point calculation. As can be seen from the figure, the best word recognition accuracy of 53.21% occurs when the images are partitioned into 8x8 cells for calculating the relative number of salient points. The effect of the number of cells that the images are partitioned into in this experiment is in line with the findings when UBs and GMs were used as features where decreasing the size of the cells from which features were extracted led to an improvement in performance up to some point afterwhich performance started to decrease.

Discrete Cosine Transform Coefficients

Figure 7.5 shows the word recognition accuracy when the Discrete Cosine Transform

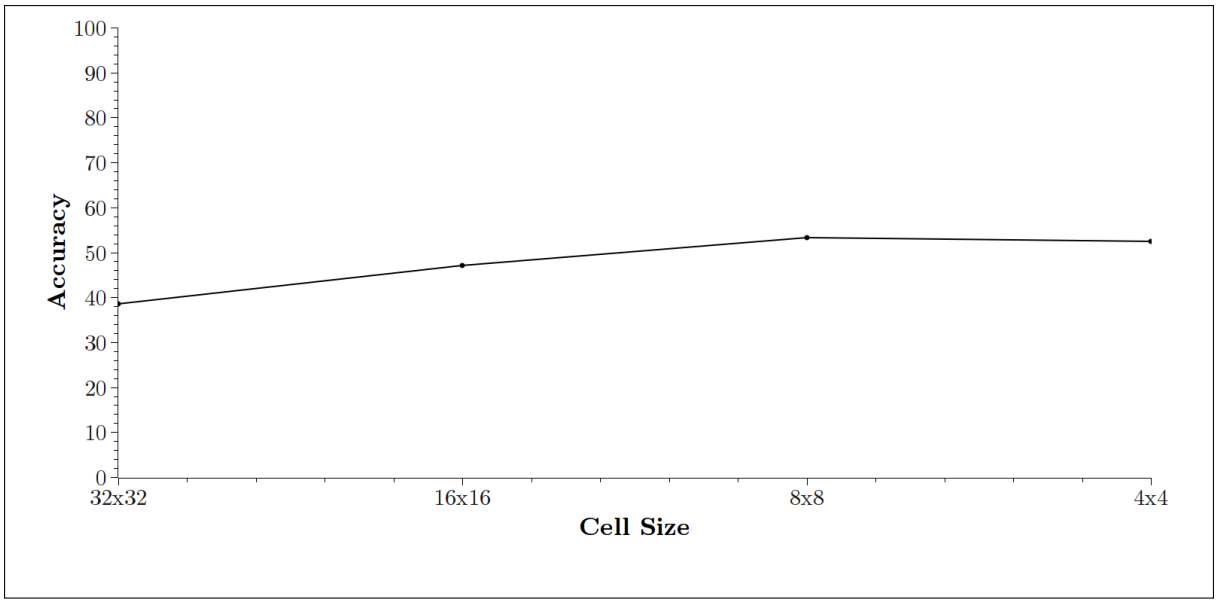


Figure 7.4: Recognition accuracy for SVM-based word recognition using GF-based features

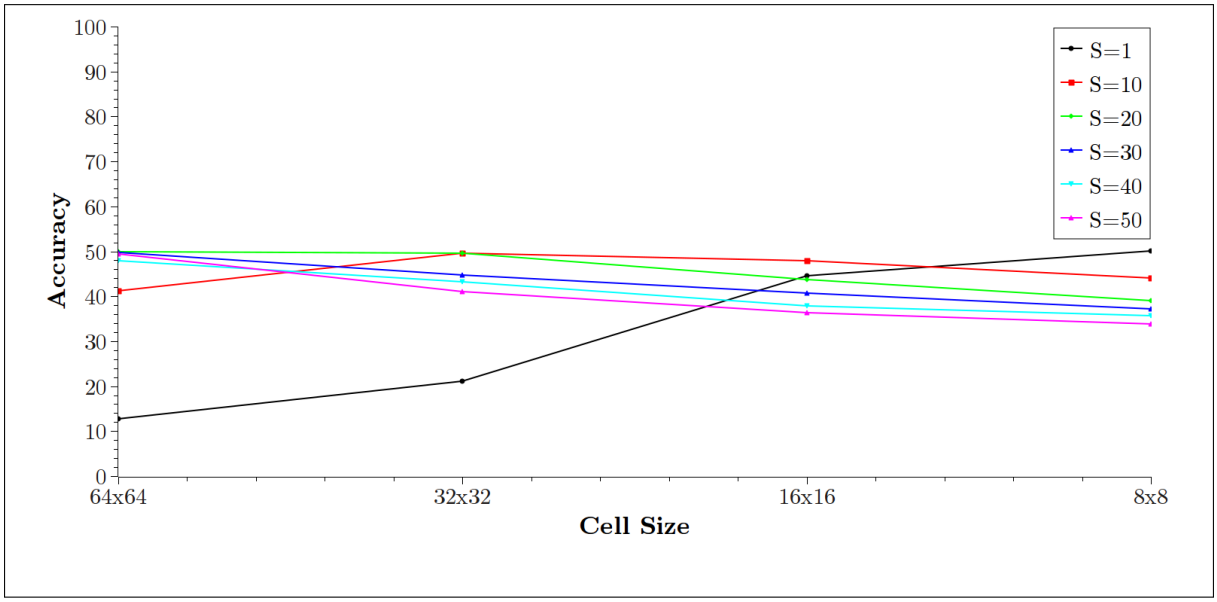


Figure 7.5: Recognition accuracy for SVM-based word recognition using DCT coefficients as features

(DCT) coefficients are used as features. Two parameters are specified when extracting DCT features: the size of the cells on which the DCT should be performed and the number of coefficients to extract as features. As can be seen from Figure 7.5, having only a few descriptors, as is the case when only one DCT coefficient is extracted from a whole image or when the size of the cells that the images are partitioned into are large, results in poor performance. Beyond that, the performance appears to be relatively uniform regardless of the number of coefficients extracted for each cell size that the images were partitioned into for the DCT calculation. The best recognition accuracy of 50.16% was achieved when the DCT was performed on 8x8 cells and only a single DCT coefficient was extracted per cell.

Discussion There are a number of descriptive features that can be used when building a SVM-based word recogniser and this experiment has investigated the use of 6 different descriptive features. It has been shown that the choice of descriptive features has a strong influence on the performance of the SVM-based word recogniser. For instance, the highest recognition accuracy of 58.49% was achieved when HoGs were used as features and the lowest recognition accuracy of 27.96% occurred when GMs were used as features. The other descriptive features all achieved a recognition accuracy in the 50% range. Table 7.1 provides an overview of the maximum recognition accuracy for each feature with the best performing feature extraction parameters.

Table 7.1: Recognition accuracy for SVM-based word recognition for all descriptive features with best performing parameter values

Feature Set	Recognition Accuracy
Histograms of Oriented Gradients	58.49%
Gabor Filter-based Features	53.21%
Undersampled Bitmaps	50.73%
Discrete Cosine Transform Coefficients	50.16%
Marti & Bunke Features	47.27%
Geometric Moments	27.96%

All of the features used in this study, except the M&B features, required that parameters be specified for feature extraction. Thus, this experiment involved investigating various feature extraction parameters and it was found that the choice of these parameters often had an effect on the recognition accuracy that could be achieved. For instance, UBs, GMs, GF-based features and DCT coefficients all required that the size of the cells that an image is partitioned into for feature calculation be specified. In each of these cases, size of the cells ranged from a single cell with size 64x64, to the smallest cell size of 4x4. In all cases it was found that the 8x8 cell sizes resulted in the best performance, while smaller and larger cell sizes led to a decrease in accuracy. These finding suggest that, to some extent, smaller cells have higher descriptive power at their smaller local levels than larger cells. At some point though (in this study when the images were partitioned into 4x4 cells) there was a decrease in performance, suggesting that, at very small local levels, the descriptive power of the features begins to decrease. There were also some cases where the parameters for feature extraction appeared to have little effect on the performance of the recogniser. For instance, for GMs, the actual moments that were extracted as features did not influence the recognition accuracy. Similarly, for DCT coefficients, the number of coefficients extracted per cell also appeared to have little effect on recognition accuracy.

7.1.1.2 Experiment: Support Vector Machine with Hybrid Features

Purpose To investigate the accuracy that can be achieved when a SVM-based recogniser is used with hybrid features for Bushman word recognition.

Procedure

- HoGs, GF-based features and UBs, which were found to produce the best results in the previous experiment, are combined to create hybrid features made up of n

individual features at a time where $n \in \{2, 3\}$ resulting in a total of 4 different hybrid features.

- The features are extracted using the best performing parameters in the previous experiment.

Results Figure 7.6 shows the recognition accuracy for the various hybrid features used with the SVM-based recogniser.

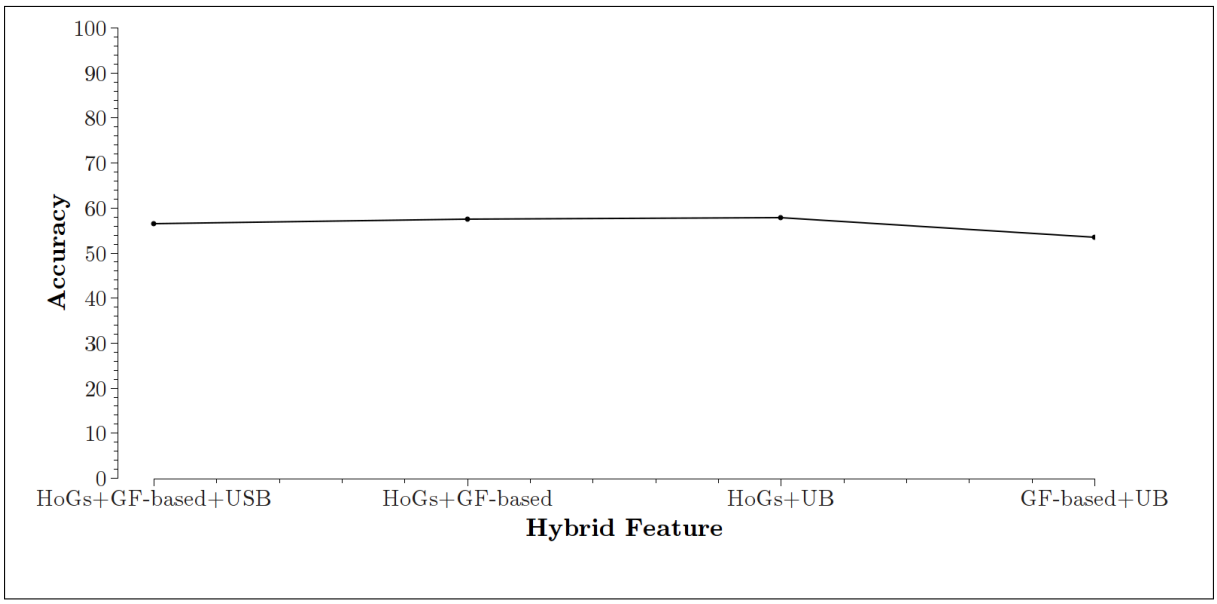


Figure 7.6: Recognition accuracy for SVM-based word recognition using hybrid features

Discussion Figure 7.6 shows that all of the hybrid features used in this experiment perform approximately the same. The maximum recognition accuracy of 57.77% was achieved when HoGs were combined with UBs. However, this best performing hybrid feature performs slightly worse than the HoGs do individually. There is a very slight improvement in performance when GF-based features are supplemented by UBs; however, it is unlikely that this is significant. Thus these findings show that the hybrid features used in this study do not improve the performance of the SVM-based recogniser and could possibly apply to hybrid features in general.

7.1.1.3 Experiment: Support Vector Machine with Synthetic Training Data

Purpose To investigate the accuracy that can be achieved when a SVM-based recogniser is trained with supplemental synthetic data for Bushman word recognition.

Procedure

- The transformations described in Section 6.1.5 are applied to each training sample to create 4 synthetic versions of each training sample.
- HoGs are used as features since they had the highest recognition accuracy in previous experiments.

Results By including the transformations described in Section 6.1.5, an average word recognition rate of 62.58% was achieved.

Discussion The 62.58% recognition accuracy achieved with synthetic data is an increase by 4.09% in the recognition accuracy with the best performing HoGs as features. Given the simplicity in creating this synthetic training data, it can be concluded that its use is beneficial. However, it is possible that the improvement gained from using synthetic training data does not generalise. Thus, this experiment is also repeated for the other machine learning algorithms and then analysed in greater detail in Section 7.3.3.

7.1.1.4 Experiment: Support Vector Machine and Number of Samples

Purpose To investigate the effect that the number of training samples has on the performance of a SVM-based word recogniser.

Procedure

- The sub-corpora described in Section 6.3.1 are used.
- HoGs are used as features since they had the highest recognition accuracy in previous experiments.

Results The results of this experiment are shown in Figure 7.7.

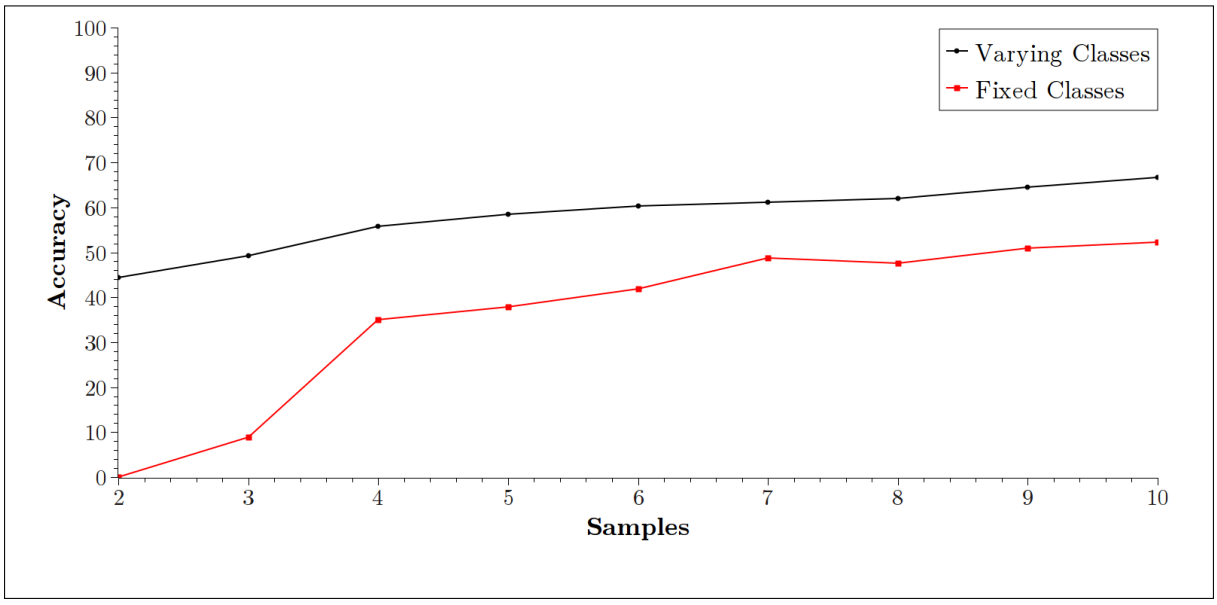


Figure 7.7: Recognition accuracy for SVM-based word recognition for different numbers of training samples

Discussion The effect of the number of samples on the performance of the SVM-based recogniser has been tested in two ways in this experiment. In the first case, represented by the *varying classes* line in Figure 7.7, the minimum number of samples per class is represented by the x-axis. As can be seen from the figure, there is a positive linear relationship between the minimum number of samples per class and recognition accuracy. This, however, needs to be considered within the context of the number of classes that need to be recognised. For instance, when there are a minimum of two samples per class, a total of of 257 classes need to be recognised whereas, when there are a minimum of 10 samples per class, only 39 classes need to be recognised. This occurs due to the higher minimum number of samples per class excluding classes with fewer samples. Therefore, it is possible that the difference in recognition accuracy may be attributable to the number of classes that need to be recognised, rather than the number of available samples. The *fixed classes* line in Figure 7.7 addresses this by fixing the number of classes that need to be recognised to 39 and then fixing the number of samples per class (as opposed to setting a minimum). The fixed classes results show the same general positive linear relationship between the number of samples and recognition accuracy as the varying classes results. Thus, it can be concluded that increasing the number of training samples is beneficial in improving the performance of the SVM-based word recogniser.

7.1.1.5 Experiment: Support Vector Machine for Multiple Authors

Purpose To investigate the effect that multiple authors have on the performance of a SVM-based word recogniser.

Procedure

- The corpus described in Section 6.3.1 containing the handwriting of two authors is used for training and recognition.
- HoGs are used as features since they had the highest recognition accuracy in previous experiments.

Results The word accuracy when recognising words from multiple authors was 55.38%.

Discussion A SVM that is able to recognise the handwriting of multiple authors is desirable. The results from this experiment show that recognising the handwriting of multiple authors led to a decrease in recognition accuracy from 58.49% to 55.38%, a decrease of 3.09%. This decrease seems relatively small when the benefit is an SVM recogniser capable of recognising the handwriting of multiple authors.

7.1.2 Artificial Neural Networks for Word Recognition

The previous section presented the results of experiments in which a SVM-based recogniser was used for Bushman word recognition. In this section, experiments investigating the use of ANNs for Bushman word recognition for the following are presented:

- The use of different descriptive features in Section 7.1.2.1.
- The use of hybrid features in Section 7.1.2.2.
- The effect of synthetic training data in Section 7.1.2.3.
- The effect of the number of training samples in Section 7.1.2.4.
- The effect of multiple authors in Section 7.1.2.5.

In addition, an analysis of ANN architectures is presented in Section 7.1.2.6.

The ANN experimental procedure described in Section 6.3.4.2 is used for all ANN-based word recognition experiments. For all experiments, except the one investigating the effect of multiple authors, the default corpus containing the handwriting of a single author, as described in Section 6.3.1, is used. The recognition accuracy reported is based on Equation 6.1 and is a measure of the proportion of words that were correctly classified.

7.1.2.1 Experiment: Artificial Neural Network with Different Features

Purpose To investigate the performance of an ANN-based word recogniser with various descriptive features.

Procedure

1. Features are extracted for each of the features described in Chapter 5 using varying parameters.
2. The best performing parameter values for each features are then used to provide a comparison of the different features for ANN-based word recognition.

Results

Undersampled Bitmaps Figure 7.8 shows the results of recognising Bushman words using an ANN-based recogniser trained with UBs. As can be seen from the figure, there is a relationship between the size of the cells that the word images are partitioned into and recognition accuracy. In this experiment, the recognition accuracy increases as the size of the cells that the images are partitioned into decreases until it hits its peak when the images have been partitioned into 8x8 cells and the recognition accuracy is 46.71%. Beyond this point, recognition accuracy begins to decrease again. The same effect was observed when UBs were used for SVM-based recognition.

Marti & Bunke Features The average word recognition accuracy when using the M&B features for ANN-based Bushman word recognition was 40.30%.

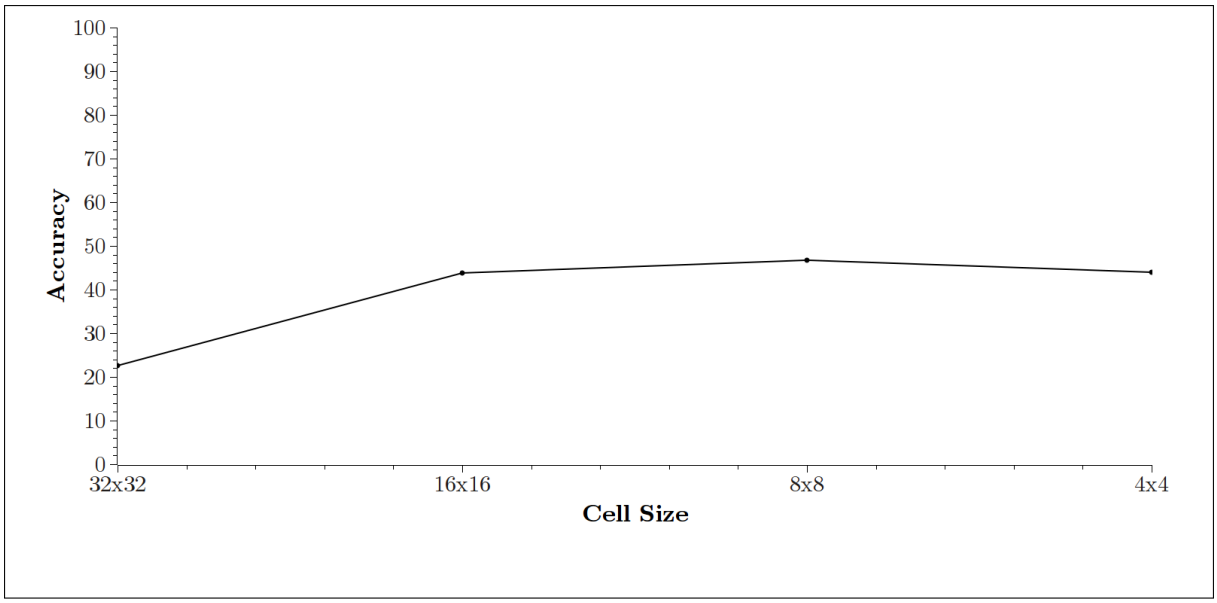


Figure 7.8: Recognition accuracy for ANN-based word recognition using UBs as features

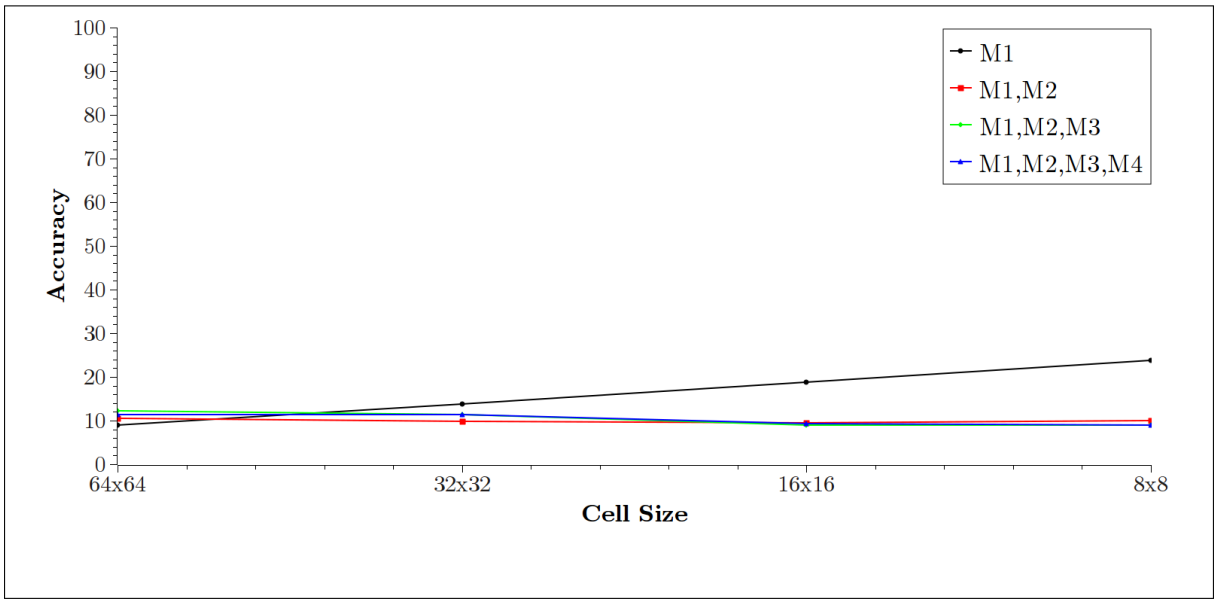


Figure 7.9: Recognition accuracy for ANN-based word recognition using GMs as features

Geometric Moments Figure 7.9 shows the word recognition accuracy when GMs are used as features for the ANN-based recogniser. As can be seen from the figure, when more than the first GM (M1) is used as a feature, the size of the cells that a word image is partitioned into appears to have little effect on recognition accuracy. When only the first GM is used, partitioning a word image into smaller cells results in a higher recognition accuracy. Overall, however, performance is quite poor with a maximum recognition accuracy of 23.79% occurring when the images are partitioned into 8x8 cells and only the first GM (M1) per cell is used as a feature.

Histograms of Oriented Gradients Figure 7.10 shows the recognition accuracy when HoGs are used as features for ANN-based Bushman word recognition. As can be seen from the figure, the resolutions at which the features are extracted has an effect on

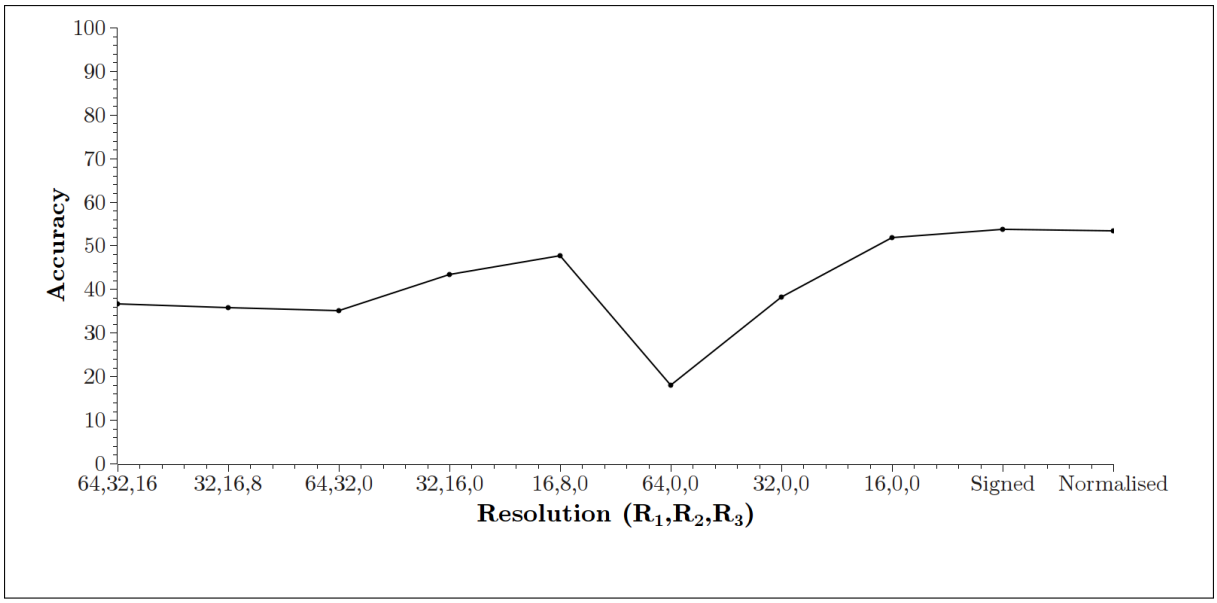


Figure 7.10: Recognition accuracy for ANN-based word recognition using HoGs as features

the recognition accuracy. For instance, excluding signed and normalised gradients, the maximum recognition accuracy of 51.76% is achieved when $R1, R2, R3 = \{16, 0, 0\}$ and the lowest recognition accuracy of 16.01% occurred when $R1, R2, R3 = \{64, 0, 0\}$. The occurrence of this minimum recognition accuracy corresponds with that for SVM-based word recognition where extracting features at a single resolution equal to the size of the word images resulted in the poorest performance. When the $R1, R2, R3 = \{16, 0, 0\}$ and the features features were signed, recognition accuracy increased to 53.76%, and normalising the signed gradients leads to a slight reduction in recognition accuracy to 53.36%.

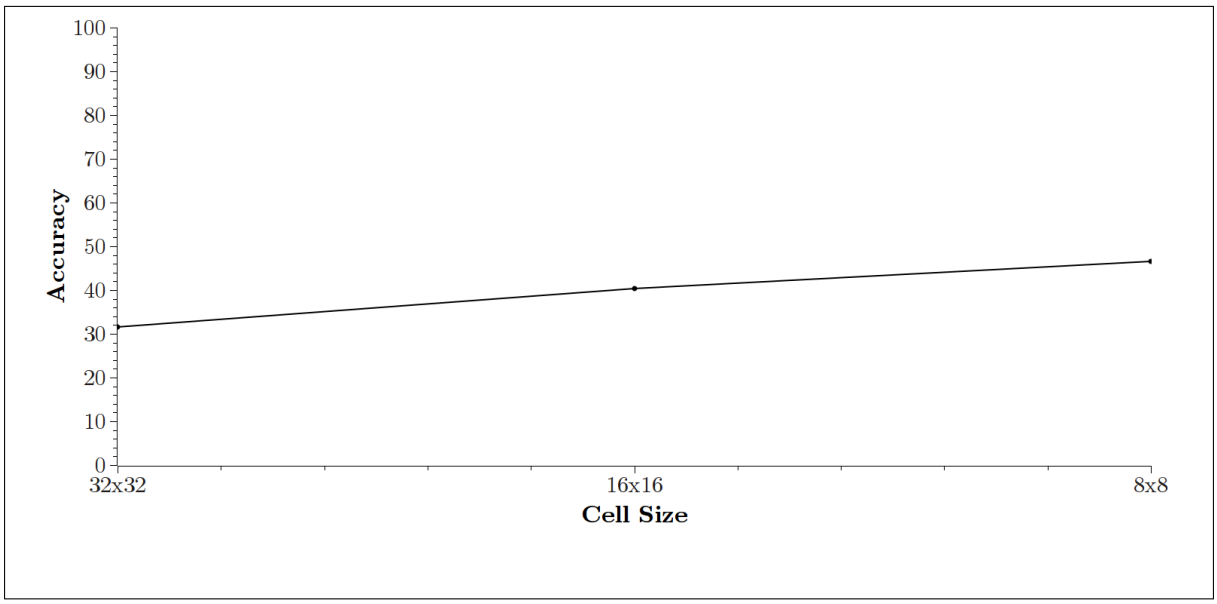


Figure 7.11: Recognition accuracy for ANN-based word recognition using GF-based features

Gabor Filter-based Features Figure 7.11 shows the performance of the ANN-based recogniser when trained with GF-based features. As was the case with UBs, there appears to be a relationship between the size of the cells that word images are partitioned into and the recognition accuracy. In this experiment, the maximum recognition accuracy of 46.64% occurred when the images were partitioned into 8x8 cells.

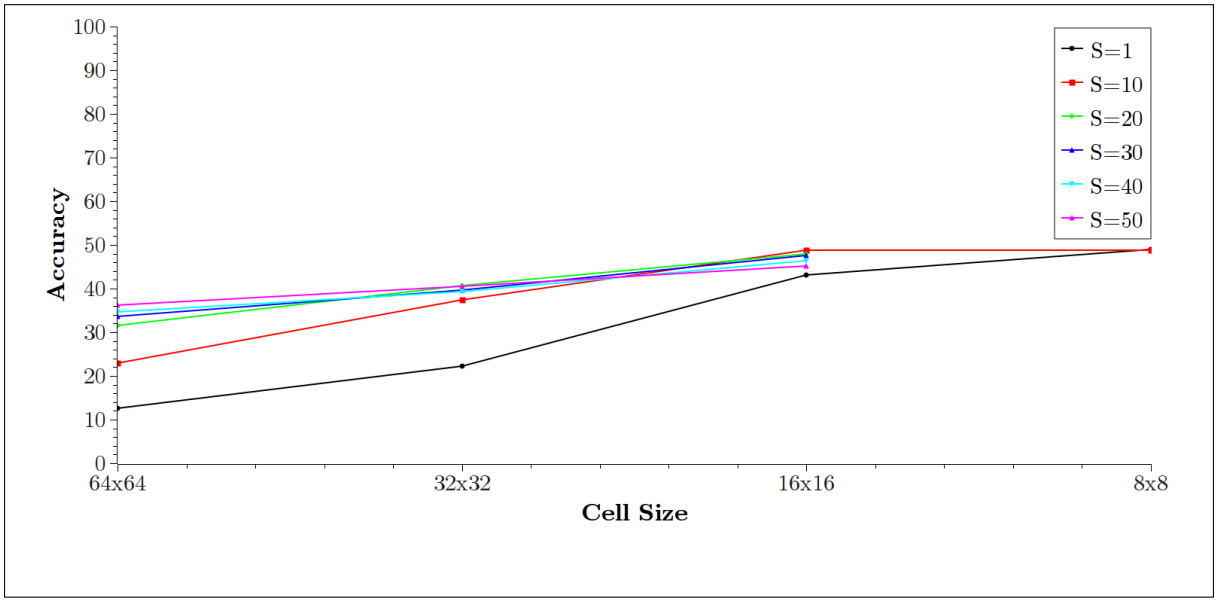


Figure 7.12: Recognition accuracy for ANN-based word recognition using DCT coefficients as features

Discrete Cosine Transform Coefficients Figure 7.12 shows the recognition accuracy when DCT coefficients are used as features for ANN-based word recognition. When extracting DCT features, the size of the cells that the DCT is performed on and the number of DCT coefficients to be used as features need to be specified. In this experiment, ANNs were not constructed when the DCT needed to be performed on 8x8 cells from which more than 10 coefficients needed to be extracted as the ANN architectures became too large. Once again, the size of the cells that the images are partitioned into has a visible effect on recognition accuracy, with smaller cells resulting in increased recognition accuracies. However, as the number of coefficients used increases, the recognition accuracy for the varying cell sizes appears to begin to converge. The highest recognition accuracy of 48.96% was achieved when the images were partitioned into 8x8 cells and a single DCT coefficient was extracted from each cell.

Discussion This experiment has investigated the use of 6 different features for ANN-based Bushman word recognition and it was found that the choice of descriptive features has an effect on the recognition accuracy. The highest recognition accuracy of 53.76% was achieved when HoGs were used and the poorest performing features were GMs, which only achieved a recognition accuracy of 23.79%. The other descriptive features all had recognition accuracies in the 40-50% range. The recognition accuracies for all of the descriptive features is summarised in Table 7.2.

In this experiment, various values for the feature extraction parameters were also investigated. As was the case with SVM-based word recognition, it was found that the values

Table 7.2: Recognition accuracy for ANN-based word recognition for all descriptive features with best performing parameter values

Feature Set	Recognition Accuracy
Histograms of Oriented Gradients	53.76%
Discrete Cosine Transform Coefficients	48.96%
Undersampled Bitmaps	46.71%
Gabor Filter-based Features	46.63%
Marti & Bunke Features	40.30%
Geometric Moments	23.79%

of these feature extraction parameters had an effect on recognition accuracy. The same trend existed regarding the size of the cells that the word images were partitioned into, where smaller cells resulted in higher recognition accuracy. Similarly, for the best performing HoGs, when HoGs were extracted from a single resolution equal to the size of the word image, the HoGs performed poorly; however, when a smaller resolution was used the highest recognition accuracy was achieved. Thus, it can be concluded that the values for the feature extraction parameters are important and an attempt should be made to optimise them when building an ANN-based recogniser.

7.1.2.2 Experiment: Artificial Neural Network with Hybrid Features

Purpose To investigate the accuracy that can be achieved when an ANN-based recogniser is used with hybrid features for Bushman word recognition.

Procedure

- HoGs, DCT coefficients and UBs, which were found to produce the best results in the previous experiment, are combined to create hybrid features made up of n individual features at a time where $n \in \{2, 3\}$ resulting in a total of 4 different hybrid features.
- The features are extracted using the best performing parameters in the previous experiment.

Results Figure 7.13 shows the results of the hybrid features that were created by combining the three best performing features from Experiment 7.1.2.1 in various ways.

Discussion Figure 7.13 shows that most of the hybrid features perform equally well, except when HoGs are supplemented by UBs, which performs slightly better and reaches a maximum recognition accuracy of 53.6%. The highest recognition accuracy for the individual features was 53.76% and occurred when HoGs were used, thus demonstrating that the hybrid features offered no advantage over the individual features. However, when the DCT coefficients and UBs were combined with HoGs, there was a slight improvement in recognition accuracy on that achieved by the DCT coefficients and UBs individually. These findings suggest that the hybrid features considered offer no visible improvement on the individual features in terms of the maximum recognition accuracy that can be achieved by the ANN-based recogniser.

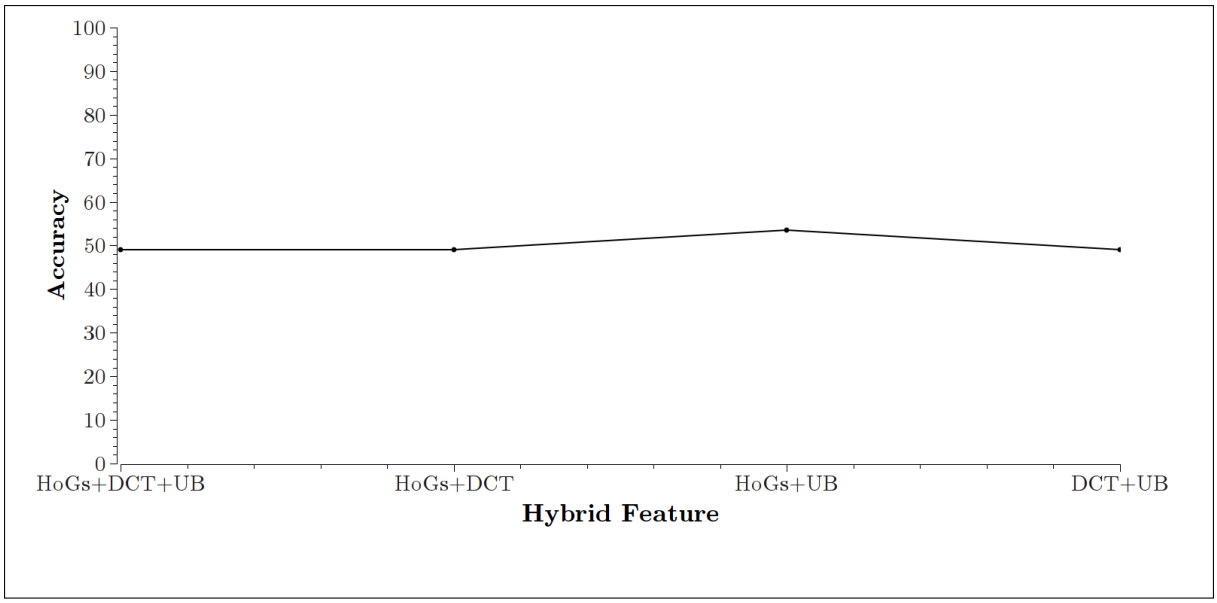


Figure 7.13: Recognition accuracy for ANN-based word recognition using hybrid features

7.1.2.3 Experiment: Artificial Neural Network with Synthetic Training Data

Purpose To investigate the accuracy that can be achieved when an ANN-based recogniser is trained with supplemental synthetic data for Bushman word recognition.

Procedure

- The transformations described in Section 6.1.5 are applied to each training sample to create 4 synthetic versions of each training sample.
- HoGs are used as features since they had the highest recognition accuracy in previous experiments.

Results By including the transformations described in Section 6.1.5, an average word recognition accuracy of 57.61% was achieved.

Discussion The 57.61% recognition accuracy achieved with synthetic data reflects an increase on the recognition accuracy with the best performing HoGs features by 3.85%. The synthetic data used in this study was relatively simple to create and was based on transformations of the training data and this experiment has shown how it can be beneficial for ANN-based word recognition.

7.1.2.4 Experiment: Artificial Neural Network and Number of Samples

Purpose To investigate the effect that the number of training samples has on the performance of an ANN-based word recogniser.

Procedure

- The sub-corpora described in Section 6.3.1 are used.
- HoGs are used as features since they had the highest recognition accuracy in previous experiments.

Results The results of this experiment are shows in Figure 7.14.

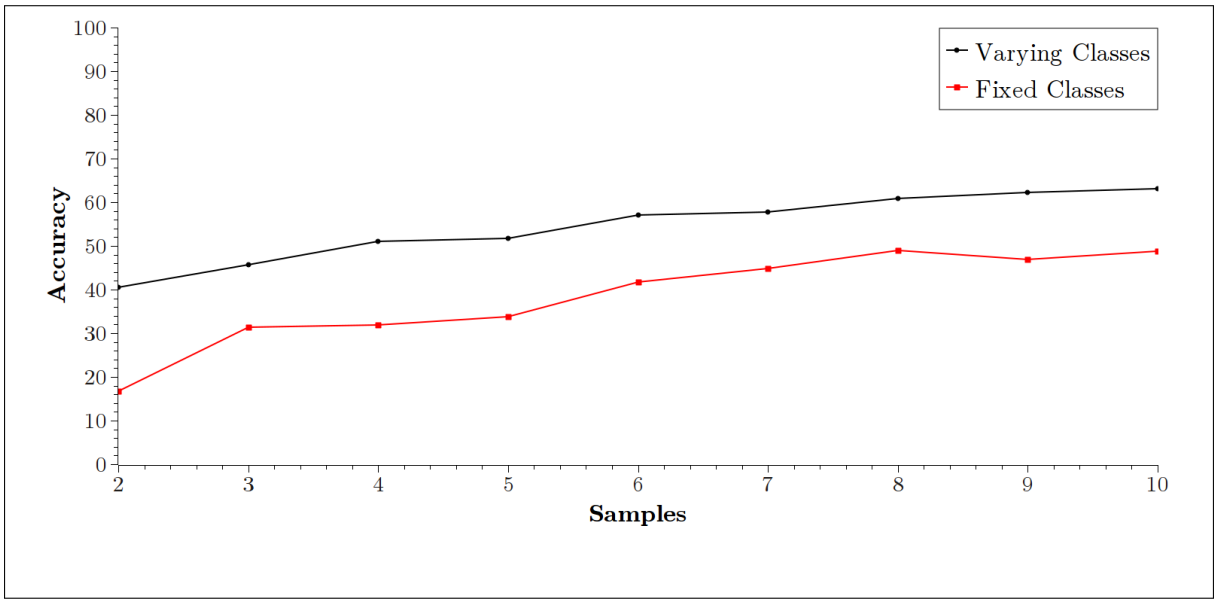


Figure 7.14: Recognition accuracy for ANN-based word recognition for different numbers of training samples

Discussion Figure 7.14 conveys two pieces of information regarding the number of training samples available. As was the case with SVM-based recognition, the *varying classes* line represents the case where the number of classes that need to be recognised is dictated by the minimum number of training samples available for that class and the *fixed classes* line represents the case where the number of classes that need to be recognised is fixed at 39 and the number of samples per class is also fixed (as opposed to a minimum number of samples per class). As can be seen from Figure 7.14, in both cases a positive linear relationship exists between the number of training samples per class and recognition accuracy.

7.1.2.5 Experiment: Artificial Neural Network for Multiple Authors

Purpose To investigate the effect that multiple authors have on the performance of an ANN-based word recogniser.

Procedure

- The corpus described in Section 6.3.1 containing the handwriting of two authors is used for training and recognition.
- HoGs are used as features since they had the highest recognition accuracy in previous experiments.

Results The recognition accuracy when using an ANN for recognising Bushman words written by multiple authors was 52.98%.

Discussion Using an ANN for recognising handwritten words for multiple authors resulted in a recognition accuracy of 52.98%, down from 53.76% for a single author. This decrease in accuracy by 0.78% is relatively small for the benefit of having a recogniser capable of recognising handwritten words from multiple authors.

7.1.2.6 Analysis: Neural Network Architecture

Section 6.3.4 discussed the ANN experimental design and included a description of a generic ANN architecture that contained an input layer, a single hidden layer and an output layer. The number of nodes in the input and output layers were determined by the size of the input vector and the number of classes that need to be recognised respectively. However, there is no easy way to determine the optimal number of neurons in the hidden layer. For the ANN-based experiment in this study, a number of ANN architectures were used where the number of neurons, N, in the hidden layer was based on the size of the input and output layers where $N \in \{S_I, \frac{S_I}{2}, S_O, \frac{S_O}{2}, \frac{S_I + S_O}{2}\}$, where S_I is the size of the input layer and S_O is the size of the output layer.

The results reported in the experiments in this section are those of the best performing ANN architectures and Figure 7.15 shows the distribution of the best performing architectures for all experimental treatments.

As can be seen from the figure, the best performing architectures, on average, generally appear to have the number of neurons in the hidden layer determined by a function of the number of inputs. However, it is hard to recommend one architecture over another as, in some cases, the other architectures result in the best performance. Thus, these findings support the argument that it is better to find an optimal architecture through experimentation.

7.1.3 Hidden Markov Models for Word Recognition

In the two previous sections the results of experiments in which SVMs and ANNs were used for Bushman word recognition were discussed. In this section, experiments investigating the use of HMMs for Bushman word recognition for the following are presented:

- The use of different descriptive features in Section 7.1.3.1.

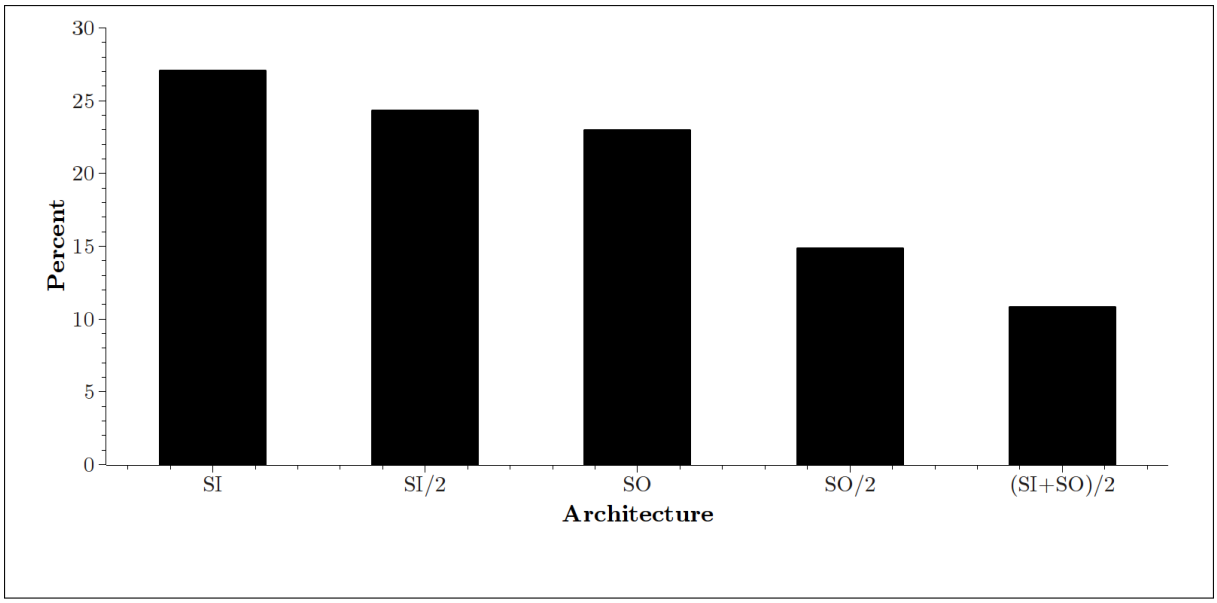


Figure 7.15: Performance of different ANN architectures

- The use of hybrid features in Section 7.1.3.2.
- The effect of synthetic training data in Section 7.1.3.3.
- The effect of the number of training samples in Section 7.1.3.4.
- The effect of multiple authors in Section 7.1.3.5.

The HMM experimental procedure described in Section 6.3.5.1 is used for all HMM-based word recognition experiments. For all experiments, except the one investigating the effect of multiple authors, the default corpus containing the handwriting of a single author, as described in Section 6.3.1, is used. The recognition accuracy reported is based on Equation 6.1 and is a measure of the proportion of words that were correctly classified.

7.1.3.1 Experiment: Hidden Markov Model with Different Features

Purpose To investigate the performance of a HMM-based word recogniser with various descriptive features.

Procedure

1. Features are extracted for each of the features described in Chapter 5 using varying parameters.
2. The best performing parameter values for each features are then used to provide a comparison of the different features for HMM-based word recognition.

Results

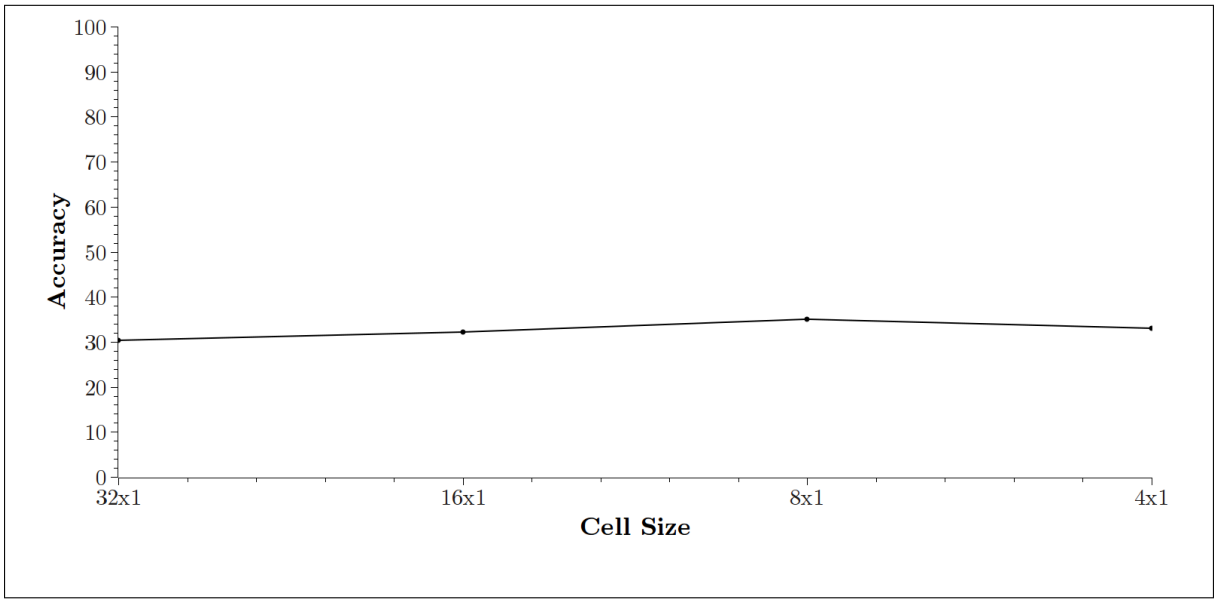


Figure 7.16: Recognition accuracy for HMM-based word recognition using UBs as features

Undersampled Bitmaps Figure 7.16 shows the results of recognising Bushman words using HMMs trained with UBs. As can be seen from the figure, the size of the cells that the image columns are partitioned into appears to have little effect on the recognition accuracy achieved. The best recognition accuracy of 35.02% was achieved when the image columns were partitioned into 8x1 cells, though this is only very slightly better than the 35.01% achieved when the image columns were partitioned into 4x1 cells.

Marti & Bunke Features The average word recognition accuracy when using the M&B features for HMM-based Bushman word recognition was 44.47%.

Geometric Moments Figure 7.17 shows the word recognition accuracy when GMs are used as features for the HMM-based recogniser. As can be seen from the figure, the recogniser appears to perform equally well when the image columns are partitioned into 32x1, 16x1 and 8x1 cells, regardless of the GMs used as features. When the image columns are not partitioned into cells, the recogniser performs relatively poorly. The highest recognition accuracy of 27.16% occurs when the image columns are partitioned into 8x1 cells and only the first GM (M1) is used.

Histograms of Oriented Gradients HoGs performed extremely poorly for HMM-based word recognition with a highest recognition accuracy of 1.28% being achieved. In this experiment, the classification for each word image was the same regardless of how the HoGs were extracted. Alternate sliding window column widths and overlaps were investigated in order to determine if they were the cause of the poor performance and it was found that, for this features, they had a negligible effect on the performance of the recogniser.

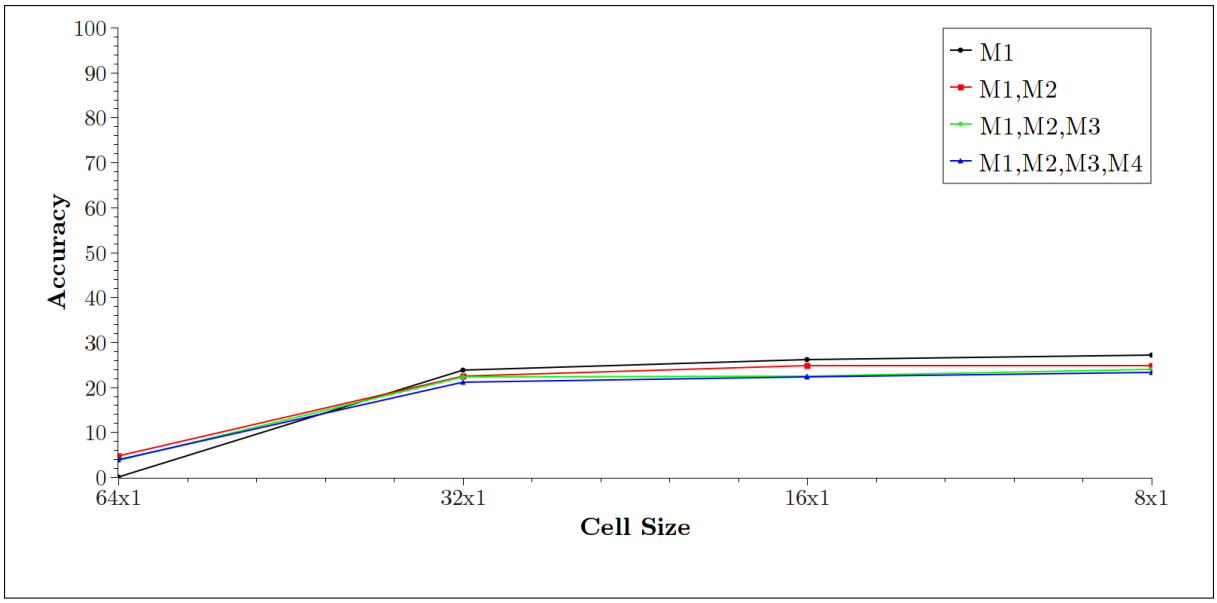


Figure 7.17: Recognition accuracy for HMM-based word recognition using GMs as features

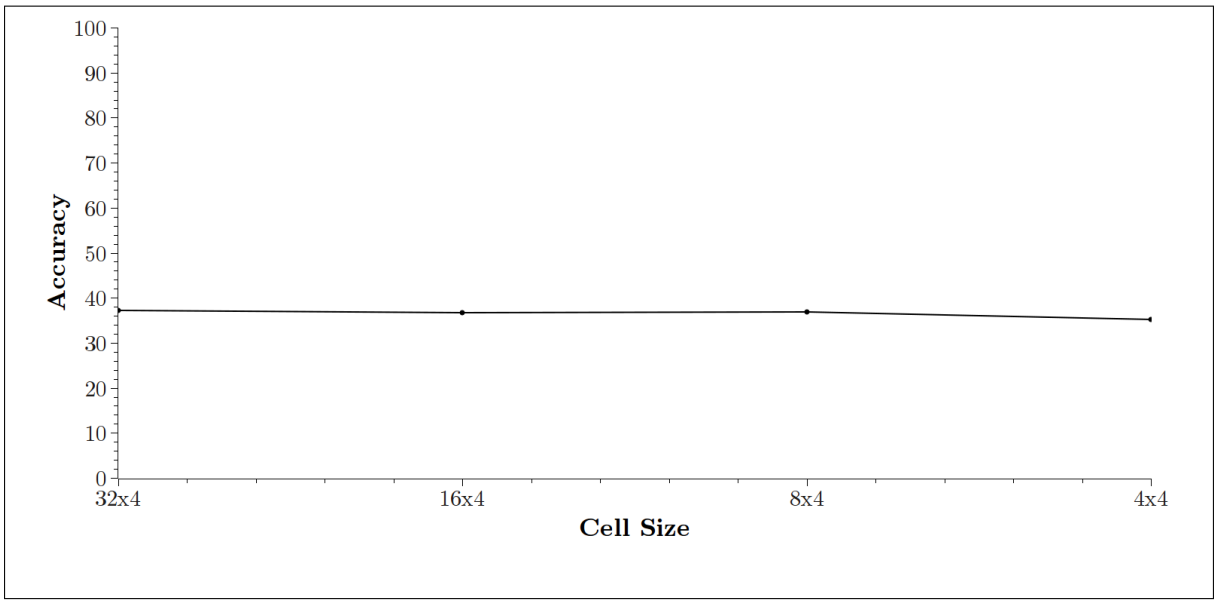


Figure 7.18: Recognition accuracy for HMM-based word recognition using GF-based features

Gabor Filter-based Features Figure 7.18 shows the performance of the HMM-based recogniser when trained with GF-based features. As can be seen from the figure, the size of the cells that the image columns are partitioned into appears to have little effect on recognition accuracy, with the highest recognition accuracy of 37.26% being achieved when the image columns were partitioned into 32x4 cells.

Discrete Cosine Transform Coefficients Figure 7.19 shows the recognition accuracy when the HMM-based recogniser is trained with DCT coefficients. In this experiment, HMMs were not trained for all of the values of the feature extraction parameters mentioned in Section 5.7 as the models became too large. As can be seen from the figure, whenever the image columns are partitioned into cells, the highest recognition accuracy is achieved

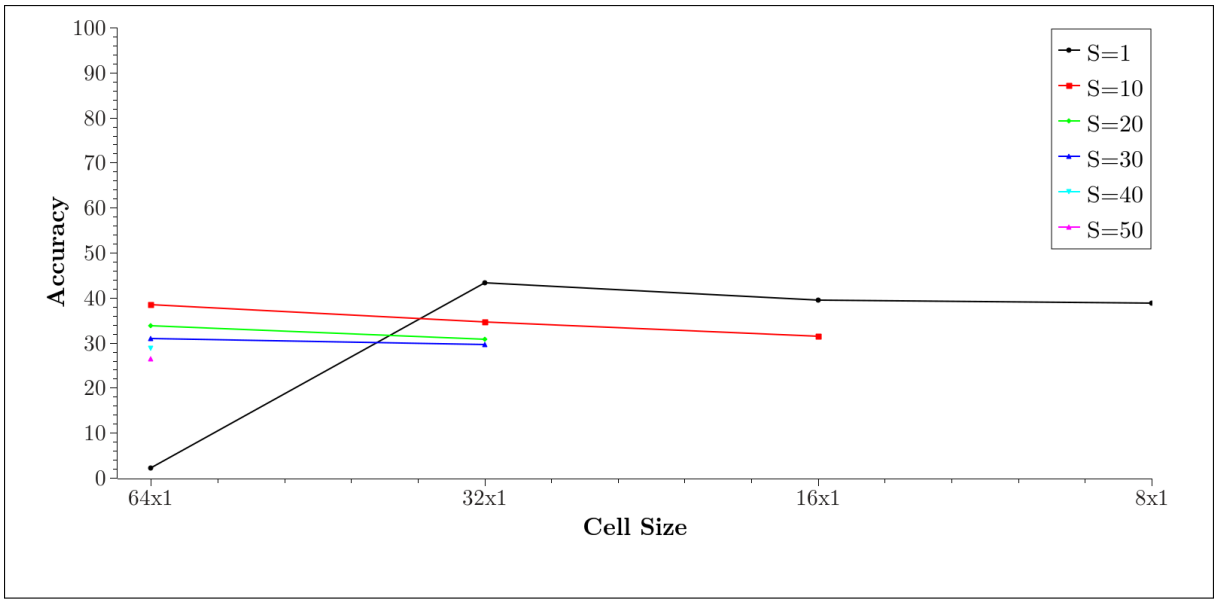


Figure 7.19: Recognition accuracy for HMM-based word recognition using DCT coefficients as features

when only a single DCT coefficient per cell is used as a feature. In fact, it appears that increasing the number of DCT coefficients results in decreasing recognition accuracy. This occurs in all cases, except when the image columns are not partitioned into cells. The highest recognition accuracy of 43.35% occurs when the image columns are partitioned into 32x1 cells and a single DCT coefficient is extracted from each cell.

Discussion There are a wide variety of features that can be used when creating a HMM-based word recogniser and it can be expected that the choice of features will have a significant effect on the performance of the recogniser. This experiment investigated the use of 6 different descriptive features for HMM-based Bushman word recognition. Features for the HMM-based word recogniser were extracted using a left-to-right sliding window modelling the time domain (see Section 5.1.2 on HMM feature extraction). It was shown that the different features result in varying recogniser performance with the highest recognition accuracy of 44.47% being achieved when the M&B features were used. The performance of the other features varied, with HoGs being the worst performing by only achieving a recognition accuracy of 1.28%. Table 7.3 summarises the recognition accuracy for the different descriptive features.

Table 7.3: Recognition accuracy for HMM-based word recognition for all descriptive features with best performing parameter values

Feature Set	Recognition Accuracy
Marti & Bunke Features	44.47%
Discrete Cosine Transform Coefficients	43.35%
Gabor Filter-based Features	37.26%
Undersampled Bitmaps	35.02%
Geometric Moments	27.16%
Histograms of Oriented Gradients	1.28%

All of the features, except the M&B features, required that parameters for feature extraction be specified. The results of this experiment have shown that, in some cases, the values of these feature extraction parameters had a visible effect on recognition accuracy. For instance, for the DCT features, the number of DCT coefficients used as features affected the recognition accuracy where increasing the number of DCT coefficients per cell decreased recognition accuracy. However, for features where one of the feature extraction parameters was the size of the cells that the image columns were partitioned into, the size of the cells had little effect on recognition accuracy. This is in contrast to the findings for SVM and ANN-based word recognition in Sections 7.1.1.1 and 7.1.2.1 where the size of the cells that an image was partitioned into had a visible effect on recognition accuracy. A possible reason for this is that, since the features are extracted using a sliding window, the smaller Cartesian space reduces the effect that the cell size has on performance compared to the larger Cartesian space for SVM and ANN-based recognition.

The findings of this experiment suggest that the choice of descriptive features is an important factor in designing a HMM-based word recogniser for Bushman texts.

7.1.3.2 Experiment: Hidden Markov Model with Hybrid Features

Purpose To investigate the accuracy that can be achieved when a HMM-based recogniser is used with hybrid features for Bushman word recognition.

Procedure

- M&B features, GF-based features and DCT coefficients, which were found to produce the best results in the previous experiment, are combined to create hybrid features made up of n individual features at a time where $n \in \{2, 3\}$ resulting in a total of 4 different hybrid features.
- The features are extracted using the best performing parameters in the previous experiment.

Results Figure 7.20 shows the results of the hybrid features that were created by combining the three best performing features in various ways.

Discussion Hybrid features could extend the descriptive capabilities of the individual features and thus could potentially improve the performance of the recogniser. This is indeed the case when the M&B features are combined with DCT features and the recognition accuracy increased slightly from 44.47% to 44.79%. However, this increase in recognition accuracy is extremely small and it could be argued that it has not actually led to any significant improvement. Furthermore, combining the DCT features with GF-based features reduced the performance of the HMM-based recogniser to 40.31%. In light of these findings, it can be concluded that the hybrid features used in this study for HMM-based Bushman word recognition do not result in any real improvement and, in fact, in some cases reduce the performance of the recogniser.

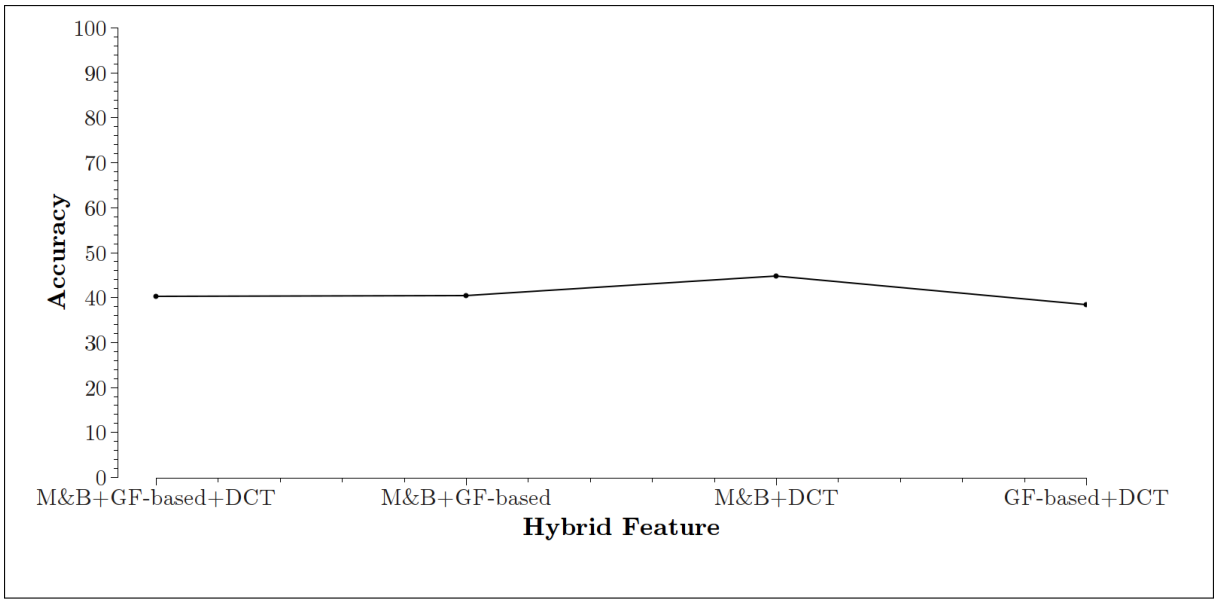


Figure 7.20: Recognition accuracy for HMM-based word recognition using hybrid features

7.1.3.3 Experiment: Hidden Markov Model with Synthetic Training Data

Purpose To investigate the accuracy that can be achieved when a HMM-based recogniser is trained with supplemental synthetic data for Bushman word recognition.

Procedure

- The transformations described in Section 6.1.5 are applied to each training sample to create 4 synthetic versions of each training sample.
- The hybrid feature made up of M&B features and DCT coefficients is used since it had the highest recognition accuracy in previous experiments.

Results By including the transformations described in Section 6.1.5, an average word recognition rate of 51.61% was achieved.

Discussion The 51.61% recognition accuracy achieved with synthetic data reflects an increase in the recognition accuracy by 6.82% compared to the best performing hybrid features. This experiment has shown how synthetic training data can be beneficial, especially since it is relatively easy to create.

7.1.3.4 Experiment: Hidden Markov Model and Number of Samples

Purpose To investigate the effect that the number of training samples has on the performance of a HMM-based word recogniser.

Procedure

- The sub-corpora described in Section 6.3.1 are used.
- The hybrid feature made up of M&B features and DCT coefficients is used since it had the highest recognition accuracy in previous experiments.

Results The results of this experiment are shows in Figure 7.1.3.4.

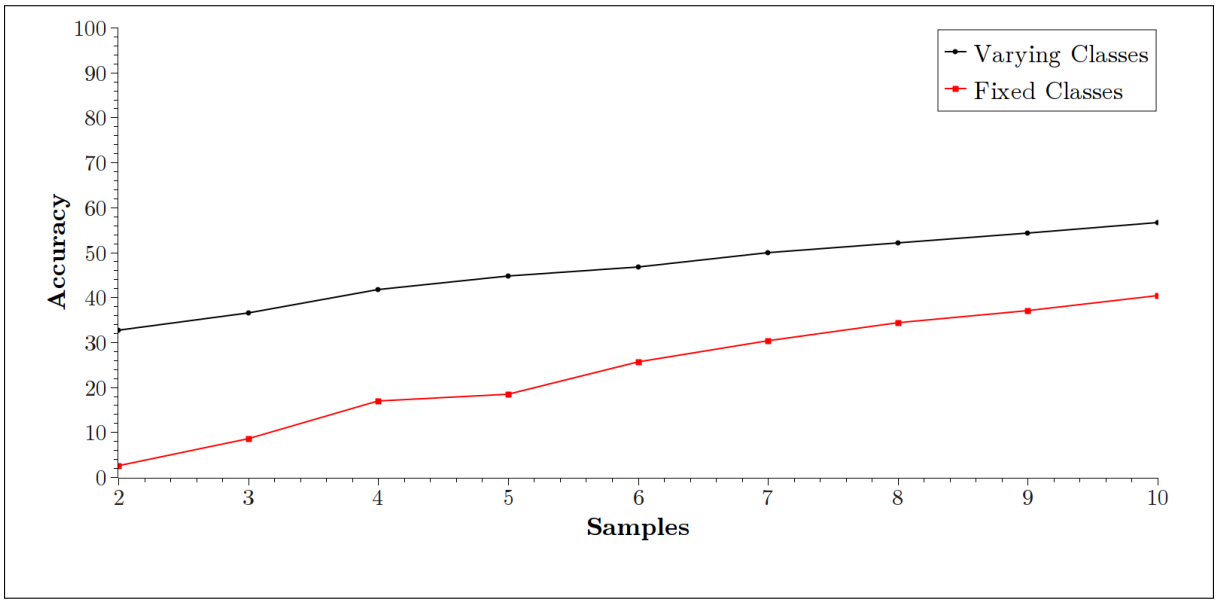


Figure 7.21: Recognition accuracy for HMM-based word recognition for different numbers of training samples

Discussion Figure 7.21 conveys the same information regarding varying classes and fixed classes as was the case for the SVM and ANN-based experiments investigating the effect of the number of training samples. Once again, the same positive linear relationship exists between the number of training samples per class and recognition accuracy.

7.1.3.5 Experiment: Hidden Markov Model for Multiple Authors

Purpose To investigate the effect that multiple authors have on the performance of a HMM-based word recogniser.

Procedure

- The corpus described in Section 6.3.1 containing the handwriting of two authors is used for training and recognition.
- The hybrid feature made up of M&B features and DCT coefficients is used since it had the highest recognition accuracy in previous experiments.

Results The recognition accuracy when using a HMM to recognise Bushman words written by multiple authors was 44.13%.

Discussion The 44.13% recognition accuracy when recognising the handwriting of multiple authors is a decrease by 0.66% on the recognition accuracy achieved when only words written by a single author were recognised. This slight decrease is arguably insignificant given the added benefit of recognising the handwriting of multiple authors.

7.2 Text Line Recognition

The previous section presented a set of experiments for Bushman word recognition using SVMs, ANNs and HMMs. In this section, experiments for Bushman text line recognition using a HMM are presented.

7.2.1 Hidden Markov Models for Line Recognition

In this section, experiments investigating the use of HMMs for Bushman text line recognition for the following are presented. This section begins with an experimented conducted in order to find suitable parameters for the HMMs used in this study. Thereafter, the following factors are investigated:

- The use of different descriptive features in Section 7.2.1.2.
- The use of hybrid features in Section 7.2.1.3.
- The effect of synthetic training data in Section 7.2.1.4.
- The effect of the number of training samples in Section 7.2.1.5.
- The effect of multiple authors in Section 7.2.1.6.
- The effect of a bigram word language model in Section 7.2.1.7.

The HMM experimental procedure described in Section 6.4.2.1 is used for all HMM-based text line recognition experiments. For all experiments, except the one investigating the effect of multiple authors, the default corpus containing the handwriting of a single author, as described in Section 6.4.1, is used. No language model is used except for in the experiment in which the use of a language model is investigated. The recognition accuracy reported is based on Equation 6.2, which is not a percentage but rather a measure of the similarity between the target transcription and the recognition output.

7.2.1.1 Experiment: Parameters for Hidden Markov Models

Purpose To determine a suitable number of states and Gaussians for all HMMs.

Procedure

- 10 DCT coefficients are extracted from each text line.
- HMMs for each pair of S states and G Gaussians are built and evaluated where $S = 4, 8, 12$ and $G = 8, 16, 24, 32$.
- M&B features are extracted from each text line.
- HMMs for each pair of S states and G Gaussians are built and evaluated where $S = 4, 8, 12$ and $G = 8, 16, 24, 32$.
- The results of the M&B-based experiments are used to validate the results of the DCT-based experiments.

Results Figure 7.22 shows the recognition accuracy for the states and Gaussians tested using DCT coefficients as features and Figure 7.23 shows the recognition accuracy using M&B features.

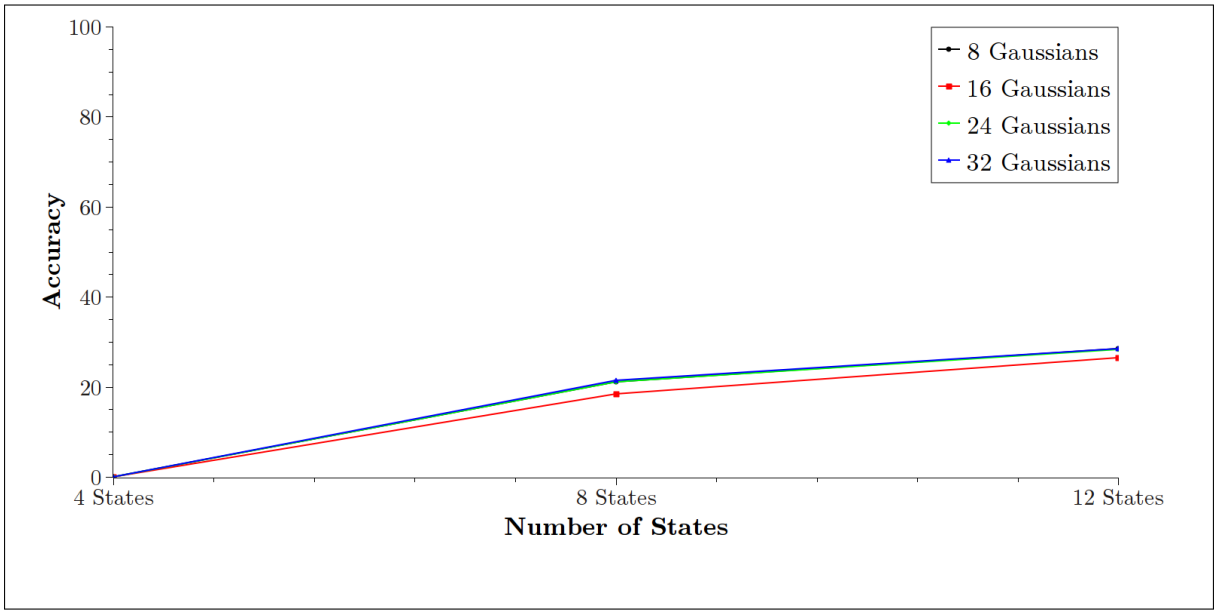


Figure 7.22: Recognition Accuracy when using DCT features to find suitable HMM model parameters

Discussion Figures 7.22 and 7.23 show the same trends even though different features have been used for classification. As can be seen in Figures 7.22 and 7.23, there is a visible difference between recognition accuracy, depending on the number of states, where having more states results in a higher recognition accuracy. Since high recognition accuracy is desirable, it is used as the main factor in deciding model parameters. A maximum of 12 states was investigated since higher numbers of states led to there not being enough data in the training samples to fit the model size.

As can be seen from Figures 7.22 and 7.23, the number of Gaussians has little effect on the recognition accuracy though it is lowest when 16 Gaussians are used for DCT coefficients

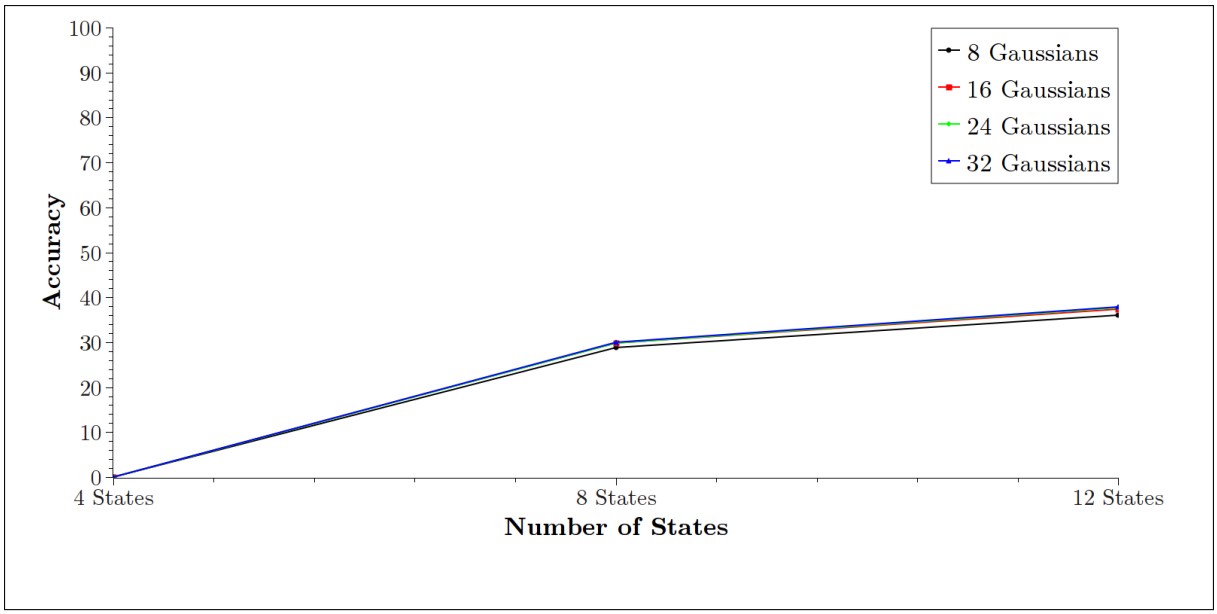


Figure 7.23: Recognition Accuracy when using M&B features to find suitable HMM model parameters

and when 8 Gaussians are used for M&B features. Given these findings, for all HMM-based experiments in this study, the HMM models are defined as having 12 states and 16 Gaussians as these both performed relatively well.

7.2.1.2 Experiment: Hidden Markov Model with Different Features

Purpose To investigate the performance of a HMM-based text line recogniser with various descriptive features.

Procedure

1. Features are extracted for each of the features described in Chapter 5 using varying parameters.
2. The best performing parameter values for each feature are then used to provide a comparison of the different features for HMM-based text line recognition.

Results

Undersampled Bitmaps Figure 7.24 shows the results of recognising Bushman lines using an HMM trained with UBs. As can be seen from the figure, the size of the cells that the image columns are partitioned into has little effect on the performance of the recogniser, except when the image column is partitioned into 32x1 cells and the recognition accuracy is 0. The best recognition recognition accuracy of 22.39 occurs when the image columns are partitioned into 4x1 cells.

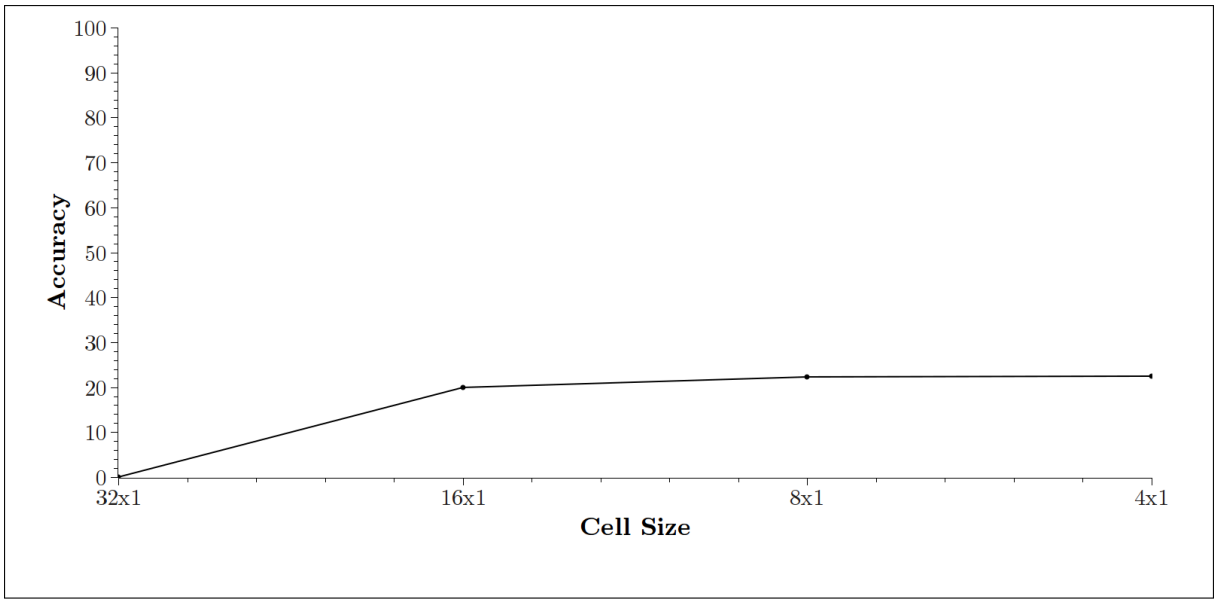


Figure 7.24: Recognition accuracy for HMM-based text line recognition using UBs as features

Marti & Bunke Features The recognition accuracy when using the M&B features for HMM-based Bushman text line recognition was 35.7.

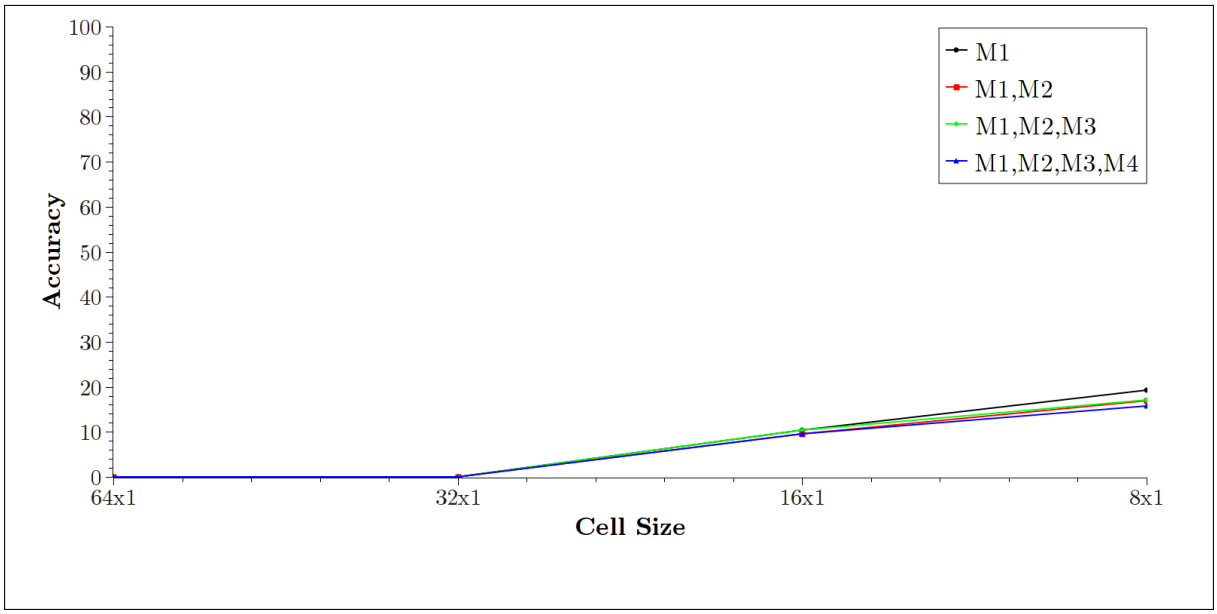


Figure 7.25: Recognition accuracy for HMM-based text line recognition using GMs as features

Geometric Moments Figure 7.25 shows the recognition accuracy when GMs are used as features for the HMM-based recogniser. As can be seen from the figure, the size of the cells that the image columns are partitioned into has an effect on the recognition accuracy, though the GMs that are used as features do not seem to affect the performance of the recogniser. The highest recognition accuracy of 19.31 occurs when the image columns are partitioned into 8x1 cells and only the first GM (M1) is extracted from each cell.

Histograms of Oriented Gradients HoGs performed extremely poorly for HMM-based text line recognition, with a negative recognition accuracy for the cases tested. This is possible since, as previously mentioned, the recognition accuracy for text lines is not actually a percentage but is a function of the number of insertion, deletion and substitution errors. As was the case for Bushman word recognition using a HMM trained with HoGs, the recogniser produced the same output for each testing sample, regardless of the data that the sample contained.

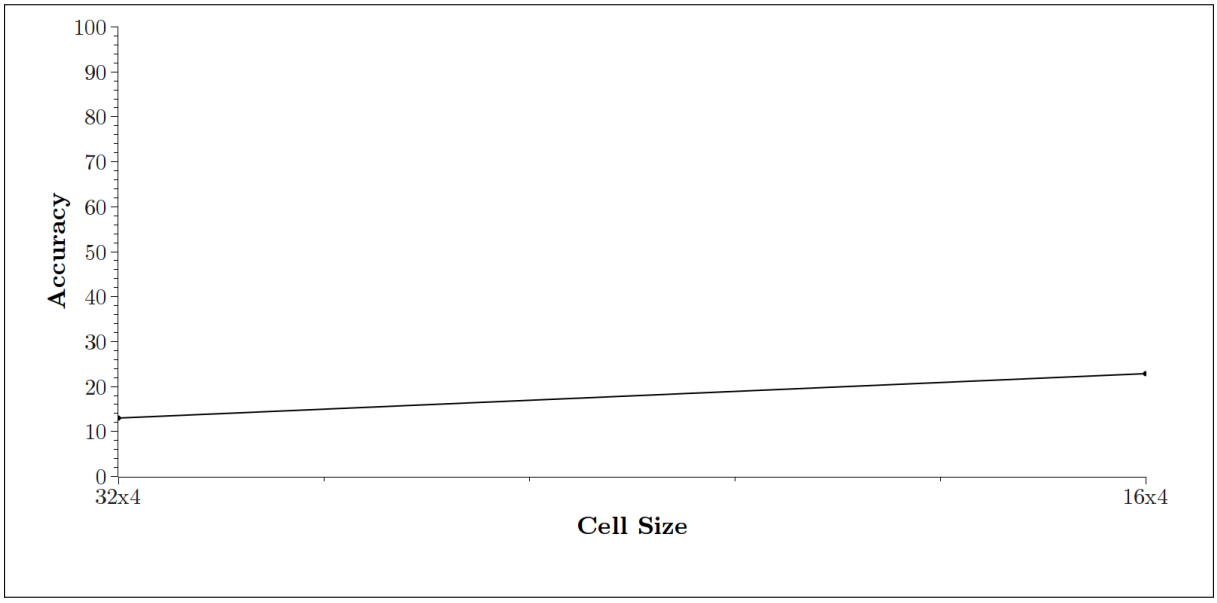


Figure 7.26: Recognition accuracy for HMM-based text line recognition using GF-based features

Gabor Filter-based Features Figure 7.26 shows the performance of the HMM-based recogniser when trained with GF-based features. As can be seen from the figure, the size of the cells that the image columns are partitioned into has an effect on the recognition accuracy, with the smaller partition resulting in a higher recognition accuracy. The highest recognition accuracy of 22.76 occurred when the image columns were partitioned into 16x4 cells.

Discrete Cosine Transform Coefficients Figure 7.27 shows the recognition accuracy when the HMM-based recogniser was trained with DCT coefficients. When extracting DCT coefficients, the size of the cells that the image columns are partitioned into and the number of DCT coefficients that are extracted as features need to be specified. As was the case with HMM-based Bushman word recognition, HMMs were not trained for all methods of feature extraction as the models became too large. These show up as missing points on the graph. As can be seen from the figure, the recognition accuracy differs little for the different feature extraction parameters. However, when the image columns are not partitioned into cells, there appears to be a slight but steady decline as more DCT coefficients are used as features. The highest recognition accuracy of 27.67 occurred when a cell size of 64x1 was used with 10 DCT coefficients.

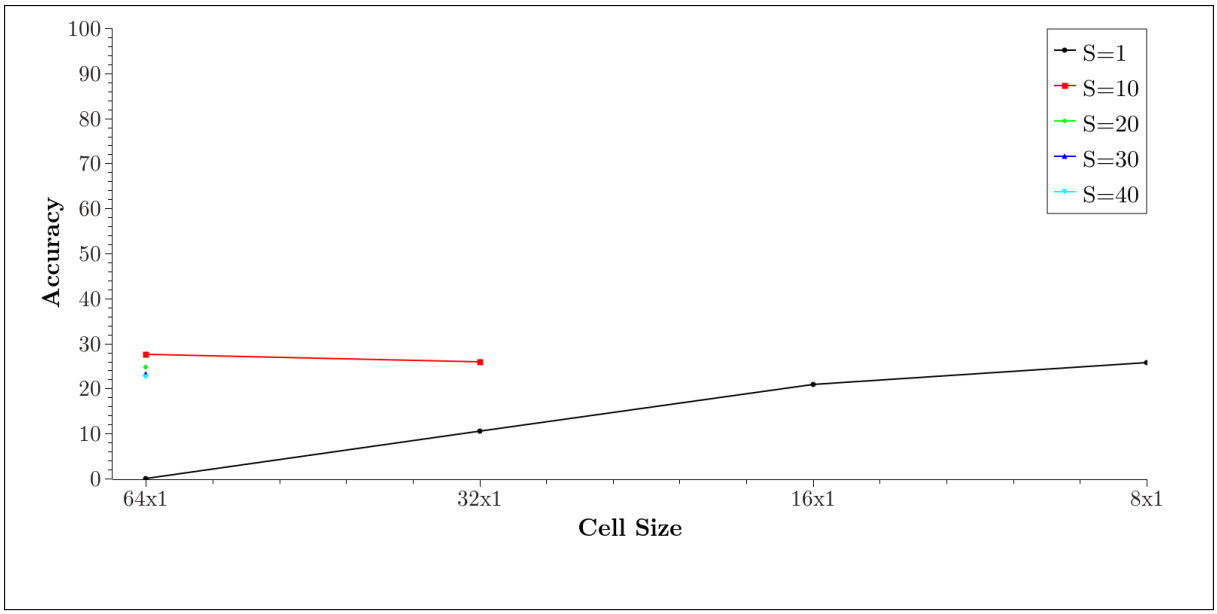


Figure 7.27: Recognition accuracy for HMM-based text line recognition using DCT coefficients as features

Discussion When building a HMM-based text line recogniser, the descriptive features that are used for training and recognition could potentially have an effect on the performance of the recogniser. This experiment investigated the use of 6 different features for Bushman text line recognition and it was found that the choice of descriptive features affected the recognition accuracy. For instance, when the best performing M&B features were used, a recognition accuracy of 35.7 was achieved. Conversely, when the worst performing HoGs were used, the recognition accuracy was negative. The other descriptive features had accuracies in the range of 19-28. Thus, given this large variation in performance, it can be concluded that the choice of descriptive features is an important decision when building a HMM-based text line recogniser. Table 7.4 summarises the performance of the recogniser for the various features.

Table 7.4: Recognition accuracy for HMM-based text line recognition for all descriptive features with best performing parameter values

Feature Set	Recognition Accuracy
Marti & Bunke Features	35.7
Discrete Cosine Transform Coefficients	27.67
Gabor Filter-based Features	22.76
Undersampled Bitmaps	22.39
Geometric Moments	19.32
Histograms of Oriented Gradients	Negative

All features, except the M&B feature, required that parameters for feature extraction be specified and it was found that, in some cases, the values for the feature extraction parameters had an effect on the performance of the recogniser. For instance, when GMs were used, it was found that the choice of GMs (M1, M2, M3, M4) had little effect on the performance of the recogniser. Conversely, it was found that, for features where one of the feature extraction parameters was the size of the cells that the image columns were partitioned into, the size of the cells had an effect on the recognition accuracy.

7.2.1.3 Experiment: Hidden Markov Model with Hybrid Features

Purpose To investigate the accuracy that can be achieved when a HMM-based recogniser is used with hybrid features for Bushman text line recognition.

Procedure

- M&B features, DCT coefficients and UBs, which were found to produce the best results in the previous experiment, are combined to create hybrid features made up of n individual features at a time where $n \in \{2, 3\}$ resulting in a total of 4 different hybrid features.
- The features are extracted using the best performing parameters in the previous experiment.

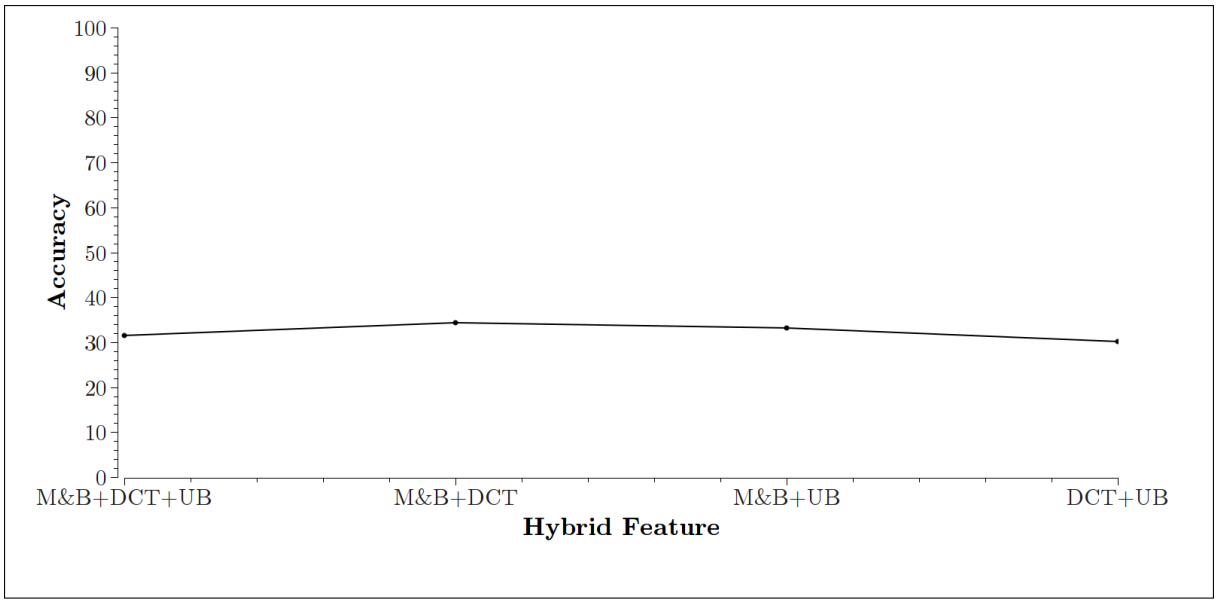


Figure 7.28: Recognition accuracy for HMM-based text line recognition using hybrid features

Results Figure 7.28 shows the recognition accuracy of the hybrid features that were created by combining the three best performing features in various ways.

Discussion As can be seen from the figure, most of the hybrid features perform similarly with the M&B and DCT combination performing slightly better than the others. When this hybrid combination is used, the recognition accuracy was 34.36. This is, in fact, slightly below the performance of the recogniser when only the M&B features were used. However, the M&B features improved on the performance when the DCT coefficients and UBs were used independently. Thus, these findings have shown that the best performing feature improves the performance of weaker features, but that the weaker features do not improve the performance of the strongest feature.

7.2.1.4 Experiment: Hidden Markov Model with Synthetic Training Data

Purpose To investigate the accuracy that can be achieved when a HMM-based recogniser is trained with supplemental synthetic data for Bushman text line recognition.

Procedure

- The transformations described in Section 6.1.5 are applied to each training sample to create 4 synthetic versions of each training sample.
- M&B features were used since they had the highest recognition accuracy in previous experiments.

Results By including the transformations described in Section 6.1.5, the text line recognition accuracy was 33.67.

Discussion The inclusion of synthetic training data led to a slight decrease in the recognition accuracy from 35.7 to 33.67. This slight decrease in the recognition accuracy is somewhat unexpected as it is well known that the performance of a handwriting recognition system often is dependent on the amount of training data available (Cano et al., 2002) and it was found in the word recognition experiments that the addition of synthetic training data led to an improvement in results. A possible explanation for this lack of improvement could be that overfitting occurs during training and the synthetic training data contributes to this. Another reason could be due to the inconsistencies in the corpus and the effect that this has on the overall performance of the recogniser. The effect that the corpus had on the performance of the recognisers is discussed in Section 7.3.5.

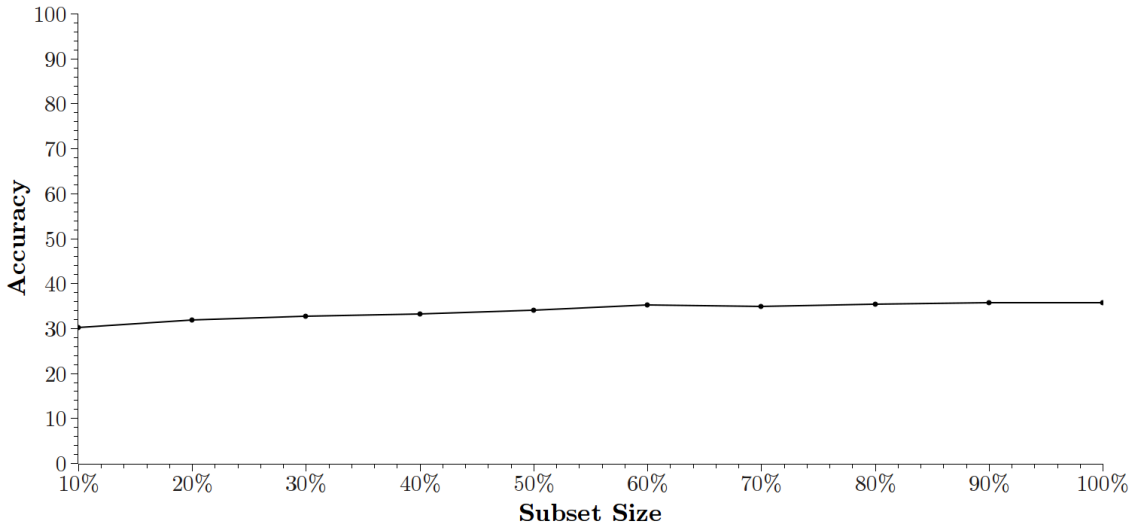
7.2.1.5 Experiment: Hidden Markov Model and Number of Samples

Purpose To investigate the effect that the number of training samples has on the performance of a HMM-based text line recogniser.

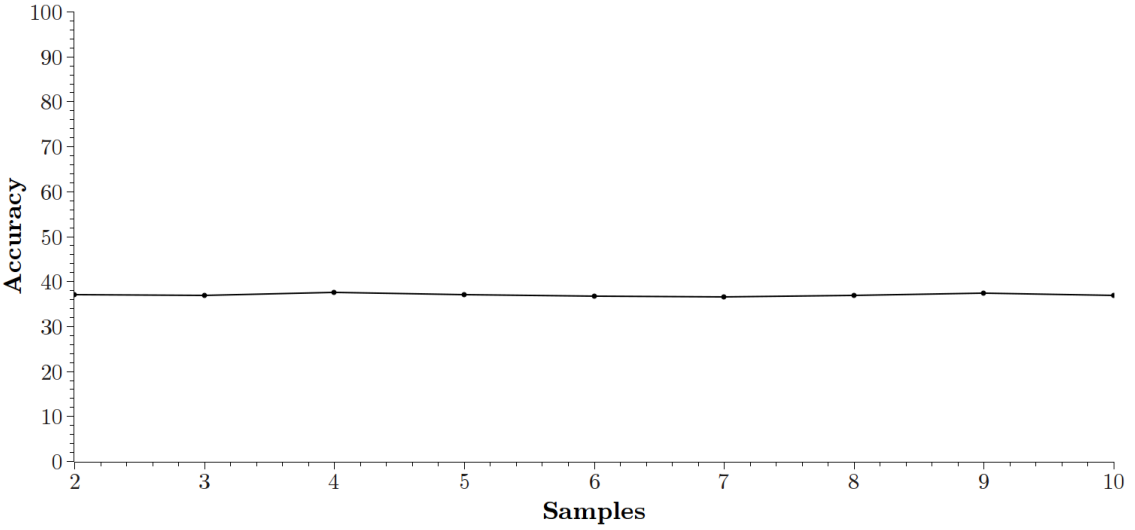
Procedure

- The sub-corpora described in Section 6.4.1 are used.
- M&B features were used since they had the highest recognition accuracy in previous experiments.

Results The results of this experiment are shows in Figure 7.29.



(a) Randomly Sampled Sub-corpora



(b) Minimum Number of Samples Per Class

Figure 7.29: Recognition accuracy for HMM-based text line recognition for different numbers of training samples

Discussion Figure 7.29 (a) shows that a very slight linear relationship exists between the size of the corpus used for training and testing and the recognition accuracy. However, the gradient of this linear relationships appears to be quite small suggesting that, for this corpus, the amount of training data does not have a large effect on the performance of the recogniser. This observation is confirmed when the minimum number of samples per class is specified and appears to have no effect on the performance of the recogniser. These observations are somewhat unexpected given the clear relationship that existed between the amount of training data and recogniser performance for word-based recognition. However, it is possible that this lack of an increase in performance could be attributed to the nature of the corpus and the effects that it had on the performance of the recogniser, which is discussed in Section 7.3.5.

7.2.1.6 Experiment: Hidden Markov Model for Multiple Authors

Purpose To investigate the effect that multiple authors have on the performance of a HMM-based text line recogniser.

Procedure

- The corpus described in Section 6.4.1 containing the handwriting of two authors was used for training and recognition.
- M&B features were used since they had the highest recognition accuracy in previous experiments.

Results When recognising the corpus containing text lines written by multiple authors the recognition accuracy was 30.04.

Discussion The recognition accuracy when recognising handwritten text lines by a single author was 35.7. Adding a second author resulted in a decrease by 4.34. This decrease in performance is, of course, not desirable. However, multiple authors increases variation within classes and therefore one would expect a decrease in performance to occur.

7.2.1.7 Experiment: Hidden Markov Model with a Statistical Language Model

Purpose To investigate the effect that a bigram language model has on the performance of a HMM-based text line recogniser.

Procedure

- A bigram word language model was used during recognition.
- M&B features were used since they had the highest recognition accuracy in previous experiments.

Results The recognition accuracy when using a bigram language model was 45.10.

Discussion No language model was used in the previous text line recognition experiments and the assumption was made that all words in the Bushman language were evenly distributed. Without a language model, the highest recognition accuracy achieved was 35.7. The incorporation of a bigram word language model led to a recognition accuracy of 45.10. This increase in recognition accuracy by 9.4 clearly demonstrates the benefit of incorporating a language model into a recogniser since it allows for additional statistical information about a language to be used to guide the recognition process.

7.3 Analysis

The previous sections presented the results of different experiments that were designed to investigate different factors that affected the performance of handwriting recognition systems. However, the results for each of the experiments were presented in isolation. In this section, the results are analysed within the greater context of all of the machine learning algorithms investigated in this study. This section begins with an analysis of the different descriptive features, followed by an analysis of the effect of hybrid features. This is then followed by an analysis of the effect of the amount of training data and multiple authors and, lastly, the results presented in this chapter are discussed within the context of the corpus used in this study.

7.3.1 Features for Handwriting Recognition

In Figure 7.30, the performance of each feature with the different machine learning algorithms is summarised. Table 7.5 shows the rankings of the different feature-machine learning algorithm pairs for word recognition, while Table 7.6 shows the ranking of the different features for text line recognition.

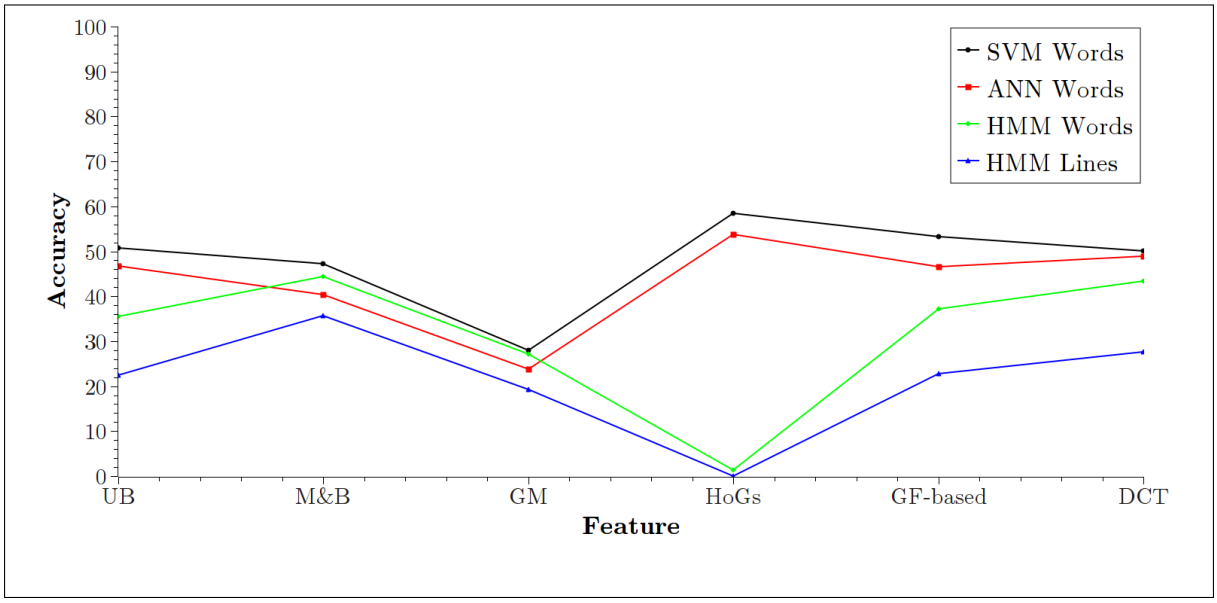


Figure 7.30: Recognition accuracy of different features for all machine learning algorithms

As can be seen from Figure 7.30, the same general performance pattern appears to exist for the different features coupled with the different machine learning algorithms with a large deviation occurring for HoGs. Tables 7.5 and 7.6 show the rankings of the different feature-machine learning algorithms combinations for word recognition and text line recognition respectively.

As can be seen from Table 7.5, the SVM-based classifiers make up 5 of the 6 best word classifiers, with the GMs-SVM classifier being the only poorly performing SVM-based classifier. The best performing word classifier overall was the HoGs-SVM classifier followed by the HoGs-ANN classifier. The fact that the two highest ranked classifiers both make use of HoGs suggests that HoGs are good features for handwriting recognition. However,

Table 7.5: Ranking of different feature-machine learning algorithm pairs for word recognition

Rank	Recogniser	Accuracy	Rank	Recogniser	Accuracy
1	HoGs-SVM	58.49%	10	M&B-HMM	44.47%
2	HoGs-ANN	53.76%	11	DCT-HMM	43.35%
3	GF-SVM	53.21%	12	M&B-ANN	40.30%
4	USB-SVM	50.73%	13	GF-HMM	37.26%
5	DCT-SVM	50.16%	14	USB-HMM	35.58%
6	DCT-ANN	48.96%	15	GM-SVM	27.96%
7	M&B-SVM	47.27%	16	GM-HMM	27.16%
8	USB-ANN	46.71%	17	GM-ANN	23.79%
9	GF-ANN	46.63%	18	HoGs-HMM	1.28%

Table 7.6: Ranking of different features for text line recognition

Rank	Recogniser	Accuracy
1	M&B	35.7
2	DCT	27.67
3	GF	22.76
4	USB	22.39
5	GM	19.31
6	HoG	0

the use of HoGs also resulted in the poorest performing HMM-based word recogniser. In the case of HMM-based recognition, HoGs were extracted using a sliding window that modeled the time domain. A possible reason for the poor HMM performance was thought to be that the relatively thin width of a sliding window column does not result in HoGs that are as descriptive as those when larger cells are used, as was the case with SVM and ANN-based recognition. However, when wider sliding window columns were used, no improvement in performance occurred. It is possible that the way in which HoGs were computed for HMM-based recognition since the regions that the sliding window column was divided into may not have provided enough support for the features to be descriptive.

The M&B features resulted in the highest recognition accuracies for HMM-based recognisers. This occurred even though the M&B features had no invariant properties as described in Section 5.3, suggesting that descriptive features with invariant properties do not necessarily result in better performance. It was discussed earlier how invariant properties may not always be desired since, when a feature is invariant to rotation, translation and scale, the descriptors for p and d will be the same. The M&B features were the best performing features for both HMM-based word and text line recognition and can be recommended as a good feature set to use for HMM-based recognition. This is somewhat expected since this feature set was designed for use with a HMM-based recogniser (Marti, 2000). However, it performed poorly as a feature for SVM and ANN-based word recognition. This large variation in performance for different machine learning algorithms suggests that the choice of features used for handwriting recognition should largely be dependent on the machine learning algorithm used.

The recognisers that made use of GMs all ranked among the worst performing recognisers occupying 3 of the last 4 positions for word recognition and the second last position

for text line recognition. This occurs even though GMs are invariant to rotation, scale and translation, as discussed in Section 5.4. This supports the notion that, invariant properties, while arguably desirable, do not necessarily guarantee the best performance.

The other recognisers, which make use of UBs, GF-based features, and DCT coefficients appear to perform similarly, making up the middle rankings with different machine learning algorithms. This average performance suggests that these features are not necessarily an optimal choice as descriptive features for Bushman handwriting recognition.

This analysis of the different descriptive features within the context of all of the machine learning algorithms has resulted in two main findings. The first of these findings is that the performance of a feature can be dependent on the machine learning algorithm being used. For instance, the HoGs and M&B features performed very differently when used with HMM-based recognisers compared to SVM and ANN-based recognisers. On the other hand UBs, GMs, GF-based features and DCT coefficients generally had the same performance for the different recognisers. However, the best performing features were in fact the ones whose performance varied the most for the different machine learning algorithms and thus, when choosing a feature for handwriting recognition, the choice should be made while considering the machine learning algorithm to be used.

The choice of machine learning algorithm depends on a number of factors, such as the type of data that is to be recognised. For instance, if it is not suitable to segment text lines into individual words then a HMM or some other machine learning algorithm that does not required word segmentation may be appropriate. Furthermore, the automatic segmentation that occurs as part of the recognition process when using a HMM may make it a better segmenter than other segmentation approaches that occur as a preprocessing step. Similarly, if units such as characters or whole words are being recognised then a SVM or ANN may be more appropriate. In this study it was found that, for word recognition, the best performing machine learning algorithm was a SVM trained using HoGs, followed closely by an ANN trained using HoGs. For text line recognition, the best performing features were the M&B features, which was also the best performing feature for HMM-based word recognition.

The second finding from this analysis of the use of different descriptive features was that invariant properties, while arguably desirable, do not appear to have a strong positive effect on the performance of the recognisers considered in this study. In fact, the opposite appears to be true with GMs, which are invariant to translation, rotation and scale being the worst overall performing feature. On the other hand, the M&B features, which are not invariant to rotation, scale or translation were the best performing features for HMM-based recognition, while HoGs, which are only invariant to translation, were the best performing features for SVM and ANN-based recognition.

Overall, this analysis has shown that the choice of a descriptive features for handwriting recognition is largely dependent on the machine learning algorithm being employed and should be considered within this context. Furthermore, it has shown that some descriptive features provide consistent and relatively average performances for all machine learning algorithms. However, since good performance is desirable, it is better to select features that perform well for the machine learning algorithm being employed, thereby maximising performance.

7.3.2 Hybrid Features

The use of hybrid features was investigated with each of the machine learning algorithms used in this study. Four hybrid features were investigated for each machine learning algorithm. The first of these was created by combining all 3 best performing individual features, while the other 3 were created by pairing the 3 best performing features. The performance of these hybrid features was then investigated. Table 7.7 shows the performance of the best performing individual feature for each machine learning algorithm, the best performing hybrid feature for each machine learning algorithm, and the relative change (in percent) between the two.

Table 7.7: Performance of individual features compared to hybrid features for all machine learning algorithms

MLA	Best individual	Best Hybrid	Change
SVM Words	58.49%	57.77%	-1.23%
ANN Words	53.76%	53.60%	-0.29%
HMM Words	44.47%	44.79%	0.72%
HMM Lines (Acc.)	35.70	34.36	-3.76%

Table 7.7 shows that, in all cases except HMM-based word recognition, the use of hybrid features had a negative effect on the performance of the recogniser. For HMM-based word recognition the improvement was only very slight at less than 1%. Intuition suggests that hybrid features should improve the performance of a recogniser rather than decrease performance since it would be expected that combining multiple features would results in an increase in descriptive power. However, it instead appears to lead to a decrease in descriptive power. A possible reason for this could be that the combined features actually introduce increased variation within classes through multiple descriptors. Another possible explanation could be that the three best features all summarise the same information and thus combining them does not actually add any additional descriptive power. Another possible reason, still, is that the increase in descriptors from hybrid features results in more parameters that need to be trained for the machine learning algorithms and thus the parameters could be over tuned to fit the training data.

While these findings suggest that hybrid features do not offer any improvement on the best performing individual features, care should be taken not to generalise the findings. There are several reasons for this. Firstly, in the results presented in previous sections in this chapter, it was shown that hybrid features did improve the performance of some of the individual features, though not the best performing individual features, thereby suggesting that hybrid features can in fact improve the performance of a recogniser, though that did not occur in this study. Secondly, the hybrid features used in this study only represent a small subset of the potentially infinite number of hybrid features that can be created by combining the previously mentioned hundreds of features used for handwriting recognition. Lastly, equal weightings were used when combining the features and it is possible that weighting the features differently could have an effect on recognition accuracy, for instance, by having some features provide the weighted majority of the information and others only being supplementary. Thus these findings should be taken in context. In this study, hybrid features, in general, offered no improvement over the best performing individual features and, in most cases, led to a decrease in

performance. However, it is possible that under different circumstances they may lead to an improvement.

7.3.3 Amount of Training Data

A generally accepted rule of thumb for handwriting recognition is that the amount of training data that is available has an effect on the performance of a handwriting recognition system (Rowley et al., 2002). In this study, the effect of the amount of training data was investigated in two ways. The first of these ways was through the use of synthetic training data in which the training data set was supplemented with synthetic data that was created using a simple set of transformations, which are discussed in Section 6.1.5. Table 7.8 shows the performance of the different machine learning algorithms trained with the best performing features with and without synthetic training data as well as the relative change in performance (in percent) as a result of the introduction of synthetic training data.

Table 7.8: Performance with and without synthetic training data for all machine learning algorithms

MLA	Without Synthetic	With Synthetic	Change
SVM Words	58.49%	62.58%	7%
ANN Words	53.76%	57.61%	7.17%
HMM Words	44.79%	51.61%	15.23%
HMM Lines	35.7	33.67	-5.68%

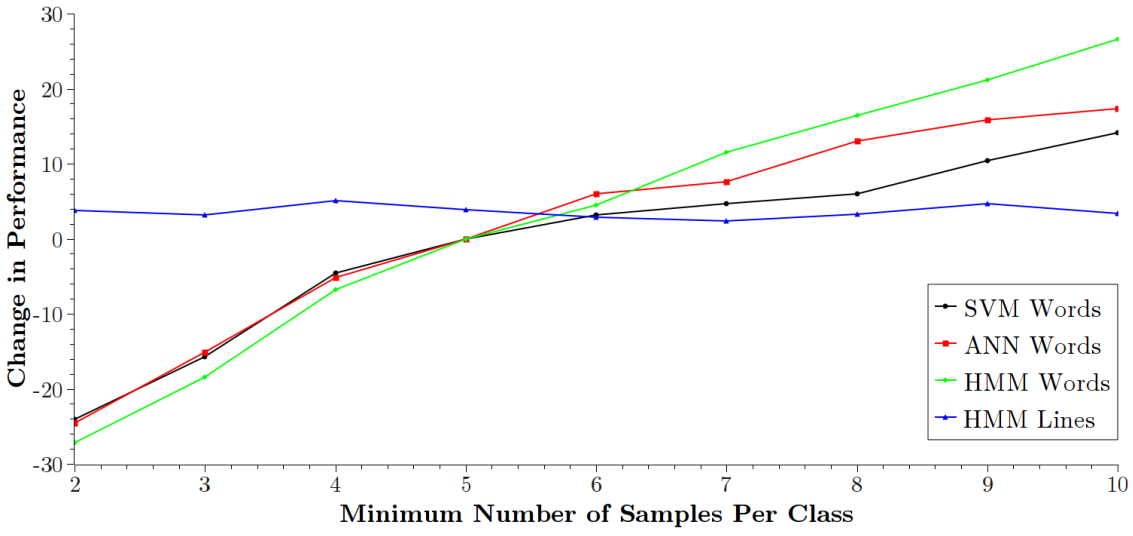
As can be seen from Table 7.8, synthetic training data led to an improvement in performance for all word recognition machine learning algorithms. The improvement was about 7% for SVM and ANN-based word recognition and 15.23% for HMM-based word recognition. Table 7.8 shows that the SVM and ANN-based word recognisers are equally sensitive to the addition of synthetic training data. The HMM-based recogniser, on the other hand, is about twice as sensitive to the addition of synthetic training data compared to the other recognisers. These findings suggest that the HMM-based recogniser may have better recognition accuracies when large amounts of training data are available - an idea that is explored further in this section. Overall, however, the SVM-based word recogniser is still the best performing recogniser even though its performance increase is not as great as that of the HMM-based word recogniser. The results are not as positive for HMM-based line recognition, where there was a decrease in the recognition accuracy.

The synthetic training data used in this study was relatively easy to create and involved simple shear operations. The generally positive effect that it had on the performance of the different recognisers was in line with other studies (Cano et al., 2002) and suggests that its use is worthwhile, especially when the amount of training data is limited. The increased amount of training data that exists when synthetic data is created does lead to an increase in the amount of time required to train recognition models. However, the training of models is usually a once-off process and the increase in training time can be justified by the increase in performance.

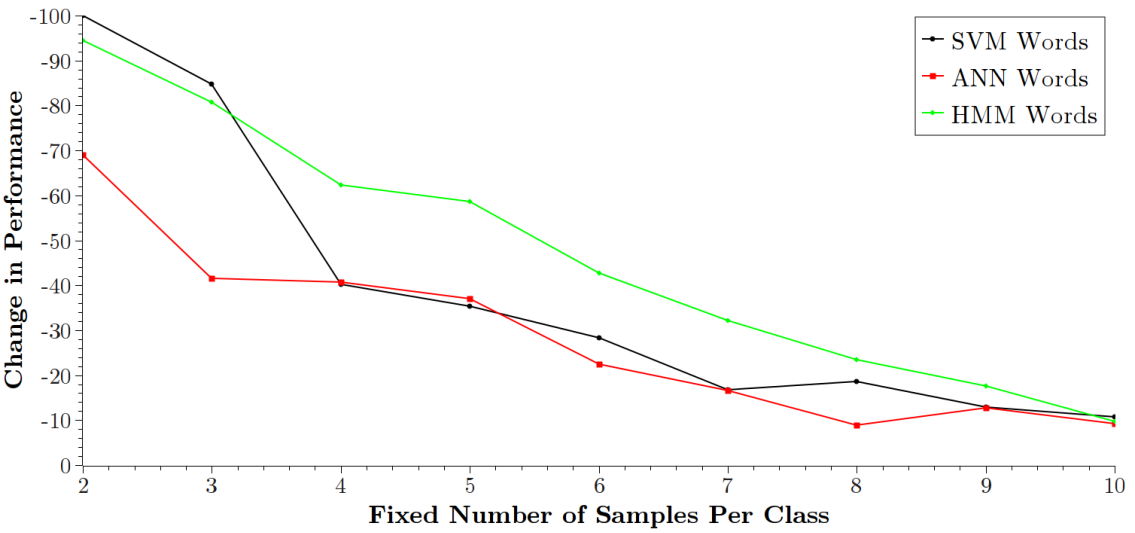
The second way in which the effect that the amount of training data had on the performance of the different recognisers was investigated was by varying the amount of real

training data that was used. The way in which this was done differed slightly for word recognition and for text line recognition. For word recognition, the effect of a fixed number of samples per class was investigated and for text line recognition the effect of varying the size of the corpus by random sampling was investigated. Then, for both cases, the effect of the minimum number of samples per class was investigated. The way in which this was done is described in Chapter 6.

Figure 7.31 shows relative change in performance (in percent) of varying the amount of real training data for the different machine learning algorithms.



(a) Minimum number of samples per class



(b) Fixed number of samples per class

Figure 7.31: Effect of varying the number of training data on the performance of all machine learning algorithms

The sub-corpora are much smaller when the number of samples per class is fixed as opposed to being set at a minimum. For these smaller corpora, the SVM and ANN-based recognisers are very sensitive to the initial increase in the amount of training data after

which their sensitivity starts to stabilise as the fixed number of samples per class increases. The HMM-based word recogniser, on the other hand, shows consistent sensitivity to the number of samples per class. When a minimum number of samples per class is required, the word recognisers initially all appear to be equally sensitive to the amount of training samples; however, beyond some point, the sensitivity begins to vary with the HMM-based word recogniser once again being the most sensitive to the amount of training data available.

Figure 7.31 shows that, for text line recognition, the HMM-based recogniser was not very sensitive to the minimum number of samples per class. That being said, requiring that there be at least 2 samples per class did lead to a slight improvement over when only a single sample per class was allowed.

These findings suggest that, when the amount of training data available is limited, using synthetic training data could be beneficial. Furthermore, the SVM and ANN-based recognisers seem most appropriate for small training sets, though as the amount of training data increases it may be worthwhile to consider the HMM-based recogniser as it has a higher rate of improvement as the size of the training set increases. For text line recognition, the addition of synthetic training data had a negative effect on the performance of the HMM-based recogniser. A possible reason for this was that the addition of synthetic training data contributed to overfitting. However, it was shown that there was a very small positive relationship between the amount of real training data and the performance of the recogniser, though not as great as that for word recognition.

7.3.4 Multiple Authors

The effect that multiple authors had on the performance of the different recognisers was investigated through a series of experiments, the results of which were presented in previous sections in this chapter. Table 7.9 provides a comparison of the effect of multiple authors for all of the machine learning algorithms.

Table 7.9: Performance for single and multiple authors for all machine learning algorithms

MLA	Single Author	Multiple Authors	Change
SVM Words	58.49%	55.4%	-5.28%
ANN Words	53.76%	52.98%	-1.44%
HMM Words	44.79%	44.13%	-1.48%
HMM Lines	35.70	30.04	-15.85%

Table 7.9 shows that, for word recognition, the SVM-based recogniser is the most sensitive to the effect of multiple authors with the addition of a second author leading to a 5.28% decrease in performance. For the ANN and HMM-based word recognisers the effect of multiple authors led to a much smaller decrease in performance of approximately 1.5%. These findings suggest that the ANN and HMM-based recognisers are better at handling the increased variation, both within and between classes, that is caused by the inclusion of the handwriting of additional authors. The SVM-based word recogniser is still the best performing recogniser for 2 authors; however, these findings allow for speculation that, if more authors were to be added beyond the second, then the performance of the ANN-based recogniser could surpass that of the SVM-based recogniser.

Text line recognition was much more sensitive to the effect of multiple authors than word recognition. One of the reasons for this is that there is a lot more variation between the text lines and there was less control over the contents of the corpus as is discussed further in the next section.

In order to handle multiple authors, steps can be taken to attempt to minimise the variation between their handwriting. Some of these steps, such as the slant correction described in Section 4.2.3.2, have been applied in this study. However, it is still possible to apply additional transformations to further reduce variation and thus possibly increase the performance of a recogniser built to recognise the handwriting of multiple authors.

7.3.5 The Effect of the Corpus

The corpus used in a handwriting recognition study will, naturally, have an effect on the performance of the recognisers used in the study. Some of the ways in which a corpus can affect the performance of recognisers have already been discussed, such as the amount of training data available and the variation between and within classes. This section seeks to explain the ways in which the corpus used in this study has affected the results achieved.

One of the reasons why the recognition of Bushman texts is difficult relates to the complex diacritics that appear in the text and which were discussed in detail in Chapter 4. The Bushman character set in this study is largely made up of Latin characters that have diacritics appearing above and below them. One would also expect that, in many cases, the variation between classes was actually quite low and could be attributed, to a large extent, to the diacritics in the text rather than the base characters. Thus, the recognition problem became largely about differentiating between diacritics, which, in many cases, only occupied a small area in the Cartesian space. Thus, the recognition process was made especially difficult due to the need to differentiate among many similar classes.

How small the variation among classes actually was became clear after an analysis of the corpus, as described in Chapter 4. The analysis revealed that the consistency of data capturers was estimated to be around 75%, as measured by the Levenshtein distance (Levenshtein, 1966). That is, data capturers only agreed on what symbols represented 75% of the time. This inconsistency between data capturers is testament to the difficulty of the problem where humans, with all of their cognitive ability, fail to agree on the classification of about a quarter of the symbols that appear in the text. The holy grail in handwriting recognition is the design of recognition systems that are able to equal humans in their ability to recognise and classify symbols. One can then go on to reason that, given the difficulty that humans had in agreeing on what symbols represented, machine learning algorithms would also struggle with the same task.

The difficulty that the data capturers had in classifying the symbols that appear in the Bushman texts also had a direct effect on the performance of the HMM-based text line recogniser used in this study. The transcriptions created by the data capturers were used both for the training of recognition models and as the gold standard to which the output of the recognisers was compared and against which performance was evaluated. Given that the estimated consistency among data capturers was 75%, it could be reasoned that, for any transcription output, the probability of a false negative or positive is 25%. For instance, if the output of the recogniser matched the ground truth perfectly, there would still be a chance that it would not match the gold standard due to the inconsistencies

within the gold standard. In this context, the ground truth refers to the absolute true transcription of the data whereas the gold standard refers to the corpus described in Chapter 4. The effect of the disagreement between data capturers does not only affect the recognition step. The training step is also affected and it could be argued that in approximately 25% of cases, a symbol would have been classified by different data capturers as belonging to different classes. Thus the effect of the inconsistencies is two-fold. In the first sense it is likely to have led to poorly trained recognition models where multiple occurrences of a single symbol were classified as being different and, secondly, it would have led to false positives and negatives when evaluating the recogniser output. Thus, when considering the recognition results achieved in this study, the effect of the corpus should be considered. However, one could argue that, even with a gold standard that closely resembles the ground truth, it would be unreasonable to expect recognition accuracies on par with those achieved for other scripts with fewer symbol classes and it would be more appropriate to compare the performance to other scripts where there are a large number of classes that need to be recognised. Furthermore, it could be argued that performance on par with other handwriting recognition studies is not feasible due to the unique properties and complexities of the Bushman texts.

7.4 Discussion

This section has presented the results of the experiments that were conducted to investigate the use of different techniques for Bushman handwriting recognition. The experiments showed that, for Bushman word recognition, the highest recognition accuracy of 62.58% occurred when a SVM was trained with real and synthetic training data from which HoGs had been extracted and that, for Bushman text line recognition, the highest recognition accuracy of 45.10 was achieved when M&B features were extracted from only real training data and a bigram word language model was incorporated into the recogniser. The experiments also revealed a number of properties about the different machine learning algorithms, such as how different machine learning algorithms have different levels of sensitivity to the amount of training data and the variation within the training data.

In this study, Bushman words were recognised as a single pattern rather than being recognised by the symbols that they were made up of. A possible alternative approach to word recognition could have been taken where the individual symbols in each word were recognised and then, in a manner similar to that for text line recognition, a dictionary or character language model could be used to concatenate the symbols to create Bushman words. By doing this, the statistical information about the Bushman language, which was shown to result in an improvement in Bushman text line recognition, could potentially also be used to improve Bushman word recognition. However, this approach was not used and, instead, the Bushman word recognition task instead focused on whole Bushman words as a single pattern, while text line recognition focused on recognising the individual symbols that appear in the texts.

A number of assumptions were made during the execution of the experiments in this chapter. For instance, it was assumed that the best features for each machine learning algorithm remained the best even when different training sets were used, as was the case for synthetic data, the number of samples and multiple authors. However, it is possible

that the best features may actually have differed for the different sets of training data. For SVM and ANN-based recognition, the model parameters were determined from the training data at the start of each experiment. For the SVM-based recognition this was done through a grid search and for ANN-based recognition the architecture was based on the size of the training vector and number of classes that needed to be recognised. However, for HMM-based recognition, an experiment was conducted in order to determine the HMM model parameters using two features and the assumption was made that these parameters were optimal not only for other features but also for both text line and word recognition. It could be argued that an assumption such as this is somewhat flawed since one would expect that the HMM parameters would, to some extent, be dependent on the features and the data being recognised. However, in a study such as this one it is infeasible to consider all possible combinations of machine learning algorithms and their parameters and features and their parameters and thus assumptions have to be made. However, the generally consistent results for most features across all machine learning algorithms showed that, to a certain extent, these assumptions were reasonable.

Chapter 8

Conclusion

There are a number of techniques and strategies that can be employed when designing a handwriting recognition system and the choice as to which techniques and strategies to use can be decided in a number of ways. For instance, a careful analysis and comparison of techniques can be conducted in order to identify which of them are the most appropriate for the recognition task at hand. Alternatively, the decision could be based on the findings of previous studies with the assumption that the findings are transferable. There are pros and cons to both of these approaches. In the first case, the best suited approach for the task at hand can be identified, but it comes at the cost of considering a large number of techniques. In the second case, not as much exploratory and experimental work needs to be done. However, this comes at the cost of making an assumption of transferable findings, which may or may not actually be the case and it is less likely that an optimal solution will be found.

Chapter 2 in this thesis discussed some of these previous techniques that have been used for automatic handwriting and revealed that there was no easy way of determining which techniques should be used for a given handwriting recognition study and why it was difficult to compare the findings of independent studies. Thus, in this study, the first approach was taken where various techniques were investigated for their use for Bushman handwriting recognition. These techniques included the use of: Support Vector Machines (SVMs), Artificial Neural Networks (ANNs) and Hidden Markov Models (HMMs) as machine learning algorithms; Undersampled Bitmaps, Geometric Moments, Marti & Bunke Features, Histograms of Oriented Gradients, Gabor Filter-based Features and Discrete Cosine Transform coefficients as features; synthetic training data; and statistical language models. The ways in which these techniques were used was discussed in Chapters 5 and 6, while Chapter 7 presented a set of experiments and their findings.

It was found that, of the techniques considered in this study, the best word recognition accuracy of 62.58% occurred when a SVM was trained with real and synthetic training data from which Histograms of Oriented Gradients had been extracted. For text line recognition, the best recognition accuracy of 45.10 was achieved when a HMM was trained with Marti & Bunke features that were extracted from only real training data and a bigram word language model was incorporated into the recogniser. Thus, of the techniques considered, these can be recommended as being the best for automatically transcribing handwritten Bushman texts. Furthermore, it is likely that these techniques may also perform well for similar scripts, especially those that contain complex diacritics.

The research questions posed in Chapter 1 dealt with: machine learning algorithms and

features; training data; and statistical language models. These research questions were addressed through the experiments in this study and are discussed below.

Machine Learning Algorithms and Features

The research question on machine learning algorithms and features was related to which were best for the automatic transcription of handwritten Bushman texts. It has already been mentioned how a SVM trained with Histograms of Oriented Gradients as features resulted in the highest recognition accuracy for Bushman words and how a HMM trained with Marti & Bunke features resulted in the highest recognition accuracy for Bushman text lines. However, in addition to this, there were other observations about machine learning algorithms and features. For instance, it was observed that there were some features that, on average, performed equally well for the different machine learning algorithms. However, there were some features that performed better with a specific machine learning algorithm suggesting that the choice as to which features should be used in a handwriting recognition system should be based on the machine learning algorithms that are to be used. Furthermore, it is possible that the diacritics had a significant effect on the performance of the different recognisers; investigating this is left for future work and discussed in Section 8.1. Lastly, it was also found that the hybrid features used in this study to did not lead to an improvement in recogniser performance, though this could possibly be attributed to the fact that the best features all summarised the same information.

Training Data

The research question on training data related to the effect that training data has on the performance of a recogniser. It is well-known that the amount of training data used has an effect on the performance of a recogniser; however, it was shown that some machine learning algorithms are more sensitive than others to the amount of training data that is available. It was also shown how synthetic training data, which is easy and simple to create, is beneficial since it leads to an improvement in the performance of a recogniser. It was also shown how increased variation in the data, caused by multiple authors, led to a decrease in recogniser performance. However, as was the case with the amount of training data, some machine learning algorithms are more sensitive than others to this increase in variation. Thus, it can be concluded that training data has a large effect on the performance of a recogniser, but the actual effect that is has is dependent on the machine learning algorithms themselves.

Statistical Language Models

The research question on statistical language models related to the effect that they have on the recognition accuracy that can be achieved. It was shown how a bigram language model led to an improvement in recognition accuracy by just under 10, thus clearly demonstrating that language models are beneficial and can be used to improve the performance of a recogniser.

The study revealed some other considerations related to the Bushman texts and automatic handwriting recognition in general. For instance, in order to create a handwriting system, a corpus is needed to train the machine learning algorithms and create recognition models.

Chapter 4 in this thesis described how a corpus of Bushman texts was created and used in this study. For a complex script like the Bushman script, it was shown how data capturers struggled to agree on what the symbols represented. Furthermore, many of the symbols in the Bushman texts appeared infrequently and it is well known that multiple samples of a symbol are required to create robust recognition systems.

Thus, while automatic handwriting recognition systems have successfully been used for well-known and well-understood scripts with relatively small character sets, perhaps they cannot be applied as successfully to more complex scripts. For instance, large collections of documents are likely to exist for well-known and well-understood scripts, thereby allowing for large amounts of training data; whereas more complex scripts are more likely to belong to smaller collections. Furthermore, data that is relatively uniform is needed in order to create robust recognition systems. This was not the case for the Bushman texts and may not be the case for other complex scripts. Thus, for smaller collections, it may be more appropriate to simply manually transcribe the collections rather than go through the difficulties of trying to create automatic recognition systems. Given the difficulty that non-expert data capturers may have in transcribing complex scripts, as was the case with the Bushman texts as described in Chapter 4, it may be necessary that the transcription of the texts is performed by experts.

One may argue that, given the complexities of the Bushman script and other scripts, it is not worth creating automatic handwriting recognition systems since the accuracies that can be achieved are perhaps too low to be very useful or the complexities in designing the system and investigating techniques are not worth the effort. While this study has shown that the recognition of the Bushman texts is difficult, it has also shown how different techniques can be used in a handwriting recognition system and the effect that they have. It has shown how some features are better than others for use in handwriting recognition and how different machine learning algorithms have different levels of sensitivity to data. Furthermore, it has also shown that there is no single best approach for designing a handwriting recognition system, but that the design depends on a number of factors. This information could be used for the design of future recognition systems, be they for simple handwriting recognition, complex handwriting recognition, or pattern recognition in general.

Furthermore, in addition to investigating techniques for automatic handwriting recognition, the problem of digitally encoding and representing the Bushman texts was also solved as part of this study. The novel technique that was developed allows for the Bushman text to be fully encoded and can be used for the transcription of the texts. Furthermore, the technique and the tool that was created may also be applicable to other complex scripts that cannot be represented in Unicode and thus could have application beyond the Bushman texts.

Finally, the importance of transcriptions of historical documents should not be underestimated. It has already been discussed how transcriptions can be used to improve digital library systems by allowing for the provision of enhanced services. However, more importantly, transcriptions of historical documents serve as a means of preserving the documents' contents and ensuring that, regardless of what may happen to the physical artefacts, the knowledge and information that they contain will remain available and accessible to all. This study has taken us one step closer to achieving that goal.

8.1 Future Work

There are many possibilities for future work that arise from this study such as an investigation into the use of additional machine learning algorithms and descriptive features for Bushman handwriting recognition. However, this study has shown why the automatic recognition of Bushman texts is difficult and thus future work would do better to focus on better understanding the texts and developing techniques to deal with their complexities.

For instance, the nature and the meaning of the diacritics that appear in the Bushman texts were not investigated directly as it fell outside of the scope of this study. However, the diacritics had a large effect on the performance of the recognisers, from their initial transcription to their recognition. Thus, in future studies it may be worthwhile to further investigate the effect that these diacritics have on recognition, for instance, what the effect is if they are ignored during recognition or removed, recognised separately from the base characters, and then re-attached as a post-processing step. This was partly investigated in the second pilot study described in Chapter 3 where the exclusion of diacritics was found to be infeasible due to their perceived importance in the meaning of the Bushman words. However, it is possible that the diacritics are perhaps not as important as one may think. Future work could possibly exploit linguistic information about the Bushman languages in order to determine the importance of diacritics and how they should be treated during recognition.

The issue of diacritics is also related to the features used for recognition in this study. Different features are likely to contain different information content and, as such, some are better at describing handwritten text than others. However, some features may be better at describing diacritics that are attached to base characters and that take up a relatively small part of the Cartesian space. Further exploration may allow for additional information about features to be discovered. For instance, the effect of the features could be investigated by determining the recognition accuracy both including and excluding diacritics. This would give a rough overview of the extent to which the features actually include the information content of the diacritics. Furthermore, the features could be analysed in order to determine what their actual information content is and could potentially reveal correlations that exist between the information content and recognition accuracy. In doing this, it may be possible to determine what information content it is desirable for features to have and, using this information, build hybrid features that maximise this desirable information. This could be used to overcome the shortcoming of the best features all potentially summarising the same information. Of course, these findings would not be limited to the recognition of handwritten texts, but would also generalise to pattern recognition in general, where the information content of features plays an important role.

Statistical information about a natural language provides a valuable source of information about the distribution of words and characters in the language and can be used to improve the recognition process. In this study, a bigram word language model was used for text line recognition and it was shown how it led to a large increase in the performance of the recogniser. However, it is possible to create higher order language models or even character language models, which would contain additional statistical information to that provided by a bigram language model. The use of these higher order language models and statistical information about the Bushman languages, in addition to potentially improving the performance of automatic recognisers, can also assist in furthering our understanding of the Bushman languages by means of statistical analysis.

Lastly, crowd-sourcing, in which a task is outsourced to a distributed and large group of people has, in many applications, proven to be more accurate than professionals (Surowiecki, 2005). Crowd-sourcing could be used as an approach to transcribing the Bushman texts by adapting xo̱ä'xo̱ä for use on top of a volunteer computing platform. In fact, efforts have already begun to do this and, as a result, make the Bushman texts, and the indigenous knowledge of some of the earliest inhabitant of the Earth, available to the world.

Bibliography

Mehmmood Abd and George Paschos. Effective Arabic character recognition using support vector machines. In *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, pages 7–11. Springer Netherlands, 2007.

N. Ahmed, T. Natarajan, and K.R. Rao. Discrete Cosine Transfom. *IEEE Transactions on Computers*, C-23(1):90 – 93, 1974.

S. Al-Ma'adeed, D. Elliman, and C.A. Higgins. A data base for Arabic handwritten text recognition research. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 485 – 489, 2002.

J.H. AlKhateeb, Jinchang Ren, Jianmin Jiang, S.S. Ipson, and H. El Abed. Word-based handwritten Arabic scripts recognition using DCT features and neural network classifier. In *Proceedings of the Fifth International Multi-Conference on Systems, Signals and Devices*, pages 1–5, 2008.

A. Apatean, A. Rogozan, and A. Bensrhair. Objects recognition in visible and infrared images from the road scene. In *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics*, volume 3, pages 327–332. IEEE Computer Society, 2008.

V. N. Manjunath Aradhya and C. Naveena. Text line segmentation of unconstrained handwritten Kannada script. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, pages 231–234. ACM, 2011.

Nafiz Arica. An off-line character recognition system for free style handwriting, 1998. MSc Thesis, The Middle East Technical University.

Sameh M. Awaidah and Sabri A. Mahmoud. A multiple feature/resolution scheme to Arabic (Indian) numerals recognition using hidden Markov models. *Signal Processing*, 89(6):1176–1184, 2009.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM, 1992.

A. Brakensiek and G. Rigoll. A comparison of character N-grams and dictionaries used for script recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 241–245. IEEE Computer Society, 2001.

Horst Bunke. Recognition of cursive Roman handwriting - past, present and future. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 1, pages 448–459. IEEE Computer Society, 2003.

Javier Cano, Juan Carlos Pérez-Cortes, Joaquim Arlandis, and Rafael Llobet. Training set expansion in handwritten character recognition. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 548–556. Springer-Verlag, 2002.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.

Donald H. Burn Chang Shu. Artificial neural network ensembles and their application in pooled flood frequency analysis. *Water Resources Research*, 40:W09301, 2004.

Jin Chen, Huaigu Cao, Rohit Prasad, Anurag Bhardwaj, and Prem Natarajan. Gabor features for offline Arabic handwriting recognition. In *Proceedings of the Ninth IAPR International Workshop on Document Analysis Systems*, pages 53–58. ACM, 2010.

Yi-Kai Chen and Jhing-Fa Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1304–1317, 2000.

Michael Hüsken Christian Igel. Improving the Rprop learning algorithm. In *Proceedings of the Second International Symposium on Neural Computation*, pages 115–121. Academic Press, 2000.

W.F. Clocksin and P.P.J. Fernando. Towards automatic transcription of Syriac handwriting. In *Proceedings of the International Conference on Image Analysis and Processing*, pages 664–669. IEEE Computer Society, 2003.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995.

Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines: and other kernel-based learning methods*. Cambridge University Press, 2000.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE Computer Society, 2005.

Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15:11–15, 1972.

eHeritage. Proceedings of the second workshop on eHeritage and digital art preservation. ACM, 2010.

S. Espana-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4):767 –779, 2011.

J. T. Favata and G. Srikantan. A multiple feature/resolution approach to handprinted digit and character recognition. *International Journal of Imaging Systems and Technology*, 7(4):304–311, 1996.

S. L. Feng and R. Manmatha. Classification models for historical manuscript recognition. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 528–532. IEEE Computer Society, 2005.

Andreas Fischer, Markus Wthrich, Marcus Liwicki, Volkmar Frinken, Horst Bunke, Gabriel Viehhauser, and Michael Stolz. Automatic transcription of handwritten medieval documents. In *Proceedings of the International Conference on Virtual Systems and MultiMedia*, pages 137–142. IEEE Computer Society, 2009.

Andreas Fischer, Emanuel Indermühle, Horst Bunke, Gabriel Viehhauser, and Michael Stolz. Ground truth creation for handwriting recognition in historical documents. In *Proceedings of the Ninth IAPR International Workshop on Document Analysis Systems*, pages 3–10. ACM, 2010.

Jan Flusser, Tomáš Suk, and Barbara Zitov. *Moments and Moment Invariants in Pattern Recognition*. John Wiley and Sons, 2009.

M. Frigo and S.G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216 –231, 2005.

Volkmar Frinken and Horst Bunke. Self-training for handwritten text line recognition. In *Proceedings of the 15th Iberoamerican Congress Conference on Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 104–112, Berlin, Heidelberg, 2010. Springer-Verlag.

Volkmar Frinken, Tim Peter, Andreas Fischer, Horst Bunke, Trinh-Minh-Tri Do, and Thierry Artieres. Improved handwriting recognition by combining two forms of hidden Markov models and a recurrent neural network. In *Proceedings of the Thirteenth International Conference on Computer Analysis of Images and Patterns*, pages 189–196. Springer-Verlag, 2009.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31: 855–868, 2009.

Simon Günter and Horst Bunke. Optimizing the number of states, training iterations and gaussians in an HMM-based handwritten word recognizer. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 1, pages 472–476, Washington, DC, USA, 2003. IEEE Computer Society.

Sofiene Haboubi, Samia Maddouri, Noureddine Ellouze, and Hailkal El-Abed. Invariant primitives for handwritten Arabic script: A contrastive study of four feature sets. In *Proceedings of the Tenth International Conference on Document Analysis and Recognition*, pages 691–697. IEEE Computer Society, 2009.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, 2001.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1998.

L. Heutte, T. Paquet, J. V. Moreau, Y. Lecourtier, and C. Olivier. A structural/statistical feature based vector for handwritten character recognition. *Pattern Recognition Letters*, 19(7):629–641, 1998.

Nicholas R. Howe, Shaolei Feng, and R. Manmatha. Finding words in alphabet soup: Inference on freeform character recognition for historical scripts. *Pattern Recognition*, 42(12):3338–3347, 2009.

Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. *A practical guide to support vector classication*, 2003.

Ming Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:187–179, 1962.

Chen Huang and Sargur N. Srihari. Word segmentation of off-line handwritten documents. In *Proceedings of SPIE Document Recognition and Retrieval XV*, page 68150, 2008.

Emanuel Indermühle, Marcus Liwicki, and Horst Bunke. Recognition of handwritten historical documents: HMM-Adaptation vs. writer specific training. In *Proceedings of The Eleventh International Conference on Frontiers in Handwriting Recognition*, pages 186–191, 2008.

S. Johansson, G. Leech, , and H. Goodluck. *Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers*, 1978. Department of English, University of Oslo.

Amit Roy John F. Pitrelli. Creating word-level language models for handwriting recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 721–725. IEEE Computer Society, 2001.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, 2nd Ed.* Prentice Hall, 2008.

Shaun Kane, Andrew Lehman, and Elizabeth Partridge. Indexing george washingtons handwritten manuscripts. Technical report, University of Massachusetts Amherst, 2001.

Amjad Rehman Khan and Zulkifli Mohammad. A simple segmentation approach for unconstrained cursive handwritten words in conjunction with the neural network. *Image Processing*, 2:29–35, 2008.

Syed Ali Khayam. The Discrete Cosine Transform (DCT): Theory and application. Technical report, Department of Electrical & Computer Engineering, Michigan State University, 2003.

Fumitaka Kimura, Tetsushi Wakabayashi, Shinji Tsuruoka, and Yasuji Miyake. Improvement of handwritten Japanese character recognition using weighted direction code histogram. *Pattern Recognition*, 30(8):1329–1337, 1997.

T. Yung Kong and Azriel Rosenfeld, editors. *Topological Algorithms for Digital Image Processing*. Elsevier Science Inc., 1996.

Jayant Kumar, Wael Abd-Almageed, Le Kang, and David Doermann. Handwritten Arabic text line segmentation using affinity propagation. In *Proceedings of the Ninth IAPR International Workshop on Document Analysis Systems*, pages 135–142. ACM, 2010a.

Munish Kumar, R. K. Sharma, and M. K. Jindal. Segmentation of lines and words in handwritten Gurmukhi script documents. In *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*, pages 25–28. ACM, 2010b.

Victor Lavrenko, Toni M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries*, pages 278–287. IEEE Computer Society, 2004.

Roberta A. Lee and Michael J. Balick. Indigenous use of hoodia gordonii and appetite suppression. *EXPLORE: The Journal of Science and Healing*, 3(4):404 – 406, 2007.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

L. Likforman-Sulem, A. Hanimyan, and C. Faure. A Hough based algorithm for extracting text lines in handwritten documents. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 2, pages 774 –777, 1995.

Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. Text line segmentation of historical documents: a survey. *International Journal on Document Analysis and Recognition*, 9(2):123–138, 2007.

Cheng-Lin Liu, Masashi Koga, and Hiromichi Fujisawa. Gabor feature extraction for character recognition: comparison with gradient feature. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 121–125. IEEE Computer Society, 2005.

Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. Chinese handwriting recognition contest 2010. In *Chinese Conference on Pattern Recognition*, pages 1–5, 2010.

Liana M. Lorigo and Venu Govindaraju. Offline Arabic handwriting recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(5):712–724, 2006.

G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169 – 3183, 2009.

A. S Bhaskarabhatla M. Agrawal and S. Madhvanath. Data collection for handwriting corpus creation in Indic scripts. In *Proceedings of the International Conference on Speech and Language Technology and Oriental COCOSDA*, 2004.

U. Mahadevan and R.C. Nagabushnam. Gap metrics for word separation in handwritten lines. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 1, pages 124–127, 1995.

Michael Mahemoff. *Ajax Design Patterns*. O'Reilly Media, Inc., 2006.

M. Makridis, N. Nikolaou, and B. Gatos. An efficient word segmentation technique for historical and degraded machine-printed documents. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, volume 1, pages 178–182. IEEE Computer Society, 2007.

R. Manmatha and Nitin Srimal. Scale space technique for word segmentation in handwritten documents. In *Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision*, pages 22–33. Springer-Verlag, 1999.

U. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 705–708, 1999.

U. Marti and H. Bunke. On the influence of vocabulary size and language models in unconstrained handwritten text recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 260–265. IEEE Computer Society, 2001.

U. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems. In *Hidden Markov models: applications in computer vision*, pages 65–90. World Scientific Publishing Co., Inc., 2002.

U.-V. Marti. *Off-line recognition of handwritten texts.* PhD thesis, University of Bern, Switzerland, 2000.

A. Nabatchian, E. Abdel-Raheem, and M. Ahmadi. Human face recognition using different moment invariants: a comparative study. In *Proceedings of the Congress on Image and Signal Processing*, volume 3, pages 661–666. IEEE Computer Society, 2008.

D. Nguyen and T.D. Bui. On the problem of classifying Vietnamese online handwritten characters. In *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, pages 803–808, 2008.

S. Nissen. Implementation of a Fast artificial Neural Network Library (fann). Technical report, Department of Computer Science, University of Copenhagen (DIKU), 2003. http://fann.sf.net.

Mark S. Nixon and Alberto S. Aguado. *Feature extraction and image processing.* Academic Press, 2008.

Oliveira, Mello, Silva, and Alves. Optical digit recognition for images of handwritten historical documents. In *Ninth Brazilian Symposium on Nueral Networks*, pages 166–171, Ribeirao Preto, Brazil, 2006.

U. Pal and S. Datta. Segmentation of Bangla unconstrained handwritten text. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 1128–1132, 2003.

Moisés Pastor, Alejandro Hector Toselli, and Enrique Vidal. Projection profile based algorithm for slant removal. In *Proceedings of the International Conference on Image Analysis and Recognition*, pages 183–190, 2004.

William K. Pratt. *Digital Image Processing.* Wiley-Intersciences, 2007.

L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

L. Ragha and M. Sasikumar. Adapting moments for handwritten Kannada Kagunita recognition. In *Second International Conference on Machine Learning and Computing*, pages 125–129, 2010.

Toni M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 1, pages 218–222. IEEE Computer Society, 2003.

Tony M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9:139–152, 2007.

Gunnar Rätsch. A brief introduction into machine learning. In *Proceedings of a Congress on Chaos Communication 21C3*, 2008.

F. Rei. TIPA: A system for processing phonetic symbols in LaTeX. *TUGboat*, 17(2): 102–114, 1996.

M. Riedmiller. Rprop - description and implemetation details. Technical report, Institut fur Logik, Komplexitat und Deduktionssyteme, 1994.

Henry A. Rowley, Manish Goyal, and John Bennett. The effect of large training set sizes on online Japanese Kanji and English cursive recognizers. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 36–40. IEEE Computer Society, 2002.

J. Ruiz-Pinales and E. Lecolinet. Cursive handwriting recognition using the Hough transform and a neural network. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 231–234, 2000.

Kim Penman Sargur N. Srihari. Penman : A system for reading unconstrained handwritten page images. In *Symposium on Document Image Understanding Technology*, pages 142–153, 1997.

T. Sari, L. Souici, and M. Sellami. Off-line handwritten Arabic character segmentation algorithm: ACSA. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 452–457, 2002.

W. S. Sarle. Neural Network FAQ - periodic posting to the Usenet newsgroup comp.ai.neural-nets, 1997. ftp://ftp.sas.com/pub/neural/FAQ.html.

Mita Nasipuri Satadal Saha, Subhadip Basu and Dipak Kr. Basu. A Hough transform based technique for text segmentation. *The Computing Research Repository*, abs/1002.4048, 2010.

Giovanni Seni and Edward Cohen. External word segmentation of off-line handwritten text lines. *Pattern Recognition*, 27:41–52, 1994.

S. Setlur, S. Kompalli, V. Ramanaprasad, and V. Govindaraju. Creation of data resources and design of an evaluation test bed for Devanagari script recognition. In *Proceedings of the 13th International Workshop on Research Issues in Data Engineering: Multi-lingual Information Management*, pages 55 – 61, March 2003.

Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

L.G. Shapiro and G.C. Stockman. *Computer vision*. Prentice Hall, 2001.

Shailedra Kumar Shrivastava and Sanjay S. Gharde. Support vector machine for handwritten Devanagari numeral recognition. *International Journal of Computer Applications*, 7(11):9–14, October 2010.

Dayashankar Singh, Maitreyee Dutta, and Sarvpal H. Singh. Neural network based handwritten Hindi character recognition system. In *Proceedings of the 2nd Bangalore Annual Compute Conference*, pages 15:1–15:4. ACM, 2009.

P. Skotnes. *Claim to the Country: the Archive of Wilhelm Bleek and Lucy Lloyd*. Jacana Media and Ohio University Press, 2007.

Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 257–286, 2002.

Tong-Hua Su, Tian-Wen Zhang, and Zhao-Wen Qiu. HMM-based system for transcribing Chinese handwriting. In *Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 6, pages 3412–3417, 2007.

Hussein Suleman. Digital libraries without databases: The Bleek and Lloyd Collection. In *Research and Advanced Technology for Digital Libraries*, pages 392–403, 2007.

J. Surowiecki. *The wisdom of crowds: why the many are smarter than the few*. Abacus, 2005.

Y.H. Tay, M. Khalid, R. Yusof, and C. Viard-Gaudin. Offline cursive handwriting recognition system based on hybrid Markov model and neural networks. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, volume 3, pages 1190–1195, 2003.

M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):401–410, 1993.

A. Toselli, V. Romero, L. Rodriguez, and E. Vidal. Computer assisted transcription of handwritten text images. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 2, pages 944–948. IEEE Computer Society, 2007.

Unesco. Convention concerning the protection of the world cultural and natural heritage. 1972.

G. Vamvakas, B. Gatos, N. Stamatopoulos, and S.J. Perantonis. A complete optical character recognition methodology for historical documents. In *Proceedings of the IAPR International Workshop on Document Analysis Systems*, pages 525–532. IEEE Computer Society, 2008.

Tamás Varga and Horst Bunke. Generation of synthetic training data for an HMM-based handwriting recognition system. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 1, pages 618–622. IEEE Computer Society, 2003.

Alessandro Vinciarelli, Samy Bengio, and Horst Bunke. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):709–720, 2004.

Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21:168–173, 1974.

Xuewen Wang, Xiaoqing Ding, and Changsong Liu. Gabor filters-based feature extraction for character recognition. *Pattern Recognition*, 38(3):369–379, 2005.

Kyle Williams. Feasibility of automatic transcription of neatly rewritten bushman texts. Technical report, Department of Computer Science, University of Cape Town, 2010.

Kyle Williams and Hussein Suleman. Translating handwritten bushman texts. In *Proceedings of the Tenth Annual International ACM/IEEE Joint Conference on Digital Libraries*, pages 109–118. ACM, 2010.

Kyle Williams and Hussein Suleman. Using a hidden Markov model to transcribe handwritten bushman texts. In *Proceedings of the Eleventh Annual International ACM/IEEE Joint Conference on Digital Libraries*, pages 445–446. ACM, 2011.

Kyle Williams, Sanvir Manilal, Lebogang Molwantoa, and Hussein Suleman. A visual dictionary for an extinct language. In *The Role of Digital Libraries in a Time of Global Change*, pages 1–4, 2010.

K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26:647–656, 1982.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying A. Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book (for HTK Version 3.4)*. Cambridge University Engineering Department, 2006.

Li Yujian and Liu Bo. A normalized Levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1091–1095, 2007.

Matthias Zimmermann and Horst Bunke. Hidden markov model length optimization for handwriting recognition systems. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 369–374. IEEE Computer Society, 2002.

George Kingsley Zipf. *Human Behaviour and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley, 1949.

Appendices

Appendix A

Diacritics Supported by xǿäʼxǿä

Characters		
ɾ̀ \textlonglegr{}	τ \tshape{}	T̄ \tshapebar{}
◌͡◌ \stackedu{}	‖ \textdoublepipe{}	ǂ \textdoublebarpipe{}
ɜ \textrevepsilon{}	θ \texttheta{}	ɣ \textfrtailgamma{}
ʘ \textbullseye{}	ǁ \textdoublepipebar{}	# \texthash{}
#ʸ \texthashy{}		

Diacrics Above Characters		
a̍̀ \dbarelipline{a}	a͡͡ \kappiebrackets{a}	a̿̀ \kappieline{a}
â \kappie{a}	ã̰ \welip{a}	a̍̀ \elipbarline{a}
a̯̍̀ \elipuline{a}	a̯̍ \elipu{a}	a̿̀ \elipbar{a}
a̍́ \elipubar{a}	a̍̀ \elipdbar{a}	a̍̀ \elipline{a}

\uelipdbar{a}

\uelipline{a}

\uelip{a}

\uuline{a}

\uline{a}

\barelipdbar{a}

\barelipline{a}

\barelip{a}

\baruline{a}

\barwu{a}

\bardbar{a}

\barubar{a}


\barline{a}

\barbar{a}

\barbar{a}

\barbar{a}

\circleelip{a}

\circleu{a}

\twodotselipu{a}

\twodotselip{a}

\twodotsuelip{a}

\twodotsuline{a}

\twodotsu{a}

\twodotsline{a}

\twodotsdbar{a}

\twodotsubar{a}

\twodotsbarelipline{a}

\twodotsbarline{a}

\twodotsbar{a}

\onedotu{a}

\onedotdbar{a}

\onedotuline{a}

\seagull{a}

\plusover{a}

\kover{a}

\epover{a}

\cover{a}

\xover{a}

udkappie{a}|


\ukappie{a}

\kappie{a}

\wover{a}

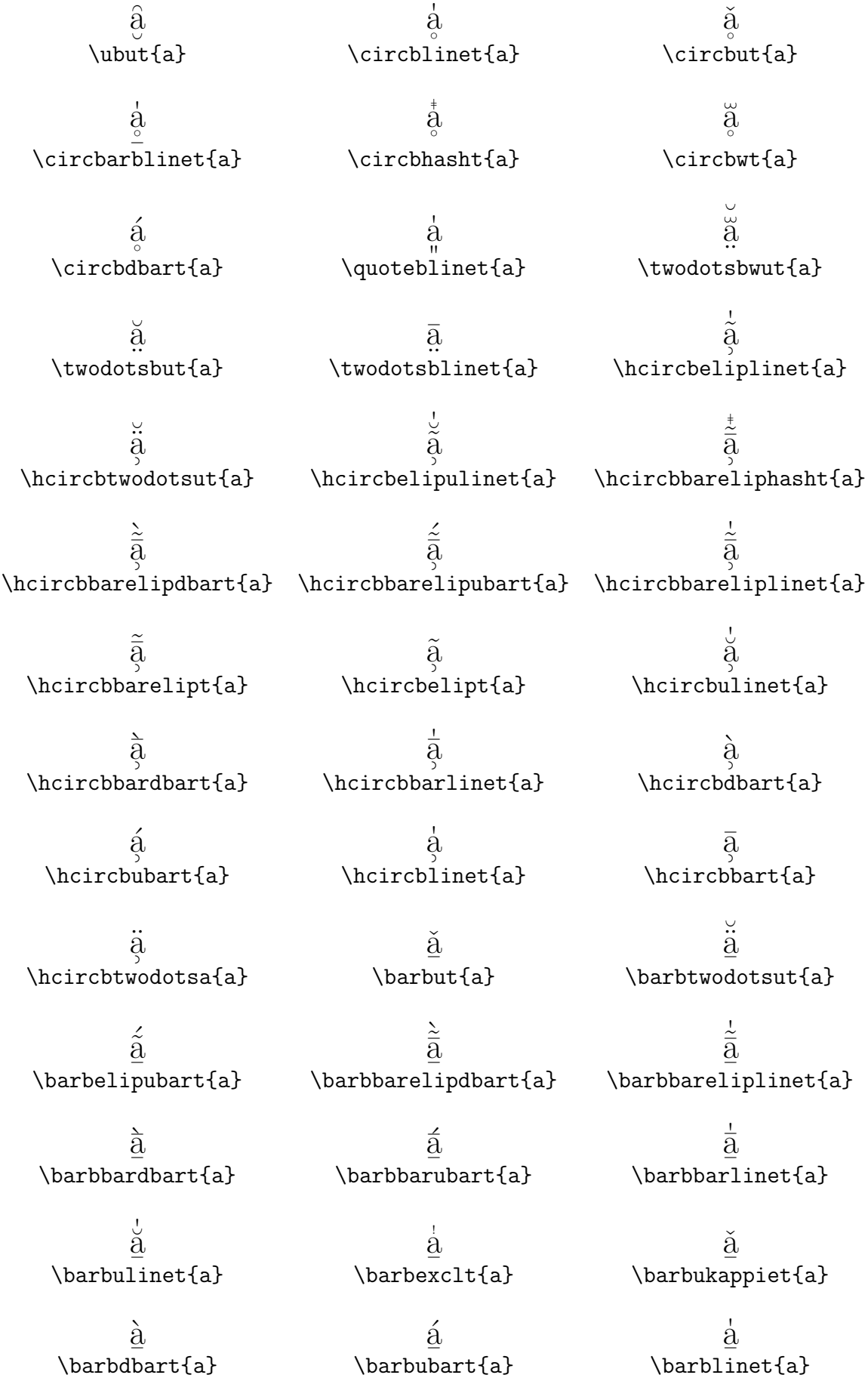
\elip{a}	\u{a}	\ubar{a}
\dbar{a}	\dialine{a}	\diabar{a}
\r{a}	\excl{a}	\twodots{a}
\onedot{a}		

Diacrics Below Characters

\xcbelow{a}	\harpoonb{a}	\xbelow{a}
\ybelow{a}	\linebelow{a}	\cbelow{a}
\ebelow{a}	\tildelinebelow{a}	\circlelinebelow{a}
\ubelow{a}	\dquotebelow{a}	\hcirclebelow{a}
\twobarsbelow{a}	\barbelow{a}	\circlebelow{a}
\twodotsbelow{a}		

Diacrics Above and Below Characters

\harpoonb{a}	\barbdotulinet{a}	\barbtwodotsbart{a}
\xbrackbdotulinet{a}	\wbut{a}	\bartildebbart{a}
		
\twobarsbhasht{a}	\ubarbdott{a}	\ubbarlinet{a}


\ubut{a}
\circblinet{a}
\circbut{a}
\circbarblinet{a}
\circbhasht{a}
\circbwt{a}
\circbdbart{a}
\quoteblinet{a}
\twodotsbwut{a}
\twodotsbut{a}
\twodotsblinet{a}
\hcircbeliplinet{a}
\hcircbtwodotsut{a}
\hcircbelipulinet{a}
\hcircbbareliphasht{a}
\hcircbbarelipdbart{a}
\hcircbbarelipubart{a}
\hcircbbareliplinet{a}
\hcircbbarelipt{a}
\hcircbelipt{a}
\hcircbulinet{a}
\hcircbbardbart{a}
\hcircbbarlinet{a}
\hcircbdbart{a}
\hcircbubart{a}
\hcircblinet{a}
\hcircbbart{a}
\hcircbtwodotsa{a}
\barbut{a}
\barbtwodotsut{a}
\barbelipubart{a}
\barbbarelipdbart{a}
\barbbareliplinet{a}
\barbbardbart{a}
\barbbarubart{a}
\barbbarlinet{a}
\barbulinet{a}
\barbexclt{a}
\barbukappiet{a}
\barbdbart{a}
\barbubart{a}
\barblinet{a}

$\underline{\bar{a}}$

```
\barbt{a}
```

Appendix B

Results of Experiment Investigating Hidden Markov Model Parameters

	Fold										
	1	2	3	4	5	6	7	8	9	10	Avg.
Gaussians	**Discrete Cosine Transform Features**										
	4 States										
8	-47.1	-54.3	-41.9	-47.8	-43.9	-50.2	-51.0	-47.6	-51.4	-50.3	-48.5
16	-38.9	-45.7	-36.2	-45.2	-38.4	-44.3	-43.2	-40.5	-43.7	-47.0	-42.3
24	-40.5	-50.8	-36.6	-47.7	-38.2	-42.2	-40.3	-39.0	-42.3	-46.9	-42.5
32	-39.8	-58.3	-35.6	-47.5	-42.5	-42.9	-40.6	-39.5	-42.3	-47.5	-43.6
	8 States										
8	18.6	15.6	19.1	18.5	21.7	18.8	19.1	18.5	18.7	15.9	18.5
16	21.5	17.9	22.3	21.3	23.7	22.2	21.9	20.7	19.7	19.4	21.1
24	21.8	19.6	21.6	21.6	24.1	21.0	23.1	21.4	20.8	19.0	21.4
32	21.3	18.6	21.2	21.7	24.3	21.1	22.3	20.5	20.9	18.8	21.1
	12 States										
8	27.1	24.8	27.4	25.2	28.9	26.5	27.5	26.9	25.7	24.1	26.4
16	29.1	24.9	30.1	27.2	29.8	29.0	29.3	28.6	28.6	26.6	28.3
24	28.6	25.3	30.0	27.1	30.4	28.8	29.3	28.7	29.3	26.9	28.4
32	28.4	24.4	30.0	27.1	30.8	28.8	29.4	29.3	29.3	27.0	28.4
Gaussians	**Marti & Bunke Features**										
	4 States										
8	-17.5	-23.3	-15.5	-19.5	-17.2	-18.8	-17.7	-22.0	-20.3	-16.6	-18.8
16	-15.0	-23.8	-12.9	-17.8	-14.2	-15.7	-15.5	-17.7	-17.8	-14.7	-16.5
24	-13.2	-23.4	-12.1	-17.9	-13.3	-15.1	-15.0	-18.9	-17.3	-15.5	-16.2
32	-12.8	-22.0	-11.1	-18.4	-11.1	-13.5	-15.5	-18.3	-15.0	-16.1	-15.4
	8 States										
8	28.6	27.8	30.9	27.2	30.2	28.8	29.4	27.8	30.3	28.0	28.9
16	30.5	27.2	30.6	28.2	32.6	29.0	30.9	30.2	30.5	29.2	29.9
24	30.1	26.5	31.5	28.1	31.7	29.4	30.4	30.3	30.6	29.1	29.8
32	29.9	27.2	31.2	28.2	31.5	30.8	30.7	30.1	30.8	29.4	30.0
	12 States										
8	36.9	34.8	36.7	35.2	38.4	36.5	36.7	35.6	35.2	34.9	36.1
16	37.7	36.4	37.9	36.5	39.4	37.0	38.3	36.3	37.1	36.5	37.3

24	37.6	36.4	38.0	37.1	39.2	38.9	39.0	36.2	37.4	36.9	37.7
32	37.7	36.7	38.5	36.7	39.3	39.2	39.1	36.2	37.6	37.5	37.8

Appendix C

Results of Experiments Using Support Vector Machines for Word Recognition

	Fold										
	1	2	3	4	5	6	7	8	9	10	Avg.
Cell Size	**Undersampled Bitmaps (UBs)**										
32x32	31.2	16.0	20.0	27.2	24.0	24.0	25.6	28.0	23.2	21.1	24.0
16x16	47.2	36.0	41.6	47.2	45.6	41.6	42.4	40.8	46.4	42.3	43.1
8x8	58.4	46.4	47.2	50.4	57.6	48.8	49.6	43.2	50.4	55.3	50.7
4x4	56.8	42.4	42.4	46.4	52.8	48.8	47.2	47.2	44.8	48.8	47.8
	Marti & Bunke (M&B) Features										
	51.2	46.4	42.4	44.0	52.8	50.4	47.2	47.2	48.8	42.3	47.3
Cell Size	**Geometric Moments (GMs)**										
	Moments = M1										
64x64	12.0	5.6	12.0	12.0	5.6	9.6	5.6	11.2	12.0	9.8	9.5
32x32	16.8	12.8	8.8	8.0	6.4	16.0	7.2	12.8	6.4	6.5	10.2
16x16	20.0	17.6	18.4	30.4	22.4	21.6	19.2	24.0	20.8	17.1	21.1
8x8	32.8	24.0	25.6	34.4	26.4	28.0	25.6	27.2	31.2	24.4	28.0
	Moments = M1,M2										
64x64	21.6	8.0	7.2	12.0	8.8	9.6	4.0	12.0	8.0	8.9	10.0
32x32	17.6	10.4	8.0	6.4	3.2	16.0	8.0	12.0	6.4	6.5	9.5
16x16	19.2	18.4	20.8	27.2	21.6	28.8	19.2	22.4	21.6	17.9	21.7
8x8	31.2	24.0	26.4	24.0	24.0	26.4	21.6	24.8	28.0	22.0	25.2
	Moments = M1,M2,M3										
64x64	20.0	7.2	10.4	14.4	8.0	8.8	8.8	6.4	8.8	9.8	10.3
32x32	20.0	8.0	6.4	5.6	2.4	14.4	4.8	8.8	7.2	8.1	8.6
16x16	20.0	19.2	17.6	25.6	20.8	28.8	20.0	17.6	20.0	15.4	20.5
8x8	27.2	22.4	25.6	28.0	23.2	24.0	22.4	24.0	28.0	19.5	24.4
	Moments = M1,M2,M3,M4										
64x64	24.0	8.0	12.8	15.2	12.8	6.4	7.2	12.8	12.0	8.9	12.0
32x32	18.4	10.4	4.8	4.0	3.2	12.8	5.6	7.2	4.8	7.3	7.9
16x16	20.0	20.8	17.6	23.2	21.6	20.0	17.6	17.6	18.4	13.0	19.0

8x8	28.8	23.2	24.0	28.0	21.6	28.0	21.6	23.2	28.8	18.7	24.6
Resolutions	**Histograms of Oriented Gradients (HoGs)**										
64x32x16	67.2	56.8	57.6	56.8	64.0	60.8	56.8	56.8	55.2	50.4	58.2
32x16x8	66.4	58.4	53.6	55.2	64.8	60.8	55.2	54.4	60.8	55.3	58.5
16x8x4	62.4	52.0	50.4	52.0	54.4	46.4	54.4	47.2	52.0	48.0	51.9
64x32x0	57.6	48.8	44.8	50.4	55.2	53.6	46.4	47.2	50.4	43.1	49.7
32x16x0	68.0	56.8	55.2	56.8	64.8	59.2	56.0	56.0	56.0	50.4	57.9
16x8x0	65.6	54.4	52.8	54.4	62.4	60.0	56.8	54.4	60.0	53.7	57.4
64x0x0	25.6	20.0	24.8	16.0	21.6	19.2	24.0	28.0	29.6	16.3	22.5
32x0x0	54.4	48.0	43.2	51.2	52.8	51.2	44.0	48.0	51.2	43.1	48.7
16x0x0	65.6	55.2	55.2	57.6	63.2	56.8	56.0	58.4	56.8	52.8	57.8
	Signed										
	68.0	52.0	54.4	55.2	63.2	56.0	56.8	52.8	60.0	55.3	57.4
	Normalised										
	63.2	52.0	56.0	52.0	60.0	59.2	59.2	58.4	57.6	57.7	57.5
Cell Size	**Gabor Filter (GF)-based Features**										
32x32	50.4	35.2	40.0	35.2	38.4	45.6	32.0	33.6	39.2	35.0	38.5
16x16	52.0	36.8	46.4	48.8	50.4	48.0	47.2	47.2	49.6	43.9	47.0
8x8	61.6	43.2	50.4	55.2	57.6	53.6	50.4	53.6	52.8	53.7	53.2
4x4	59.2	43.2	46.4	52.8	56.8	54.4	52.0	54.4	53.6	52.0	52.5
Cell Size	**Discrete Cosine Transform (DCT) Coefficients**										
	Coefficients = 1										
64x64	9.6	9.6	12.0	17.6	12.0	14.4	11.2	14.4	14.4	12.2	12.7
32x32	7.2	22.4	26.4	25.6	19.2	33.6	8.8	23.2	26.4	18.7	21.1
16x16	50.4	34.4	42.4	44.8	51.2	44.8	42.4	42.4	45.6	48.0	44.6
8x8	52.8	51.2	49.6	48.0	52.0	48.0	50.4	45.6	52.0	52.0	50.2
	Coefficients = 10										
64x64	36.8	40.8	43.2	37.6	41.6	44.0	40.0	40.8	44.8	43.1	41.3
32x32	55.2	48.8	47.2	47.2	53.6	50.4	44.0	48.0	47.2	53.7	49.5
16x16	59.2	44.0	44.8	45.6	48.8	51.2	48.8	44.8	44.8	47.2	47.9
8x8	52.0	40.8	37.6	47.2	44.0	47.2	45.6	43.2	40.0	43.1	44.1
	Coefficients = 20										
64x64	54.4	50.4	46.4	47.2	53.6	50.4	49.6	46.4	51.2	48.8	49.8
32x32	60.8	45.6	50.4	48.0	48.0	48.8	48.0	44.0	50.4	51.2	49.5
16x16	50.4	41.6	39.2	43.2	48.8	47.2	42.4	44.0	42.4	37.4	43.7
8x8	45.6	36.8	35.2	40.8	37.6	43.2	36.0	38.4	39.2	37.4	39.0
	Coefficients = 30										
64x64	57.6	49.6	44.8	48.8	50.4	48.0	48.8	50.4	47.2	52.0	49.8
32x32	56.0	41.6	40.8	41.6	44.8	46.4	44.0	43.2	44.0	44.7	44.7
16x16	48.0	37.6	36.0	41.6	40.0	44.0	44.0	41.6	38.4	36.6	40.8
8x8	39.2	36.8	32.8	40.0	36.8	40.8	32.8	36.8	40.0	35.8	37.2
	Coefficients = 40										
64x64	60.8	43.2	46.4	48.8	50.4	48.0	40.0	46.4	46.4	48.0	47.8
32x32	55.2	37.6	38.4	39.2	44.8	43.2	47.2	42.4	42.4	42.3	43.3
16x16	41.6	35.2	32.0	43.2	41.6	39.2	38.4	34.4	39.2	34.1	37.9
8x8	36.8	38.4	28.8	39.2	34.4	40.0	32.8	35.2	39.2	32.5	35.7
	Coefficients = 50										
64x64	59.2	44.0	46.4	49.6	52.0	50.4	48.0	43.2	50.4	51.2	49.4

32x32	52.0	36.0	36.0	37.6	44.0	39.2	44.8	41.6	40.0	39.8	41.1
16x16	40.0	36.8	33.6	40.0	35.2	41.6	34.4	33.6	36.8	31.7	36.4
8x8	36.8	34.4	27.2	39.2	35.2	39.2	27.2	32.8	35.2	31.7	33.9
Features	**Hybrid Features**										
HoGs,GF,UBs	64.0	52.0	55.2	55.2	59.2	56.8	52.8	55.2	56.0	57.7	56.4
HoGs,GF	65.6	52.8	52.0	55.2	64.0	56.8	57.6	56.8	57.6	56.9	57.5
HoGs,UBs	64.8	53.6	54.4	56.0	60.8	56.8	57.6	58.4	59.2	56.1	57.8
GF,UBs	62.4	44.8	49.6	55.2	56.8	54.4	51.2	51.2	52.0	56.9	53.5
	Synthetic Data										
	68.8	60.0	57.6	56.0	69.6	63.2	65.6	61.6	62.4	61.0	62.6
Samples	**Number of Samples**										
	Minimum Samples										
2	35.1	48.0	49.7	43.3	43.3	45.0	44.4	44.4	48.0	43.0	44.4
3	49.0	47.0	53.6	45.0	50.3	44.4	53.0	56.3	51.7	42.9	49.3
4	52.9	55.9	51.5	55.9	52.9	61.0	55.1	55.1	60.3	57.6	55.8
5	66.4	58.4	53.6	55.2	64.8	60.8	55.2	54.4	60.8	55.3	58.5
6	62.2	63.0	64.7	59.7	58.8	57.1	60.5	52.9	65.5	59.0	60.4
7	61.9	65.5	64.6	61.9	67.3	49.6	58.4	59.3	67.3	56.2	61.2
8	59.6	58.7	71.6	63.3	61.5	57.8	66.1	63.3	54.1	63.6	62.0
9	66.0	59.2	65.0	62.1	70.9	67.0	71.8	64.1	55.3	63.9	64.5
10	64.3	71.4	64.3	66.3	59.2	73.5	66.3	72.4	69.4	60.2	66.7
	Fixed Samples										
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	8.3	8.3	16.7	33.3	22.2	8.9
4	25.0	12.5	25.0	50.0	31.3	25.0	50.0	31.3	50.0	50.0	35.0
5	45.0	35.0	35.0	50.0	40.0	40.0	25.0	35.0	20.0	53.3	37.8
6	29.2	41.7	20.8	37.5	58.3	37.5	58.3	37.5	37.5	61.1	41.9
7	53.6	50.0	39.3	46.4	60.7	57.1	46.4	35.7	50.0	47.6	48.7
8	50.0	43.8	53.1	43.8	53.1	53.1	37.5	43.8	43.8	54.2	47.6
9	55.6	44.4	58.3	36.1	50.0	52.8	47.2	63.9	52.8	48.1	50.9
10	55.0	45.0	42.5	57.5	50.0	42.5	70.0	55.0	55.0	50.0	52.3
	Multiple Authors										
	53.4	59.0	52.8	56.5	53.4	61.5	56.5	53.4	59.0	48.4	55.4

Appendix D

Results of Experiments Using Artificial Neural Networks for Word Recognition

	Fold										
	1	2	3	4	5	6	7	8	9	10	Avg.
Cell Size	**Undersampled Bitmaps (UBs)**										
	Architecture: Inputs										
32x32	27.2	16.0	20.8	21.6	16.8	26.4	22.4	22.4	19.2	14.6	20.7
16x16	50.4	35.2	34.4	40.8	40.8	38.4	36.8	35.2	40.8	37.4	39.0
8x8	55.2	40.0	46.4	46.4	49.6	50.4	47.2	42.4	45.6	43.9	46.7
4x4	56.0	34.4	38.4	41.6	48.8	44.0	45.6	41.6	44.8	43.9	43.9
	Architecture: Inputs/2										
32x32	20.0	12.8	16.0	22.4	15.2	25.6	14.4	20.0	19.2	8.9	17.5
16x16	42.4	28.8	28.0	40.0	41.6	36.0	30.4	31.2	38.4	35.8	35.3
8x8	54.4	41.6	43.2	47.2	48.0	48.8	44.0	40.8	48.8	45.5	46.2
4x4	56.8	36.8	37.6	44.8	44.0	44.8	46.4	44.0	40.8	42.3	43.8
	Architecture: Outputs										
32x32	26.4	19.2	13.6	21.6	21.6	18.4	21.6	24.0	19.2	21.1	20.7
16x16	46.4	30.4	43.2	40.0	41.6	44.0	42.4	40.0	42.4	45.5	41.6
8x8	52.0	38.4	44.8	47.2	51.2	43.2	48.0	44.0	44.8	44.7	45.8
4x4	49.6	38.4	37.6	40.8	44.0	44.0	39.2	44.0	42.4	39.0	41.9
	Architecture: Outputs/2										
32x32	27.2	18.4	16.0	21.6	23.2	20.8	26.4	24.0	22.4	19.5	22.0
16x16	45.6	34.4	44.0	46.4	47.2	42.4	43.2	43.2	47.2	43.9	43.8
8x8	61.6	38.4	42.4	51.2	49.6	47.2	45.6	40.8	44.8	43.9	46.6
4x4	51.2	39.2	38.4	40.8	41.6	42.4	44.0	38.4	40.0	44.7	42.1
	Architecture: (Input+Outputs)/2										
32x32	30.4	20.0	13.6	23.2	25.6	20.0	24.8	25.6	20.8	21.1	22.5
16x16	46.4	32.0	40.0	44.0	44.8	42.4	42.4	40.8	43.2	43.9	42.0
8x8	50.4	39.2	43.2	48.0	51.2	45.6	46.4	44.0	48.8	47.2	46.4
4x4	50.4	35.2	36.0	39.2	46.4	44.0	44.0	44.0	44.0	44.7	42.8
	Marti & Bunke (M&B) Features										

	Architecture: Inputs										
	43.2	35.2	36.8	40.8	44.8	42.4	38.4	44.8	36.8	37.4	40.1
	Architecture: Inputs/2										
	44.0	40.8	34.4	41.6	48.0	40.0	35.2	42.4	39.2	37.4	40.3
	Architecture: Outputs										
	43.2	32.8	40.0	41.6	42.4	43.2	38.4	39.2	41.6	38.2	40.1
	Architecture: Outputs/2										
	41.6	32.8	32.8	44.8	40.0	39.2	40.0	38.4	37.6	32.5	38.0
	Architecture: (Inputs+Outputs)/2										
	43.2	36.8	34.4	40.0	47.2	36.0	36.8	40.0	42.4	37.4	39.4
Cell Size	**Geometric Moments (GMs)**										
	Moments = M1										
	Architecture: Inputs										
64x64	11.2	5.6	8.8	12.8	6.4	13.6	5.6	12.8	13.6	8.9	9.9
32x32	12.0	6.4	8.8	8.8	8.8	14.4	12.0	20.0	13.6	9.8	11.5
16x16	18.4	11.2	17.6	18.4	19.2	20.8	16.0	21.6	17.6	14.6	17.5
8x8	31.2	18.4	21.6	25.6	22.4	29.6	21.6	24.0	24.0	19.5	23.8
	Architecture: Inputs/2										
32x32	12.0	5.6	8.8	8.8	8.8	15.2	11.2	16.8	11.2	9.8	10.8
16x16	18.4	10.4	18.4	20.0	20.0	19.2	16.8	20.0	15.2	14.6	17.3
8x8	27.2	14.4	20.0	25.6	21.6	28.8	22.4	20.8	21.6	18.7	22.1
	Architecture: Outputs										
64x64	11.2	7.2	11.2	15.2	7.2	10.4	4.0	13.6	11.2	8.9	10.0
32x32	12.0	7.2	16.8	13.6	12.0	18.4	15.2	24.0	13.6	13.0	14.6
16x16	20.0	10.4	16.0	20.0	19.2	21.6	16.8	27.2	16.8	14.6	18.3
8x8	28.0	17.6	20.8	27.2	21.6	28.0	21.6	24.0	23.2	20.3	23.2
	Architecture: Outputs/2										
64x64	11.2	6.4	11.2	15.2	5.6	10.4	4.0	13.6	11.2	8.9	9.8
32x32	12.8	6.4	15.2	12.0	11.2	18.4	14.4	24.0	17.6	13.8	14.6
16x16	20.0	11.2	16.0	20.0	19.2	24.0	17.6	27.2	17.6	15.4	18.8
8x8	29.6	14.4	21.6	24.8	23.2	28.8	21.6	24.8	26.4	20.3	23.6
	Architecture: (Inputs+Outputs)/2										
64x64	11.2	6.4	11.2	15.2	5.6	10.4	4.0	13.6	11.2	8.9	9.8
32x32	12.8	6.4	16.0	14.4	11.2	16.8	15.2	24.0	12.8	11.4	14.1
16x16	21.6	10.4	16.0	20.0	19.2	23.2	16.0	25.6	18.4	13.8	18.4
8x8	28.8	16.8	21.6	24.8	24.0	29.6	21.6	24.0	24.0	19.5	23.5
	Moments = M1,M2										
	Architecture: Inputs										
64x64	16.0	7.2	11.2	15.2	8.8	13.6	7.2	17.6	12.0	10.6	11.9
32x32	12.0	4.8	8.8	9.6	9.6	12.8	11.2	16.0	10.4	9.8	10.5
16x16	8.8	5.6	6.4	8.8	12.0	12.0	8.8	14.4	8.8	7.3	9.3
8x8	10.4	8.8	5.6	9.6	9.6	12.0	9.6	12.0	12.8	8.9	9.9
	Architecture: Inputs/2										
64x64	14.4	6.4	8.8	8.0	8.0	14.4	12.8	16.8	11.2	8.9	11.0
32x32	12.0	5.6	8.8	8.8	8.8	12.0	11.2	16.8	11.2	9.8	10.5
16x16	8.8	4.0	7.2	7.2	12.8	12.8	8.8	14.4	8.8	8.1	9.3
8x8	10.4	6.4	6.4	9.6	8.0	12.0	10.4	12.0	11.2	10.6	9.7

	Architecture: Outputs										
64x64	14.4	7.2	8.0	5.6	8.0	1.6	4.0	5.6	12.8	2.4	7.0
32x32	12.0	4.8	8.0	8.8	9.6	12.0	10.4	15.2	10.4	9.8	10.1
16x16	9.6	5.6	6.4	8.0	12.0	12.0	9.6	13.6	8.0	8.1	9.3
8x8	12.0	7.2	6.4	11.2	8.8	10.4	8.8	13.6	11.2	9.8	9.9
	Architecture: Outputs/2										
64x64	15.2	7.2	8.8	4.8	7.2	1.6	3.2	8.8	11.2	2.4	7.0
32x32	12.0	4.8	8.0	8.8	8.8	12.8	11.2	16.0	10.4	9.8	10.3
16x16	9.6	5.6	6.4	8.8	12.0	12.8	9.6	13.6	8.8	8.1	9.5
8x8	10.4	4.0	5.6	10.4	8.8	11.2	8.8	12.0	9.6	8.1	8.9
	Architecture: (Inputs+Outputs)/2										
64x64	16.0	6.4	8.8	4.8	7.2	1.6	4.0	4.8	12.8	2.4	6.9
32x32	12.0	4.8	8.0	8.8	9.6	12.8	11.2	16.0	11.2	9.8	10.4
16x16	8.8	4.8	7.2	8.8	12.8	13.6	9.6	13.6	8.0	7.3	9.5
8x8	10.4	8.8	6.4	9.6	8.0	9.6	9.6	13.6	10.4	9.8	9.6
	Moments = M1,M2,M3										
	Architecture: Inputs										
64x64	8.0	8.0	9.6	10.4	10.4	10.4	4.0	12.0	12.0	12.2	9.7
32x32	12.0	6.4	7.2	8.8	8.8	14.4	12.0	14.4	10.4	10.6	10.5
16x16	8.8	4.8	7.2	10.4	8.8	11.2	8.8	12.0	8.8	7.3	8.8
8x8	10.4	4.8	7.2	7.2	8.8	11.2	8.8	12.0	9.6	6.5	8.7
	Architecture: Inputs/2										
64x64	11.2	8.0	7.2	8.0	10.4	15.2	6.4	12.0	12.0	10.6	10.1
32x32	12.0	8.0	9.6	8.8	8.8	14.4	11.2	15.2	9.6	9.8	10.7
16x16	8.8	4.8	7.2	10.4	9.6	11.2	8.8	13.6	8.8	7.3	9.1
8x8	10.4	4.8	7.2	8.0	8.8	11.2	8.8	12.0	9.6	6.5	8.7
	Architecture: Outputs										
64x64	8.0	8.0	9.6	12.0	11.2	12.0	5.6	12.0	12.0	10.6	10.1
32x32	11.2	4.8	7.2	8.8	8.8	14.4	12.0	12.8	9.6	10.6	10.0
16x16	8.8	4.8	6.4	8.0	9.6	13.6	8.8	13.6	8.8	7.3	9.0
8x8	11.2	4.8	7.2	8.0	8.8	11.2	8.8	12.0	9.6	6.5	8.8
	Architecture: Outputs/2										
64x64	8.0	8.8	9.6	12.0	11.2	12.0	5.6	10.4	12.0	8.9	9.9
32x32	12.0	4.8	7.2	8.8	8.8	14.4	11.2	12.8	10.4	11.4	10.2
16x16	8.8	4.8	7.2	8.8	8.0	11.2	9.6	13.6	8.8	7.3	8.8
8x8	11.2	4.8	8.0	8.0	8.8	11.2	8.8	12.8	9.6	6.5	9.0
	Architecture: (Inputs+Outputs)/2										
64x64	8.0	8.0	9.6	12.0	11.2	10.4	5.6	11.2	12.0	8.9	9.7
32x32	12.8	4.8	7.2	8.8	8.8	14.4	12.0	12.8	9.6	10.6	10.2
16x16	8.8	4.8	7.2	9.6	8.8	11.2	8.8	12.8	8.8	7.3	8.8
8x8	10.4	4.8	7.2	8.0	8.8	10.4	8.8	12.0	9.6	6.5	8.7
	Moments = M1,M2,M3,M4										
	Architecture: Inputs										
64x64	8.0	8.8	9.6	10.4	10.4	11.2	5.6	10.4	12.0	10.6	9.7
32x32	12.0	4.8	7.2	8.0	8.8	14.4	10.4	12.8	9.6	11.4	9.9
16x16	9.6	4.8	7.2	9.6	8.8	12.8	8.8	12.0	8.8	7.3	9.0
8x8	10.4	4.8	8.0	8.0	8.8	11.2	8.8	12.8	9.6	6.5	8.9
	Architecture: Inputs/2										

64x64	9.6	9.6	9.6	11.2	11.2	12.0	5.6	12.8	12.0	11.4	10.5
32x32	12.0	7.2	7.2	8.0	8.8	14.4	10.4	14.4	10.4	10.6	10.3
16x16	8.8	4.8	8.0	9.6	7.2	11.2	8.8	13.6	8.8	7.3	8.8
8x8	11.2	4.8	7.2	8.0	8.0	10.4	8.8	13.6	9.6	6.5	8.8
	Architecture: Outputs										
64x64	8.8	7.2	9.6	7.2	9.6	10.4	4.8	12.8	11.2	9.8	9.1
32x32	12.0	4.8	7.2	8.8	8.8	13.6	10.4	13.6	10.4	9.8	9.9
16x16	9.6	4.8	6.4	8.0	8.8	13.6	8.8	12.8	9.6	7.3	9.0
8x8	10.4	4.8	7.2	8.0	8.8	11.2	8.0	12.8	9.6	6.5	8.7
	Architecture: Outputs/2										
64x64	8.8	6.4	9.6	8.8	9.6	9.6	5.6	11.2	11.2	9.8	9.1
32x32	12.0	4.8	7.2	8.8	8.8	14.4	10.4	12.8	10.4	11.4	10.1
16x16	8.0	4.8	8.0	8.8	11.2	12.0	8.8	11.2	8.8	7.3	8.9
8x8	12.0	4.8	8.0	7.2	8.8	11.2	8.8	12.8	9.6	6.5	9.0
	Architecture: (Inputs+Outputs)/2										
64x64	8.8	7.2	9.6	10.4	9.6	7.2	5.6	10.4	11.2	9.8	9.0
32x32	12.0	5.6	7.2	8.8	8.8	14.4	10.4	12.8	8.8	10.6	9.9
16x16	9.6	4.8	8.0	10.4	8.8	13.6	8.8	13.6	8.8	7.3	9.4
8x8	10.4	4.8	7.2	8.0	8.8	10.4	8.8	12.0	9.6	6.5	8.7
Resolutions	**Histograms of Oriented Gradients (HoGs)**										
	Architecture: Inputs										
64x32x16	48.0	40.0	26.4	47.2	40.8	36.0	32.0	44.8	35.2	15.4	36.6
32x16x8	45.6	30.4	16.8	42.4	37.6	36.0	36.8	33.6	32.8	17.1	32.9
64x32x0	44.8	35.2	26.4	42.4	34.4	31.2	35.2	40.0	41.6	10.6	34.2
32x16x0	59.2	40.0	32.0	46.4	44.8	48.8	47.2	36.0	43.2	29.3	42.7
16x8x0	55.2	40.8	44.8	49.6	46.4	48.0	46.4	41.6	47.2	46.3	46.6
64x0x0	20.0	18.4	12.0	28.0	20.0	14.4	12.0	25.6	21.6	5.7	17.8
32x0x0	50.4	40.0	24.0	42.4	40.8	45.6	37.6	38.4	39.2	17.1	37.5
16x0x0	57.6	45.6	52.8	51.2	55.2	49.6	53.6	47.2	48.0	51.2	51.2
	Architecture: Inputs/2										
64x32x16	47.2	38.4	26.4	44.0	35.2	39.2	33.6	40.8	40.0	8.1	35.3
32x16x8	48.0	31.2	20.8	44.8	37.6	38.4	37.6	37.6	31.2	18.7	34.6
64x32x0	42.4	31.2	24.8	40.8	32.0	29.6	30.4	37.6	36.8	6.5	31.2
32x16x0	56.0	40.8	32.8	48.0	43.2	48.0	42.4	40.0	44.8	27.6	42.4
16x8x0	56.8	38.4	47.2	47.2	54.4	47.2	44.8	45.6	47.2	47.2	47.6
64x0x0	20.0	14.4	14.4	18.4	16.0	20.8	11.2	20.8	21.6	11.4	16.9
32x0x0	48.0	28.8	19.2	41.6	40.8	41.6	36.0	35.2	34.4	19.5	34.5
16x0x0	61.6	47.2	52.8	51.2	54.4	52.8	52.8	45.6	48.8	50.4	51.8
	Architecture: Outputs										
64x32x16	49.6	34.4	24.8	46.4	32.8	37.6	36.8	38.4	39.2	13.8	35.4
32x16x8	52.0	32.0	23.2	45.6	36.0	38.4	37.6	36.0	36.0	20.3	35.7
64x32x0	42.4	37.6	26.4	45.6	35.2	34.4	37.6	40.0	40.8	10.6	35.1
32x16x0	62.4	35.2	32.0	44.8	43.2	46.4	44.0	39.2	42.4	30.9	42.0
16x8x0	55.2	41.6	43.2	44.0	45.6	48.0	44.8	44.8	43.2	43.9	45.4
64x0x0	16.8	20.0	6.4	25.6	19.2	10.4	12.0	23.2	26.4	3.3	16.3
32x0x0	50.4	36.0	24.0	44.0	43.2	43.2	42.4	39.2	38.4	20.3	38.1
16x0x0	58.4	47.2	49.6	49.6	53.6	53.6	52.0	48.8	47.2	50.4	51.0
	Architecture: Outputs/2										

64x32x16	48.0	34.4	25.6	44.0	39.2	32.8	33.6	43.2	36.8	13.0	35.1
32x16x8	44.8	29.6	22.4	46.4	36.8	38.4	40.0	36.8	30.4	22.8	34.8
64x32x0	45.6	36.0	28.8	45.6	36.8	32.8	35.2	41.6	39.2	6.5	34.8
32x16x0	54.4	34.4	29.6	48.8	42.4	47.2	45.6	39.2	40.0	27.6	40.9
16x8x0	48.8	36.8	42.4	44.0	44.0	44.0	41.6	36.8	43.2	40.7	42.2
64x0x0	17.6	20.8	11.2	28.8	20.0	11.2	12.8	26.4	27.2	4.1	18.0
32x0x0	47.2	39.2	24.8	47.2	39.2	42.4	38.4	36.0	41.6	17.9	37.4
16x0x0	59.2	45.6	51.2	48.8	52.0	52.8	52.0	49.6	48.8	47.2	50.7
	Architecture: (Inputs+Outputs)/2										
64x32x16	52.8	38.4	29.6	46.4	38.4	37.6	29.6	37.6	34.4	12.2	35.7
32x16x8	44.8	32.0	24.0	42.4	38.4	39.2	38.4	34.4	34.4	25.2	35.3
64x32x0	45.6	38.4	21.6	44.0	34.4	33.6	32.8	39.2	38.4	11.4	33.9
32x16x0	58.4	44.0	30.4	46.4	45.6	49.6	48.0	39.2	44.8	27.6	43.4
16x8x0	53.6	40.8	40.8	46.4	44.0	48.0	47.2	41.6	44.8	41.5	44.9
64x0x0	16.8	21.6	10.4	27.2	19.2	8.8	12.0	24.8	25.6	4.9	17.1
32x0x0	46.4	36.8	28.8	40.8	42.4	40.8	40.0	38.4	40.0	12.2	36.7
16x0x0	58.4	46.4	49.6	53.6	53.6	53.6	49.6	48.0	47.2	49.6	51.0
	Signed										
	Architecture: Inputs										
	64.8	49.6	53.6	54.4	57.6	51.2	52.8	53.6	52.0	48.0	53.8
	Architecture: Inputs/2										
	64.0	47.2	52.0	55.2	55.2	50.4	52.0	52.0	48.8	46.3	52.3
	Architecture: Outputs										
	63.2	48.0	52.8	52.8	57.6	53.6	50.4	53.6	52.0	47.2	53.1
	Architecture: Outputs/2										
	63.2	50.4	49.6	54.4	56.0	52.8	50.4	52.8	48.8	50.4	52.9
	Architecture: (Inputs+Outputs)/2										
	66.4	48.8	52.8	52.0	54.4	48.8	49.6	48.8	53.8	48.0	52.3
	Normalised										
	Architecture: Inputs										
	60.0	47.2	46.4	48.0	55.2	52.8	53.6	51.2	51.2	47.2	51.3
	Architecture: Inputs/2										
	61.6	44.8	50.4	49.6	59.2	52.0	53.6	53.6	47.2	51.2	52.3
	Architecture: Outputs										
	58.4	47.2	51.2	48.8	58.4	54.4	50.4	52.0	49.6	48.0	51.8
	Architecture: Outputs/2										
	59.2	47.2	47.2	52.0	52.0	52.8	49.6	52.0	51.2	48.8	51.2
	Architecture: (Inputs+Outputs)/2										
	60.0	49.6	52.0	49.6	57.6	54.4	53.6	55.2	52.8	48.8	53.4
Cell Size	**Gabor Filter (GF)-based Features**										
	Architecture: Inputs										
32x32	32.8	30.4	12.8	42.4	36.0	44.0	26.4	28.8	43.2	19.5	31.6
16x16	37.6	34.4	37.6	34.4	24.0	48.0	40.8	49.6	44.0	45.5	39.6
8x8	55.2	42.4	44.0	44.8	48.0	46.4	42.4	45.6	51.2	46.3	46.6
	Architecture: Inputs/2										
32x32	31.2	23.2	16.8	40.0	33.6	42.4	30.4	28.0	40.0	19.5	30.5
16x16	43.2	34.4	38.4	37.6	32.0	45.6	39.2	44.8	40.8	48.0	40.4
8x8	52.0	44.0	42.4	40.0	47.2	48.0	44.8	42.4	48.8	48.8	45.8

	Architecture: Outputs										
32x32	24.8	28.0	8.8	34.4	33.6	44.8	24.8	24.8	42.4	14.6	28.1
16x16	40.8	35.2	40.8	39.2	28.0	48.8	36.0	42.4	44.8	47.2	40.3
8x8	52.8	40.8	41.6	41.6	44.8	48.0	42.4	42.4	49.6	45.5	45.0
	Architecture: Outputs										
32x32	30.4	26.4	12.8	39.2	33.6	43.2	24.0	24.8	44.8	19.5	29.9
16x16	44.0	32.0	36.0	37.6	32.8	44.8	40.8	42.4	44.8	38.2	39.3
8x8	55.2	39.2	38.4	41.6	45.6	44.8	44.0	42.4	51.2	48.0	45.0
	Architecture: (Inputs+Outputs)/2										
32x32	21.6	26.4	11.2	42.4	35.2	43.2	22.4	28.0	43.2	15.4	28.9
16x16	40.0	36.0	39.2	32.8	26.4	46.4	38.4	44.0	45.6	44.7	39.4
8x8	54.4	41.6	38.4	44.0	45.6	48.0	42.4	44.8	52.8	48.8	46.1
Cell Size	**Discrete Cosine Transform (DCT) Coefficients**										
	Coefficients = 1										
	Architecture: Inputs										
64x64	11.2	5.6	12.8	13.6	8.8	12.8	10.4	14.4	14.4	9.8	11.4
32x32	24.8	16.8	16.0	24.0	17.6	22.4	16.8	20.8	16.0	11.4	18.7
16x16	48.0	35.2	32.8	40.0	40.0	40.0	39.2	33.6	40.0	38.2	38.7
8x8	54.4	46.4	42.4	44.8	50.4	51.2	47.2	41.6	47.2	48.8	47.4
	Architecture: Inputs/2										
32x32	15.2	8.8	14.4	14.4	15.2	16.0	19.2	10.4	16.8	7.3	13.8
16x16	40.0	25.6	28.0	32.0	28.0	32.8	27.2	31.2	30.4	29.3	30.4
8x8	56.0	48.8	42.4	48.8	50.4	47.2	48.0	46.4	48.8	52.8	49.0
	Architecture: Outputs										
64x64	9.6	9.6	14.4	15.2	15.2	14.4	9.6	12.8	13.6	11.4	12.6
32x32	31.2	18.4	21.6	16.0	24.8	28.8	23.2	14.4	25.6	18.7	22.3
16x16	48.8	36.0	42.4	41.6	48.0	40.8	42.4	43.2	46.4	40.7	43.0
8x8	55.2	46.4	44.0	47.2	52.0	49.6	49.6	43.2	49.6	48.0	48.5
	Architecture: Outputs/2										
64x64	10.4	9.6	14.4	16.8	12.0	14.4	9.6	12.8	15.2	8.1	12.3
32x32	29.6	19.2	20.8	16.8	24.8	28.0	23.2	13.6	20.8	18.7	21.5
16x16	50.4	33.6	41.6	43.2	45.6	41.6	40.8	42.4	48.0	39.8	42.7
8x8	54.4	44.8	40.8	48.8	54.4	48.8	49.6	44.0	46.4	50.4	48.2
	Architecture: (Inputs+Outputs)/2										
64x64	9.6	8.8	15.2	15.2	12.8	15.2	9.6	12.8	14.4	9.8	12.3
32x32	29.6	18.4	21.6	15.2	22.4	28.8	21.6	13.6	19.2	17.1	20.7
16x16	52.0	36.8	40.8	42.4	46.4	41.6	40.0	39.2	46.4	40.7	42.6
8x8	56.8	48.8	48.0	43.2	48.8	48.0	42.4	40.0	48.0	48.8	47.3
	Coefficients = 10										
	Architecture: Inputs										
64x64	10.4	20.0	19.2	35.2	28.8	30.4	24.0	24.8	23.2	13.8	23.0
32x32	52.0	33.6	33.6	40.0	40.8	44.8	30.4	18.4	40.0	35.8	36.9
16x16	57.6	44.0	45.6	46.4	56.0	48.8	44.8	43.2	50.4	51.2	48.8
8x8	55.2	46.4	48.0	47.2	50.4	50.4	46.4	44.8	48.0	50.4	48.7
	Architecture: Inputs/2										
64x64	12.8	16.0	20.0	25.6	25.6	21.6	19.2	23.2	24.8	10.6	19.9
32x32	47.2	33.6	38.4	40.0	37.6	45.6	34.4	20.8	44.0	33.3	37.5
16x16	55.2	40.0	44.0	47.2	54.4	52.0	44.8	45.6	48.8	52.0	48.4

8x8	58.4	43.2	44.0	48.0	49.6	52.0	48.8	42.4	46.4	46.3	47.9
	Architecture: Outputs										
64x64	12.8	8.0	3.2	18.4	18.4	25.6	4.0	3.2	15.2	4.9	11.4
32x32	52.0	35.2	30.4	38.4	39.2	47.2	27.2	20.8	34.4	30.9	35.6
16x16	54.4	45.6	44.0	45.6	55.2	53.6	43.2	41.6	51.2	48.0	48.2
8x8	56.0	43.2	45.6	48.0	54.4	51.2	50.4	40.0	47.2	48.8	48.5
	Architecture: Outputs/2										
64x64	9.6	7.2	5.6	16.0	24.0	36.0	4.0	5.6	19.2	7.3	13.5
32x32	52.0	33.6	33.6	40.0	40.8	44.8	30.4	18.4	40.0	35.8	36.9
16x16	54.4	44.8	42.4	47.2	51.2	50.4	43.2	40.0	48.0	48.8	47.0
8x8	56.8	44.8	44.8	48.8	52.8	50.4	44.8	41.6	48.0	47.2	48.0
	Architecture: (Inputs+Outputs)/2										
64x64	8.8	8.0	3.2	17.6	22.4	31.2	3.2	5.6	16.8	4.1	12.1
32x32	52.8	34.4	21.6	40.8	43.2	43.2	29.6	17.6	40.0	37.4	36.1
16x16	54.4	43.2	43.2	47.2	55.2	49.6	44.8	43.2	48.0	51.2	48.0
8x8	57.6	42.4	42.4	45.6	52.8	48.0	49.6	40.8	47.2	51.2	47.8
	Coefficients = 20										
	Architecture: Inputs										
64x64	22.4	28.0	21.6	42.4	40.0	43.2	30.4	24.0	38.4	25.2	31.6
32x32	47.2	36.0	32.0	39.2	40.0	46.4	36.8	14.4	39.2	43.1	37.4
16x16	56.0	40.8	42.4	45.6	53.6	46.4	42.4	45.6	48.8	47.2	46.9
	Architecture: Inputs/2										
64x64	24.0	19.2	24.0	36.0	27.2	28.8	20.8	27.2	27.2	21.1	25.6
32x32	48.8	37.6	31.2	44.8	39.2	44.0	35.2	29.6	42.4	43.1	39.6
16x16	55.2	42.4	45.6	45.6	50.4	49.6	47.2	46.4	51.2	46.3	48.0
	Architecture: Outputs										
64x64	14.4	13.6	12.8	41.6	39.2	40.0	10.4	12.8	37.6	16.3	23.9
32x32	48.8	37.6	45.6	40.0	40.8	48.8	37.6	28.8	39.2	39.0	40.6
16x16	53.6	43.2	47.2	46.4	52.0	49.6	40.8	42.4	52.8	50.4	47.8
	Architecture: Outputs/2										
64x64	14.4	30.4	24.0	39.2	40.0	42.4	27.2	22.4	36.0	24.4	30.0
32x32	48.8	37.6	31.2	44.8	39.2	44.0	35.2	29.6	42.4	43.1	39.6
16x16	55.2	40.8	44.0	44.0	51.2	43.2	42.4	42.4	48.8	46.3	45.8
	Architecture: (Inputs+Outputs)/2										
64x64	20.8	25.6	17.6	40.8	40.0	44.8	23.2	12.8	37.6	23.6	28.7
32x32	47.2	36.0	32.0	39.2	40.0	46.4	36.8	14.4	39.2	43.1	37.4
16x16	55.2	41.6	45.6	42.4	48.8	48.8	45.6	46.4	50.4	51.2	47.6
	Coefficients = 30										
	Architecture: Inputs										
64x64	20.8	36.0	20.8	44.0	44.8	40.8	34.4	28.0	38.4	28.5	33.6
32x32	48.8	32.0	41.6	38.4	39.2	48.0	37.6	15.2	41.6	41.5	38.4
16x16	55.2	44.0	41.6	46.4	48.0	48.8	45.6	47.2	45.6	48.0	47.0
	Architecture: Inputs/2										
64x64	20.8	25.6	25.6	40.0	36.8	35.2	24.0	24.8	32.0	27.6	29.2
32x32	51.2	37.6	36.8	36.0	43.2	47.2	35.2	22.4	43.2	37.4	39.0
16x16	56.0	38.4	44.8	44.0	53.6	48.0	46.4	44.8	49.6	49.6	47.5
	Architecture: Outputs										
64x64	16.8	26.4	16.8	40.0	42.4	42.4	31.2	24.0	30.4	24.4	29.5

32x32	49.6	35.2	41.6	37.6	41.6	46.4	34.4	24.8	44.0	41.5	39.7
16x16	55.2	44.0	43.2	45.6	47.2	48.0	46.4	40.8	53.6	47.2	47.1
	Architecture: Outputs/2										
64x64	19.2	28.0	14.4	43.2	44.0	42.4	23.2	31.2	38.4	21.1	30.5
32x32	49.6	35.2	36.0	37.6	41.6	47.2	39.2	24.8	42.4	39.8	39.3
16x16	56.8	40.8	44.8	43.2	48.0	46.4	42.4	44.0	46.4	44.7	45.8
	Architecture: (Inputs+Outputs)/2										
64x64	18.4	31.2	20.0	41.6	39.2	46.4	31.2	22.4	38.4	25.2	31.4
32x32	53.6	32.8	35.2	40.8	38.4	47.2	34.4	15.2	43.2	37.4	37.8
16x16	53.6	40.0	40.0	44.8	50.4	50.4	43.2	40.0	48.0	51.2	46.2
	Coefficients = 40										
	Architecture: Inputs										
64x64	24.8	29.6	28.0	46.4	41.6	41.6	32.0	27.2	44.0	31.7	34.7
32x32	50.4	31.2	35.2	35.2	39.2	45.6	32.0	25.6	42.4	38.2	37.5
16x16	53.6	39.2	41.6	42.4	50.4	48.8	44.8	44.0	49.6	49.6	46.4
	Architecture: Inputs/2										
64x64	24.0	27.2	30.4	39.2	40.8	44.8	32.0	27.2	35.2	25.2	32.6
32x32	51.2	36.8	39.2	36.8	40.0	47.2	36.8	24.0	41.6	39.8	39.3
16x16	53.6	40.0	39.2	42.4	44.8	46.4	44.8	43.2	46.4	46.3	44.7
	Architecture: Outputs										
64x64	20.0	28.8	21.6	46.4	43.2	42.4	33.6	27.2	40.8	24.4	32.8
32x32	48.8	35.2	34.4	36.0	46.4	47.2	39.2	17.6	40.0	39.0	38.4
16x16	53.6	38.4	41.6	43.2	49.6	46.4	40.8	40.0	46.4	47.2	44.7
	Architecture: Outputs/2										
64x64	24.8	29.6	28.0	46.4	41.6	41.6	32.0	27.2	44.0	31.7	34.7
32x32	47.2	36.8	34.4	40.0	42.4	45.6	34.4	24.0	40.0	37.4	38.2
16x16	52.8	39.2	43.2	41.6	45.6	42.4	42.4	40.8	47.2	39.8	43.5
	Architecture: (Inputs+Outputs)/2										
64x64	23.2	24.0	27.2	41.6	44.8	42.4	31.2	24.8	36.0	29.3	32.4
32x32	41.6	34.4	33.6	39.2	42.4	46.4	33.6	18.4	40.0	39.0	36.9
16x16	51.2	42.4	38.4	43.2	47.2	51.2	43.2	40.8	52.0	45.5	45.5
	Coefficients = 50										
	Architecture: Inputs										
64x64	27.2	33.6	24.8	43.2	47.2	44.0	34.4	36.8	38.4	26.0	35.6
32x32	48.0	33.6	32.8	38.4	42.4	47.2	30.4	16.8	43.2	33.3	36.6
16x16	51.2	39.2	36.8	43.2	50.4	48.0	39.2	39.2	47.2	47.2	44.2
	Architecture: Inputs/2										
64x64	25.6	35.2	26.4	44.0	40.0	44.8	35.2	31.2	36.8	28.5	34.8
32x32	52.0	35.2	40.8	39.2	47.2	44.0	36.0	27.2	44.0	39.8	40.5
16x16	55.2	39.2	37.6	44.0	48.8	49.6	39.2	43.2	44.8	47.2	44.9
	Architecture: Outputs										
64x64	21.6	28.0	28.0	41.6	46.4	44.8	38.4	28.8	43.2	26.8	34.8
32x32	48.0	35.2	30.4	39.2	41.6	45.6	29.6	29.6	43.2	39.8	38.2
16x16	53.6	39.2	39.2	42.4	49.6	49.6	43.2	44.0	46.4	44.7	45.2
	Architecture: Outputs/2										
64x64	22.4	36.0	28.0	48.0	42.4	44.0	30.4	35.2	44.8	31.7	36.3
32x32	48.8	36.8	32.0	37.6	42.4	47.2	40.0	21.6	43.2	35.0	38.5
16x16	52.8	41.6	37.6	40.0	46.4	48.8	39.2	42.4	45.6	38.2	43.3

	Architecture: (Inputs+Outputs)/2										
64x64	21.6	34.4	29.6	42.4	45.6	44.8	38.4	25.6	37.6	26.0	34.6
32x32	49.6	34.4	37.6	38.4	37.6	46.4	33.6	18.4	43.2	39.8	37.9
16x16	50.4	37.6	40.0	45.6	47.2	47.2	44.0	41.6	46.4	46.3	44.6
Features	**Hybrid Features**										
	Architecture: Inputs										
HoGs,DCT,UBs	55.2	46.4	47.2	48.0	52.0	51.2	46.4	40.8	49.6	52.8	49.0
HoGs,DCT	56.8	47.2	45.6	46.4	52.0	51.2	48.0	42.4	48.8	49.6	48.8
HoGs,UBs	63.2	49.6	52.8	55.2	58.4	51.2	49.6	52.0	54.4	49.6	53.6
DCT,UBs	55.2	47.2	41.6	46.4	48.8	48.8	41.6	41.6	51.2	47.2	47.0
	Architecture: Inputs/2										
HoGs,DCT,UBs	54.4	46.4	40.8	48.0	52.8	48.0	46.4	44.8	52.0	48.8	48.2
HoGs,DCT	57.6	50.4	42.4	49.6	57.6	49.6	46.4	44.8	46.4	46.3	49.1
HoGs,UBs	61.6	47.2	51.2	55.0	54.4	50.4	49.6	48.8	56.0	51.2	52.5
DCT,UBs	58.4	48.0	43.2	47.2	55.2	49.6	46.4	43.2	47.2	51.2	49.0
	Architecture: Outputs										
HoGs,DCT,UBs	53.6	45.6	39.2	48.8	52.0	52.0	51.2	41.6	51.2	49.6	48.5
HoGs,DCT	56.8	46.4	44.0	47.2	53.6	48.8	48.0	41.6	52.0	47.2	48.6
HoGs,UBs	63.2	47.2	49.6	54.4	60.0	49.6	54.4	52.8	52.8	48.0	53.2
DCT,UBs	56.0	50.4	41.6	48.8	52.8	48.8	48.8	40.8	47.2	48.0	48.3
	Architecture: Outputs										
HoGs,DCT,UBs	55.2	42.4	40.8	48.8	50.4	49.6	50.4	42.4	51.2	46.3	47.8
HoGs,DCT	59.2	45.6	41.6	46.4	53.6	46.4	44.8	43.2	48.0	50.4	47.9
HoGs,UBs	58.4	50.4	48.8	53.6	56.8	52.8	51.2	50.4	52.8	49.6	52.5
DCT,UBs	52.8	47.2	45.6	48.8	51.2	51.2	46.4	44.0	47.2	48.0	48.2
	Architecture: (Inputs+Outputs)/2										
HoGs,DCT,UBs	55.2	49.6	45.6	48.8	51.2	48.0	49.6	44.8	49.6	47.2	49.0
HoGs,DCT	55.2	44.0	47.2	48.0	53.6	47.2	44.0	40.0	46.4	50.4	47.6
HoGs,UBs	61.6	48.0	52.0	52.8	56.8	54.4	52.8	53.6	56.0	46.3	53.4
DCT,UBs	55.2	47.2	44.8	47.2	52.8	51.2	49.6	44.8	48.8	48.8	49.0
	Synthetic Data										
	Architecture: Inputs										
	66.4	56.0	52.0	57.6	64.0	56.8	52.0	59.2	56.8	55.3	57.6
	Architecture: Inputs/2										
	63.2	50.4	53.6	56.8	62.4	51.2	54.4	55.2	56.8	53.7	55.8
	Architecture: Outputs										
	68.0	53.6	56.8	56.8	60.0	56.0	52.8	56.0	56.0	56.1	57.2
	Architecture: Outputs/2										
	62.4	51.2	52.0	55.2	59.2	51.2	51.2	51.2	56.8	53.7	54.4
	Architecture: (Inputs+Outputs)/2										
	67.2	54.4	51.2	60.0	61.6	52.0	53.6	53.6	58.4	52.8	56.5
Samples	**Number of Samples**										
	Minimum Samples										
	Architecture: Inputs										
2	33.9	39.2	47.4	35.1	43.9	36.3	40.4	37.4	38.6	38.8	39.1
3	43.7	45.0	49.7	42.4	51.0	36.4	46.4	51.0	48.3	42.2	45.6
4	52.2	52.2	50.7	49.3	47.1	51.5	50.0	49.3	51.5	49.2	50.3
5	57.6	45.6	52.8	51.2	55.2	49.6	53.6	47.2	48.0	51.2	51.2

6	58.0	55.5	63.9	60.5	53.8	54.6	53.8	52.1	52.1	56.4	56.1
7	56.6	64.6	61.9	51.3	61.1	52.2	53.1	56.6	57.5	54.3	56.9
8	57.8	53.2	63.3	62.4	61.5	56.0	60.6	60.6	53.2	61.6	59.0
9	62.1	54.4	61.2	59.2	64.1	63.1	66.0	62.1	57.3	55.7	60.5
10	66.3	64.3	58.2	65.3	60.2	65.3	61.2	67.3	61.2	61.4	63.1
	Architecture: Inputs/2										
2	35.1	38.0	46.8	35.1	43.9	37.4	45.0	40.4	40.4	38.2	40.0
3	39.7	43.0	47.0	43.7	51.7	41.1	47.7	50.3	49.0	42.2	45.5
4	51.5	50.7	51.5	47.8	45.6	51.5	49.3	49.3	53.7	49.2	50.0
5	61.6	47.2	52.8	51.2	54.4	52.8	52.8	45.6	48.8	50.4	51.8
6	55.5	60.5	63.0	60.5	54.6	56.3	56.3	52.9	52.9	57.3	57.0
7	56.6	63.7	64.6	55.8	64.6	49.6	51.3	53.1	60.2	55.2	57.5
8	56.9	57.8	66.1	63.3	64.2	56.0	56.9	63.3	58.7	64.6	60.8
9	64.1	56.3	58.3	59.2	68.0	59.2	68.0	64.1	50.5	57.7	60.5
10	67.3	65.3	61.2	68.4	56.1	63.3	60.2	67.3	59.2	58.0	62.6
	Architecture: Outputs										
2	34.5	40.4	44.4	35.7	45.6	38.0	45.6	40.9	40.9	39.4	40.5
3	45.7	45.0	47.0	45.0	46.4	37.1	45.7	53.6	49.0	40.1	45.5
4	51.5	50.0	50.7	44.1	48.5	54.4	52.2	51.5	55.9	50.8	51.0
5	58.4	47.2	49.6	49.6	53.6	53.6	52.0	48.8	47.2	50.4	51.0
6	58.0	56.3	63.0	58.0	52.1	52.9	57.1	52.1	53.8	53.0	55.6
7	53.1	62.8	61.1	54.9	63.7	54.0	54.9	57.5	59.3	54.3	57.6
8	56.0	53.2	62.4	61.5	62.4	54.1	58.7	60.6	56.9	66.7	59.2
9	61.2	54.4	67.0	60.2	70.9	62.1	66.0	63.1	52.4	59.8	61.7
10	64.3	62.2	63.3	64.3	62.2	60.2	64.3	65.3	59.2	60.2	62.6
	Architecture: Outputs/2										
2	38.0	41.5	44.4	35.7	39.2	38.6	41.5	39.8	36.8	41.8	39.7
3	42.4	45.0	45.7	39.7	51.7	39.1	45.7	51.0	47.0	43.5	45.1
4	50.7	50.0	50.7	48.5	47.1	50.0	50.7	54.4	52.2	49.2	50.4
5	59.2	45.6	51.2	48.8	52.0	52.8	52.0	49.6	48.8	47.2	50.7
6	58.0	56.3	64.7	61.3	52.9	53.8	55.5	52.9	54.6	53.0	56.3
7	50.4	62.8	59.3	53.1	61.9	52.2	51.3	54.9	53.1	52.4	55.1
8	56.0	53.2	61.5	60.6	59.6	51.4	57.8	60.6	56.0	62.6	57.9
9	58.3	51.5	67.0	61.2	67.0	59.2	65.0	59.2	55.3	50.5	59.4
10	65.3	61.2	61.2	61.2	56.1	59.2	57.1	65.3	59.2	59.1	60.5
	Architecture: (Inputs+Outputs)/2										
2	32.2	39.8	46.8	31.6	43.3	37.4	40.9	39.2	38.0	38.2	38.7
3	41.7	43.7	49.0	42.4	50.3	37.7	45.0	51.0	46.4	40.1	44.7
4	51.5	49.3	50.7	48.5	45.6	55.9	51.5	52.2	50.0	47.7	50.3
5	58.4	46.4	49.6	53.6	53.6	53.6	49.6	48.0	47.2	49.6	51.0
6	56.3	58.0	61.3	58.8	56.3	53.8	54.6	51.3	56.3	54.7	56.1
7	58.4	61.9	61.9	54.0	66.4	54.0	51.3	52.2	61.9	56.2	57.8
8	59.6	52.3	61.5	65.1	59.6	56.9	59.6	60.6	56.9	61.6	59.4
9	66.0	56.3	65.0	63.1	68.9	59.2	68.0	63.1	55.3	57.7	62.3
10	66.3	66.3	61.2	64.3	57.1	59.2	61.2	65.3	59.2	64.8	62.5
	Fixed Samples										
	Architecture: Inputs										
2	0.0	12.5	12.5	12.5	0.0	0.0	25.0	0.0	25.0	16.7	10.4

3	16.7	41.7	33.3	41.7	25.0	16.7	8.3	8.3	25.0	22.2	23.9
4	25.0	18.8	6.3	37.5	43.8	18.8	37.5	25.0	43.8	41.7	29.8
5	30.0	20.0	40.0	35.0	30.0	20.0	20.0	30.0	20.0	40.0	28.5
6	37.5	50.0	25.0	41.7	50.0	41.7	37.5	45.8	37.5	50.0	41.7
7	42.9	60.7	28.6	46.4	35.7	46.4	39.3	46.4	46.4	47.6	44.0
8	53.1	37.5	46.9	40.6	59.4	56.3	28.1	53.1	46.9	50.0	47.2
9	47.2	38.9	63.9	36.1	36.1	50.0	38.9	58.3	55.6	40.7	46.6
10	47.5	30.0	40.0	55.0	45.0	30.0	62.5	45.0	52.5	43.3	45.1
Architecture: Inputs/2											
2	12.5	0.0	12.5	12.5	12.5	25.0	12.5	0.0	12.5	16.7	11.7
3	16.7	33.3	33.3	41.7	8.3	16.7	8.3	8.3	25.0	44.4	23.6
4	37.5	18.8	12.5	25.0	43.8	25.0	25.0	31.3	37.5	41.7	29.8
5	25.0	25.0	30.0	45.0	30.0	20.0	25.0	40.0	30.0	46.7	31.7
6	33.3	54.2	33.3	37.5	41.7	33.3	33.3	41.7	37.5	38.9	38.5
7	42.9	50.0	39.3	39.3	53.6	42.9	39.3	35.7	53.6	52.4	44.9
8	46.9	31.3	40.6	46.9	59.4	56.3	25.0	43.8	43.8	45.8	44.0
9	52.8	41.7	58.3	30.6	36.1	50.0	41.7	66.7	47.2	44.4	46.9
10	50.0	40.0	47.5	47.5	50.0	35.0	62.5	45.0	57.5	53.3	48.8
Architecture: Outputs											
2	12.5	25.0	12.5	12.5	0.0	25.0	37.5	0.0	25.0	16.7	16.7
3	16.7	33.3	50.0	50.0	16.7	33.3	8.3	33.3	50.0	22.2	31.4
4	37.5	31.3	12.5	43.8	31.3	25.0	25.0	25.0	37.5	50.0	31.9
5	35.0	30.0	30.0	40.0	40.0	15.0	30.0	35.0	15.0	33.3	30.3
6	29.2	45.8	41.7	37.5	45.8	45.8	41.7	45.8	33.3	50.0	41.7
7	42.9	46.4	39.3	39.3	39.3	53.6	39.3	39.3	42.9	57.1	43.9
8	40.6	53.1	53.1	46.9	59.4	53.1	31.3	53.1	53.1	45.8	49.0
9	50.0	36.1	61.1	30.6	38.9	47.2	44.4	58.3	44.4	40.7	45.2
10	47.5	35.0	42.5	50.0	35.0	30.0	62.5	47.5	57.5	50.0	45.8
Architecture: Outputs/2											
2	12.5	12.5	0.0	12.5	12.5	12.5	12.5	0.0	12.5	16.7	10.4
3	16.7	33.3	41.7	41.7	25.0	16.7	8.3	16.7	33.3	33.3	26.7
4	31.3	25.0	12.5	37.5	37.5	6.3	43.8	12.5	37.5	25.0	26.9
5	40.0	25.0	35.0	35.0	20.0	15.0	30.0	30.0	25.0	26.7	28.2
6	29.2	45.8	29.2	29.2	41.7	33.3	41.7	33.3	33.3	44.4	36.1
7	46.4	60.7	39.3	32.1	35.7	39.3	46.4	35.7	39.3	57.1	43.2
8	40.6	31.3	40.6	43.8	65.6	50.0	31.3	56.3	50.0	33.3	44.3
9	52.8	33.3	55.6	38.9	33.3	52.8	44.4	52.8	50.0	37.0	45.1
10	47.5	37.5	45.0	52.5	45.0	32.5	60.0	55.0	55.0	46.7	47.7
Architecture: (Inputs+Outputs)/2											
2	12.5	12.5	12.5	12.5	12.5	25.0	12.5	0.0	12.5	16.7	12.9
3	16.7	41.7	25.0	25.0	25.0	16.7	0.0	8.3	41.7	33.3	23.3
4	31.3	18.8	18.8	37.5	37.5	18.8	18.8	18.8	43.8	41.7	28.5
5	25.0	20.0	35.0	45.0	45.0	35.0	25.0	45.0	30.0	33.3	33.8
6	37.5	54.2	33.3	33.3	45.8	29.2	50.0	45.8	33.3	38.9	40.1
7	35.7	42.9	39.3	35.7	50.0	42.9	39.3	35.7	57.1	52.4	43.1
8	37.5	40.6	43.8	43.8	56.3	46.9	21.9	53.1	43.8	58.3	44.6
9	47.2	44.4	58.3	30.6	33.3	50.0	41.7	52.8	47.2	48.1	45.4
10	52.5	42.5	42.5	55.0	42.5	42.5	60.0	47.5	50.0	40.0	47.5

	Multiple Authors										
	Architecture: Inputs										
	48.4	59.0	51.6	51.6	49.7	55.9	52.8	50.9	51.6	51.6	52.3
	Architecture: Inputs/2										
	49.1	57.8	50.9	52.2	54.0	57.1	52.8	49.1	50.3	51.6	52.5
	Architecture: Outputs										
	47.2	59.6	51.6	51.6	52.2	56.5	50.3	49.7	52.2	49.7	52.0
	Architecture: Outputs/2										
	50.9	59.0	49.7	49.1	52.8	54.0	54.7	49.1	50.3	49.7	51.9
	Architecture: (Inputs+Outputs)/2										
	46.6	60.9	49.7	52.2	55.9	54.0	54.0	51.6	54.0	51.0	53.0

Appendix E

Results of Experiments Using Hidden Markov Models for Word Recognition

	Fold										
	1	2	3	4	5	6	7	8	9	10	Avg.
Cell Size	**Undersampled Bitmaps (UBs)**										
32x1	43.2	27.2	24.0	36.0	28.0	30.4	28.8	24.0	33.6	28.5	30.4
16x16	40.0	27.2	30.4	34.4	36.0	35.2	28.0	32.0	28.8	29.3	32.1
8x1	44.8	29.6	26.4	37.6	32.0	37.6	30.4	39.2	36.0	36.6	35.0
4x1	44.0	27.2	29.6	38.4	32.8	33.6	27.2	33.6	34.4	29.3	33.0
	Marti & Bunke (M&B) Features										
	60.0	39.2	33.6	48.8	45.6	45.6	40.0	40.8	48.8	42.3	44.5
Cell Size	**Geometric Moments (GMs)**										
	Moments = M1										
64x1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32x1	31.2	17.6	20.8	26.4	28.0	22.4	20.8	24.8	23.2	22.8	23.8
16x16	33.6	20.0	20.8	33.6	23.2	28.0	27.2	19.2	26.4	28.5	26.0
8x1	36.0	18.4	18.4	33.6	25.6	33.6	22.4	25.6	30.4	27.6	27.2
	Moments = M1,M2										
64x1	3.2	4.8	4.8	6.4	4.8	4.8	5.6	4.0	5.6	2.4	4.6
32x1	27.2	20.8	23.2	20.0	23.2	20.8	20.0	21.6	24.0	23.6	22.4
16x16	27.2	23.2	20.8	28.8	25.6	28.0	23.2	20.0	24.0	26.8	24.8
8x1	28.0	20.0	19.2	31.2	24.8	24.8	21.6	24.0	28.0	26.0	24.8
	Moments = M1,M2,M3										
64x1	2.4	4.0	4.0	4.8	5.6	5.6	4.8	2.4	1.6	2.4	3.8
32x1	24.0	22.4	23.2	20.8	26.4	21.6	19.2	16.8	24.8	23.6	22.3
16x16	27.2	22.4	20.8	24.0	20.8	27.2	23.2	20.0	18.4	21.1	22.5
8x1	27.2	19.2	19.2	28.0	24.0	28.0	23.2	20.8	29.6	21.1	24.0
	Moments = M1,M2,M3,M4										
64x1	4.0	4.0	3.2	5.6	6.4	5.6	1.6	2.4	4.0	2.4	3.9
32x1	22.4	20.8	22.4	22.4	23.2	22.4	19.2	18.4	20.8	19.5	21.2
16x16	21.6	20.0	20.8	28.0	21.6	22.4	21.6	18.4	21.6	26.0	22.2

8x1	25.6	19.2	19.2	25.6	19.2	28.8	23.2	20.0	28.0	23.6	23.2
Cell Size	**Gabor Filter (GF)-based Features**										
64x4	44.8	30.4	43.2	36.8	38.4	41.6	29.6	31.2	40.0	36.6	37.3
32x4	40.0	32.0	34.4	40.8	37.6	40.0	31.2	33.6	36.8	40.7	36.7
16x4	47.2	30.4	30.4	39.2	40.8	37.6	34.4	36.0	36.8	35.0	36.8
8x4	45.6	29.6	30.4	36.8	37.6	34.4	32.0	31.2	35.2	39.0	35.2
Cell Size	**Discrete Cosine Transform (DCT) Coefficients**										
	Coefficients = 1										
64x1	1.6	0.8	3.2	1.6	1.6	1.6	3.2	1.6	2.4	4.1	2.2
32x1	53.6	34.4	45.6	44.8	50.4	39.2	42.4	39.2	40.8	43.1	43.3
16x16	44.8	31.2	40.0	40.8	45.6	36.8	40.8	39.2	36.0	39.8	39.5
8x1	45.6	34.4	36.8	38.4	40.0	41.6	35.2	40.8	37.6	38.2	38.9
	Coefficients = 10										
64x1	48.8	33.6	29.6	45.6	41.6	37.6	30.4	39.2	35.2	43.1	38.5
32x1	47.2	28.0	28.8	38.4	35.2	36.0	27.2	39.2	34.4	31.7	34.6
16x16	40.8	28.8	26.4	36.8	33.6	32.8	25.6	29.6	30.4	29.3	31.4
	Coefficients = 20										
64x1	44.0	27.2	28.8	39.2	29.6	40.0	28.0	34.4	33.6	32.5	33.7
32x1	41.6	23.2	26.4	34.4	28.0	36.0	25.6	33.6	33.6	25.2	30.8
	Coefficients = 30										
64x1	37.6	25.6	24.8	38.4	28.8	34.4	25.6	28.8	32.0	33.3	30.9
32x1	37.6	23.2	24.8	32.8	28.0	32.8	24.0	33.6	32.0	26.8	29.6
	Coefficients = 40										
64x1	34.4	24.8	22.4	32.0	29.6	33.6	23.2	28.0	32.0	26.8	28.7
	Coefficients = 50										
64x1	32.8	20.8	23.2	32.8	23.2	27.2	23.2	28.8	28.8	22.8	26.4
Features	**Hybrid Features**										
M&B,GF,DCT	47.2	33.6	36.0	44.8	46.4	41.6	36.0	38.4	40.0	38.2	40.2
M&B,GF	45.6	33.6	33.6	43.2	41.6	39.2	38.4	43.2	42.4	42.3	40.3
M&B,DCT	58.4	37.6	34.4	48.8	48.0	48.0	43.2	44.8	43.2	41.5	44.8
GF,DCT	45.6	34.4	33.6	34.4	40.0	39.2	36.0	40.0	36.8	43.9	38.4
	Synthetic Data										
	58.4	48.8	44.0	46.4	58.4	52.8	53.6	50.4	47.2	56.1	51.6
Samples	**Number of Samples**										
	Minimum Samples										
2	28.7	33.3	33.9	29.8	31.6	33.9	36.8	33.9	32.2	32.1	32.6
3	37.1	35.1	36.4	37.1	32.5	30.5	39.7	39.1	43.1	34.7	36.5
4	39.0	41.2	41.9	40.4	37.5	44.9	41.9	49.3	39.0	42.4	41.7
5	58.4	37.6	34.4	48.8	48.0	48.0	43.2	44.8	43.2	41.5	44.8
6	48.7	47.1	45.4	43.7	51.3	48.7	41.2	45.4	49.6	47.0	46.8
7	49.6	53.1	52.2	45.1	58.4	50.4	51.3	53.1	47.8	38.1	49.9
8	45.9	56.9	56.0	48.6	57.8	43.1	53.2	49.5	55.1	55.6	52.2
9	58.3	50.5	53.4	59.2	64.1	49.5	61.2	52.4	45.6	48.5	54.3
10	56.1	59.2	57.1	57.1	55.1	57.1	56.1	60.2	54.1	54.6	56.7
	Fixed Samples										
2	0.0	0.0	0.0	12.5	0.0	0.0	0.0	12.5	0.0	0.0	2.5
3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	8.3	11.1	8.6

4	18.8	12.5	18.8	25.0	12.5	12.5	18.8	12.5	12.5	25.0	16.9
5	35.0	5.0	15.0	10.0	15.0	5.0	20.0	25.0	35.0	20.0	18.5
6	29.2	20.8	16.7	20.8	20.8	37.5	29.2	25.0	29.2	27.8	25.7
7	46.4	35.7	14.3	28.6	28.6	25.0	25.0	32.1	39.3	28.6	30.4
8	21.9	40.6	40.6	25.0	43.8	40.6	18.8	34.4	43.8	33.3	34.3
9	61.1	22.2	47.2	38.9	36.1	27.8	25.0	44.4	44.4	22.2	36.9
10	37.5	37.5	37.5	37.5	40.0	37.5	52.5	30.0	47.5	46.7	40.4
	Multiple Authors										
	46.6	44.7	37.9	46.6	46.6	45.3	43.5	41.0	48.5	40.7	44.1

Appendix F

Results of Experiments Using Hidden Markov Models for Text Line Recognition

	Fold										
	1	2	3	4	5	6	7	8	9	10	Avg.
Cell Size	**Undersampled Bitmaps (UBs)**										
32x1	3.6	-5.3	2.2	1.8	-4.1	-3.0	-5.2	3.9	5.6	-4.9	-0.5
16x16	20.1	18.7	18.6	18.3	21.4	19.5	21.2	19.7	20.4	20.6	19.9
8x1	21.8	19.8	24.0	22.1	23.4	22.3	23.2	21.2	23.2	21.6	22.2
4x1	21.7	22.8	22.9	21.1	23.8	20.7	23.4	22.2	23.4	21.9	22.4
	Marti & Bunke (M&B) Features										
	35.7	34.1	35.9	35.6	35.8	36.6	36.4	35.8	35.8	35.4	35.7
Cell Size	**Geometric Moments (GMs)**										
	Moments = M1										
64x1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32x1	-25.0	-27.2	-21.1	-26.8	-23.5	-26.4	-21.4	-21.8	-24.0	-29.4	-24.7
16x16	10.5	9.0	13.4	8.9	15.8	9.9	11.8	11.2	10.9	10.5	11.2
8x1	20.3	17.2	20.0	17.9	19.3	18.6	19.0	17.5	19.0	19.3	18.8
	Moments = M1,M2										
32x1	-18.5	-23.9	-15.5	-25.4	-18.4	-23.1	-17.5	-19.3	-18.6	-26.9	-20.7
16x16	11.1	9.9	11.4	9.9	13.1	10.3	10.2	11.3	10.3	9.5	10.7
8x1	17.2	15.2	17.9	15.7	19.7	17.5	17.3	18.2	18.1	17.0	17.4
	Moments = M1,M2,M3										
64x1	-54.6	-55.8	-49.2	-56.0	-48.2	-52.0	-52.5	-52.2	-50.8	-53.4	-52.5
32x1	-15.9	-21.2	-17.2	-24.3	-18.7	-18.1	-16.6	-17.6	-19.5	-19.9	-18.9
16x16	10.6	10.0	13.5	8.9	13.9	7.5	11.4	12.5	11.6	10.3	11.0
8x1	18.1	15.6	18.5	15.9	18.8	16.4	18.0	17.5	18.0	17.1	17.4
	Moments = M1,M2,M3,M4										
64x1	-53.0	-52.4	-49.2	-55.0	-46.5	-51.5	-52.1	-53.4	-48.9	-53.2	-51.5
32x1	-15.5	-22.9	-17.5	-22.0	-18.9	-22.3	-16.6	-22.9	-21.0	-24.2	-20.4
16x16	9.1	7.0	11.3	8.2	13.5	8.8	10.8	11.2	12.2	9.6	10.2
8x1	18.0	15.0	16.9	18.3	18.4	16.6	17.6	16.5	17.4	15.7	17.0

Cell Size	**Gabor Filter (GF)-based Features**										
32x4	11.7	12.4	12.2	11.1	14.1	12.4	13.1	12.2	15.2	15.0	12.9
16x4	22.0	21.3	23.4	23.0	23.8	22.6	23.6	22.4	23.0	22.8	22.8
Cell Size	**Discrete Cosine Transform (DCT) Coefficients**										
	Coefficients = 1										
32x1	11.7	9.1	11.0	7.9	11.0	10.6	12.1	9.5	10.1	11.9	10.5
16x16	21.3	19.4	21.8	19.4	23.4	20.4	21.1	21.0	20.3	20.3	20.8
8x1	27.0	24.9	26.4	23.5	27.2	24.8	27.8	24.9	24.4	26.5	25.7
	Coefficients = 10										
64x1	28.1	25.6	28.6	26.2	29.5	29.4	28.0	26.5	28.2	26.7	27.7
32x1	24.6	25.8	28.1	24.9	27.2	26.0	25.1	25.2	26.3	25.3	25.8
	Coefficients = 20										
64x1	25.3	23.8	27.1	22.6	26.0	27.0	25.7	22.0	25.3	23.4	24.8
	Coefficients = 30										
64x1	22.6	19.8	24.8	24.7	26.1	24.8	22.7	21.9	23.4	21.9	23.3
	Coefficients = 40										
64x1		18.9	22.2	20.4	22.8	22.6	26.7	23.6	23.6	22.7	22.6
Features	**Hybrid Features**										
M&B,DCT, UBs	31.7	29.2	32.4	30.6	33.6	32.8	32.1	31.7	30.7	29.6	31.4
M&B,DCT	35.2	33.3	34.2	32.9	35.7	35.9	34.3	34.6	34.0	33.5	34.4
M&B,UBs	33.5	31.3	34.9	33.3	35.2	34.3	33.6	32.4	31.9	31.5	33.2
DCT,UBs	30.1	28.2	30.8	30.2	31.2	31.3	30.8	29.1	30.3	29.4	30.1
	Synthetic Data										
	33.4	32.0	33.0	31.3	35.6	34.8	35.7	33.4	35.2	32.4	33.7
	Number of Samples										
Samples	Minimum Samples										
2	38.4	36.9	35.3	38.6	38.9	35.1	35.9	36.7	36.4	38.3	37.1
3	38.4	36.8	36.9	36.9	39.2	35.3	36.2	35.5	36.7	36.3	36.8
4	39.9	37.1	38.1	38.0	39.3	37.5	35.7	36.7	36.9	35.9	37.5
5	38.6	39.9	38.8	35.7	38.9	35.0	37.4	34.1	37.6	34.7	37.1
6	39.1	38.3	35.9	36.0	39.5	35.9	35.5	35.0	35.8	36.2	36.7
7	37.4	38.5	36.0	36.6	38.4	36.4	36.8	35.3	34.2	35.9	36.6
8	42.2	38.3	35.7	36.8	40.5	33.6	36.6	36.0	31.2	37.7	36.9
9	39.8	39.4	36.1	36.3	41.2	36.2	37.0	37.5	34.2	36.0	37.4
10	36.8	40.0	37.2	35.5	41.6	33.9	35.6	35.6	36.5	36.4	36.9
Size	Size of Sub-Corpus										
10%	34.3	28.8	28.3	29.7	30.9	29.3	33.1	24.9	35.1	27.9	30.2
20%	31.5	29.7	33.0	31.0	36.9	32.1	30.4	31.4	32.8	29.4	31.8
30%	35.2	27.2	32.8	34.6	32.4	33.3	33.7	31.1	34.0	32.6	32.7
40%	33.6	34.3	33.5	33.0	30.6	35.3	31.3	28.4	36.2	34.9	33.1
50%	33.6	32.9	35.3	35.5	36.1	34.0	34.6	31.3	34.9	32.8	34.1
60%	34.3	34.4	35.3	33.1	38.4	36.6	36.5	34.4	34.0	34.5	35.1
70%	34.8	34.2	35.4	34.9	37.3	32.9	33.8	34.4	36.7	34.1	34.8
80%	34.3	35.3	36.4	32.8	36.4	34.8	36.3	35.4	37.0	35.3	35.4
90%	35.6	35.0	37.2	34.6	35.9	36.2	35.6	34.9	35.8	35.9	35.7
100%	35.7	34.1	35.9	35.6	35.8	36.6	36.4	35.8	35.8	35.4	35.7

	Multiple Authors										
	24.8	29.9	23.3	27.8	29.9	34.0	31.8	35.1	32.4	31.3	30.0