CHARACTER RECOGNITION USING FOURIER DESCRIPTORS

by

Jared A. Hopkins

A project

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

April 2006

The project presented by *Jared A. Hopkins* entitled *Character Recognition Using Fourier Descriptors* is hereby approved.

Tim Andersen, Advisor	Date
Elisa Barney Smith, Committee Member	Date
Amit Jain, Committee Member	Date
John R. Pelton, Graduate Dean	Date

ACKNOWLEDGEMENTS

I would like to thank the members of my committee for their excellent help. I would also like to thank my family for their support.

This material is based upon work supported by the National Science Foundation under Grant No. 0321233.

ABSTRACT

A set of features for performing optical character recognition of bi-tonal images is implemented within the Gamera framework. The features are based on the Fourier Descriptor but include a method which allows classification of images which contain multiple boundaries. This is accomplished by assigning to each character image a signature which encodes the boundary types that are present in the image as well as the positional relationships that exist between them. This allows a boundary-wise comparison of different images to be accomplished using their Fourier Descriptors. Under this approach, only images having the same signature are comparable. This fact in turn affects the architecture of exemplar-based classifiers which use these features. Effectively, a meta-classifier is used which first computes the signature of an input image and dispatches the image to a classifier which is trained to recognize images having that signature. The implementation is carried out by extending the functionality of Gamera, an existing open-source framework for building document analysis applications. The features are then tested by implementing a fuzzy-knn and a neural-network classifier based on them. The fuzzy-knn classifier achieves an estimated generalization accuracy of 94%, which is the best rate yet achieved on the particular data set used. The neural-network classifier achieves an estimated generalization accuracy of 91%.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Optical Character Recognition (OCR) is an important application of pattern classification. There are many documents of historical, technical, and economic importance which exist only in printed form. OCR can help to reduce the cost of digitizing these documents. There exist many successful OCR techniques which have been applied to areas such as handwriting recognition, recognition of mechanically printed text, and recognition of musical notes. Such applications have been carried out in the context of bi-tonal as well as gray-scale images. These systems employ a large number of different image features and many of them make use of multiple features in order to obtain good performance. Descriptions of features commonly used for OCR can be found in [1], [2], [10], [14], [15], [19], and [20]. When evaluating the usefulness of a feature set it is important to look not only at the accuracy of a classifier based on it, but also at the correlation between errors made by the classifier and those made by a classifier which uses an independent set of features. If two classifiers are found which have a low error correlation then it may be possible to combine them in order to attain a classification accuracy greater than either classifier alone can provide.

This project develops a new set of image features and investigates their use for performing OCR on bi-tonal images. Fuzzy-knn and neural-network based classifiers are constructed and their performance is compared to that of an independent neural-network-based classifier which makes use of a completely different feature set. In addition, the error correlation between each pair of classifiers is measured.

The goals of this project are to develop a set of image features for performing OCR on bi-tonal images and to develop image-processing software for handling the features, including neural-network and fuzzy-knn based classifiers. The features need to perform well and should differ from those typically used for OCR, with the intention that the errors made by classifiers based on them will not be highly correlated with those made by existing classifiers. In addition, the software is integrated with the Gamera framework and made available as an open-source project.

The project has achieved these goals and represents a significant contribution in the following areas of OCR:

1. **Theoretical Contribution**: New image features were developed which are based on Fourier Descriptors. The features are novel and different from typical features used for OCR.

2. **Material Contribution**:

   (a) An open-source software library for extracting the features from images was created and made publicly available.

   (b) Open-source software was created which implements fuzzy-knn and neural-network based classifiers, both of which use the new image features to perform OCR.

(c) This software was integrated into an existing open-source framework for document processing research.

3. **Experimental Results**: It was experimentally verified that the features have the desired attributes of good performance and, in addition, have errors which are not highly correlated with those of classifiers making use of other features.

Chapter 2

BACKGROUND

This chapter covers related work in optical character recognition and in applications of the Fourier Transform that are similar to those presented in this paper. It covers the Gamera [22] framework and discusses the use of `C++` templates within that system.

2.1 Related work

2.1.1 Fourier Descriptors

The term "Fourier Descriptor" describes a family of related image features, originally introduced by Cosgriff [17]. Generally, the term refers to the use of a Fourier Transform to analyze a closed planar curve. In the context of OCR, the curve is generally derived from a character boundary. Since each of a character's boundaries is a closed curve, the sequence of (x, y) coordinates which specifies the curve is periodic. This makes it ideal for analysis with a Discrete Fourier Transform. There are several variations of Fourier Descriptor features. The method developed in this project is most similar to the Elliptic Fourier Descriptors used by Kuhl and Giardina [6]. This method involves applying separate Fourier transforms to the sequence of x components and the sequence of y components of each curve. The formulation used by Zahn and Roskies [3] is to apply the Fourier Transform to the function which gives the total angular change between a point on the curve and some fixed starting point on the curve. The method used by Granlund [7] is to apply the transform to the sequence of complex numbers formed by $x + i\ y$, where the point on the curve is (x, y).

The features used in [6] and [3] allow the curve to be reconstructed exactly from the feature vector, provided that all components of the transform are saved. It is not typically necessary to do this, however, as most of the information about typical curves is contained in the low frequency components of their transforms. Similarly, the features investigated in this project are formed by saving only a fixed number of the low-frequency components of the transform. None of the prior works discussed here provide a means to handle images which contain multiple curves. In contrast, the features investigated in this project can be applied to images containing an arbitrary number and arrangement of curves.

The Descriptors of Kuhl and Giardina

The features used in [6] are formed from closed, piecewise continuous, planar curves. Each curve comprises line segments chosen from the following set of 8 vectors, represented in phasor notation:

$$\left\{ \left[1 + \frac{\sqrt{2}-1}{2} \left(1 - (-1)^i \right) \right] \angle \frac{\pi}{4} i : i \epsilon \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\} \right\} \tag{2.1}$$

The Fourier Transform is applied separately to the x and y projections of the curves. Kuhl and Giardina apply their features to the problem of classifying 2-dimensional projections of 3-dimensional objects.

The Descriptors of Zahn and Roskies

The features used in [3] are constructed as follows:

1. Let C be a clockwise parameterization of a curve in terms of its arc-length, beginning at an arbitrary point, $C(0)$. This implies that $\left|\frac{\partial C(l)}{\partial l}\right| \equiv 1$. Let the total arc-length of the curve be L.

2. Let $\theta(l)$ be the angular direction of the vector $\frac{\partial C(l)}{\partial l}$.

3. Let $\phi(l)$ be the total angular change between $C(l)$ and $C(0)$. Thus:

$$\phi(l) = \theta(l) - \theta(0) \pmod{2\pi} \tag{2.2}$$

4. Let $\phi^*(l) = \phi\left(\frac{Ll}{2\pi}\right) + l$. Note that $\phi^*(0) = 0$ and $\phi^*(L) = -2\pi$ hold for any closed curve. The domain of ϕ^* is $[0,\ 2\pi]$. Also, note that $\phi^*(0) = 0$ and $\phi^*(2\pi) = 0$, and that $\phi^* \equiv 0$ when the curve is a circle.

The feature vector is formed by computing the Fourier Transform of ϕ^* over the domain $[0,\ L]$ and discarding all but some specified number of the transform's low-order coefficients.

The Descriptors of Granlund

The features used in [7] are constructed as follows:

1. Given a curve, C, let $(x(t), \ y(t))$ be the position, at time t, of a point moving on C with constant velocity and period 2π. Define $u(t)$ as follows:

$$u(t) = x(t) + i\ y(t) \tag{2.3}$$

2. Define a_n as follows:

$$a_n = \frac{1}{2\pi}\int_0^{2\pi} u(t)\ e^{-nti} dt \tag{2.4}$$

3. Let $d_{m,n}$ be defined as follows:

$$d_{m,n} = \frac{{a_{1-m}}^{\frac{n}{gcd(m,n)}}\, {a_{1-n}}^{\frac{m}{gcd(m,n)}}}{{a_1}^{\frac{m+n}{gcd(m,n)}}} \tag{2.5}$$

4. The feature vector for a given curve is defined to be $[d_{1,1}\ d_{2,2}\ d_{2,1}\ d_{1,2}\ d_{3,1}\ d_{1,3}\ d_{4,4}]$.

2.1.2 Sequence Matching

Similar uses of the Fourier Transform to reduce the dimensionality of a numeric sequence can be found in applications where the sequence is derived from a data source other than an image. For example, if the price of a particular good is measured at uniform time intervals, the resulting sequence of prices often shows variation over large time intervals which is much higher in magnitude than any variation shown over small time intervals. This means that the Fourier Transform can be applied to such sequences and then truncated in order to reduce their dimensionality without losing the bulk of the information that they contain.

Sequence matching refers to the following scenario: Given a database of sequences and a query sequence, we wish to find all sequences in the database which are within some distance ϵ of the query sequence according to a Euclidean distance metric. An application of the Fourier Transform to sequence matching is given in Agrawal [16]. Agrawal, Faloutsos, and Swami apply the Fourier Transform to their sequences to reduce their dimensionality under the assumption that the low-frequency components of the transform will typically carry most of the sequence's information. If the number of transform components saved is kept small enough then an approximate search on them can be carried out efficiently using any of a variety of multidimensional data structures. Since many of the high-frequency components are ignored by this search it can produce incorrect matches which must be removed in a later step. However, the fact that the low-frequency components tend to describe the sequence well means that such false matches will be few in number. In addition, Parseval's theorem guarantees that the Euclidean distance between any pair of sequences is the same as the distance between their corresponding transforms. This allows the results of a similarity search carried out using the frequency components to be meaningful within the time (or spatial) domain.

2.1.3 Optical Character Recognition

There exist many features and classification methods which are useful for performing OCR. A few of the most common techniques are briefly discussed here. For a survey of OCR techniques see [14].

Template Matching

Template matching is an image classification technique which attempts to match an input image against a database of template images in order to find the template images which are most similar to it. Various measures of similarity are possible. In all cases the matching is performed at the pixel level rather than being based on features extracted from the image. Thus, the template and input images must have the same dimensions in order to be comparable at all. Some pre-processing such as scaling or augmentation may be needed in order to ensure that the template and input images have the same dimensions. Examples of similarity measures used are:

1. Minkowski distance: The Minkowski distance (or dissimilarity), M, between two images a and b, each containing N pixels, is given by Equation (2.6), where $a\left(x_i, y_i\right)$ and $b\left(x_i, y_i\right)$ give, respectively, the value of the pixel at coordinate $\left(x_i, y_i\right)$ in images a and b. Higher values of M imply a lower degree of similarity between the two images. Note that this metric can be applied to both gray-scale and bi-tonal images.

$$M = \sqrt[p]{\sum_{i=1}^{N} \left(a\left(x_i, y_i\right) - b\left(x_i, y_i\right)\right)^p} \tag{2.6}$$

2. Yule's Q statistic [8]: If A and B are sets then Yule's Q statistic gives a measure of correlation between them which is shown in Equation (2.7). In the case of matching a pair of bi-tonal images, a and b, the sets A and B contain the coordinates (x, y) at which images a and b, respectively, have a darkened pixel. Higher values of Q imply a greater degree of similarity between the two images. Note that this measure applies to bi-tonal images only.

$$Q = \frac{|A \cap B| \left|\overline{A} \cap \overline{B}\right| - \left|\overline{A} \cap B\right| \left|A \cap \overline{B}\right|}{|A \cap B| \left|\overline{A} \cap \overline{B}\right| + \left|\overline{A} \cap B\right| \left|A \cap \overline{B}\right|} \tag{2.7}$$

3. Jaccard distance, shown in Equation (2.8) can be used to measure the similarity between two bi-tonal images. Higher values of J imply a greater degree of similarity between the two images.

$$J = \frac{|A \cap B|}{|A \cap B| + \left|\overline{A} \cap B\right| + \left|A \cap \overline{B}\right|} \tag{2.8}$$

Geometric Moments

Template matching is a classification technique which uses the input image directly. In contrast, geometric moments can be used to build a set of image features which can be given as input to any of a variety of general-purpose classification methods. Geometric moment invariants were originally proposed as features for image classification by Hu [13]. Each geometric moment $\mu_{p,q}$, of image a, of order $p+q$, about the point (x,y) is computed according to Equation (2.9). Various functions of the moments $\mu_{p,q}$ can then be defined which are invariant under certain transformations such as scaling and rotation. These invariants can be used as features for image classification. For example, Hu [13] demonstrated that (2.10) is invariant under image scaling. Functions of geometric moments which are invariant under general linear transformations can be found in Reiss [19].

$$\mu_{p,q} = \sum_{i=1}^{N} a\left(x_i, y_i\right)\left(x_i - x\right)^p (y_i - y)^q \tag{2.9}$$

$$\frac{\mu_{p,q}}{{\mu_{0,0}}^{1+\frac{p+q}{2}}} \tag{2.10}$$

Zernike Moments

Zernike moments are features constructed by viewing an image as a function of two variables and projecting it onto the orthogonal basis formed by the family of functions $V_{p,q}\left(\rho,\theta\right)$. These are given in [9] and are shown in Equation (2.11), where:

1. p is a non-negative integer

2. q is an integer less than or equal in magnitude to p such that $p-|q|$ is even

3. $(\rho,\theta)\,\epsilon R\times[0,2\pi)$

The Zernike basis functions are orthogonal but not orthonormal [9].

$$V_{p,q}(\rho,\theta) = exp(iq\theta) \sum_{s=0}^{\frac{p-|q|}{2}} \frac{(-1)^s (p-s)!}{s! \left(\frac{p+|q|}{2} - s\right)! \left(\frac{p-|q|}{2} - s\right)!} \rho^{p-2s} \tag{2.11}$$

2.2 Gamera

Gamera [22] is an extensible framework for developing document analysis applications. It was developed at the Digital Knowledge Center at Johns Hopkins University by Michael Droettboom, Ichiro Fujinaga and Karl MacMillian. It provides a library of image processing algorithms as well as a graphical user interface framework. Gamera is based on the object-oriented language Python [23]. Python is interpreted and provides dynamic typing which makes it suitable as a “glue” language as well as for interactive scripting. Gamera provides a facility for creating plug-ins via `C++` templates which can be leveraged from Python. This allows performance-critical algorithms to be implemented efficiently in `C++` while allowing them to be utilized from a scripted environment.

2.3 `C++` Templates

The Gamera framework is capable of handling various image formats in a way that is relatively seamless to the user. In addition, most image processing algorithms can be implemented more efficiently in `C++` than in Python. This means not only that the image processing algorithms in Gamera must be implemented in `C++`, but that the `C++` code must be specialized for different image formats. `C++` templates and function overloading provide a natural way to accomplish this by allowing the following two scenarios to be handled in a way that is transparent to the code which is calling the image processing algorithm (in this case, the Gamera build system):

1. If an image-processing algorithm can be expressed in a way that allows it to be directly applied to many different image types then the algorithm can be written as a function template which accepts the image type as a template parameter. This template will then be instantiated by the Gamera build system for each image type that it applies to. The template code is written in terms of a set of methods that are common to all image types in Gamera. Since the compiler has all available type information about the image object being passed to the function it will be able to generate more efficient code than it could if the function was simply written against a common base class that all image types were derived from.

2. If an algorithm must be expressed in a fundamentally different way for each image type then it can be implemented in terms of several different functions which are overloaded based on the image type. The function can then be called by the Gamera build system with an instance of a specific image type and the correct version of the function will be bound seamlessly by the compiler.

The Gamera Python API provides an `Image` class which can hold an image in a variety of formats. In both of the above cases the Gamera build system generates code which performs roughly the following steps when an operation is invoked on an `Image` object:

1. find the image's format
2. switch based on that format
3. cast the image data to the native format
4. pass the image data to the native `C++` function which was specialized for that image type at compile-time

Essentially, this mechanism allows the function dispatch overhead to be incurred a single time for each operation on the Python image object rather than once for each low-level operation within the image processing code.

Chapter 3

PROJECT

This chapter covers the overall goals of the project and the integration provided with the Gamera framework. The neural network architecture selection, the training methods used, the training data storage format, the features used by the classifier, the methods for feature extraction, as well as the advantages and disadvantages of this feature set.

This project has the following goals:

1. Develop a set of features which can be used to perform OCR on bi-tonal images. The features should yield good classifier accuracy and should be computationally efficient to extract and manipulate. In addition, the features should differ sufficiently from those used in other classifiers so that the errors made by classifiers based on them are not highly correlated with the errors made by existing classifiers. This property of decorrelated errors enables classifiers to be combined in order to improve overall classification accuracy. This is discussed in Section 3.6.

2. Develop software for extracting the features mentioned above from an image.

This software will be made publicly available to enable future classifiers to make use of these features.

3. Implement software classifiers which make use of these features in order to classify character images. Two classifiers are developed: one which employs neural-networks and another which employs fuzzy-nearest-neighbor search. This software will be made publicly available. These classifiers are discussed in Section 3.3 and Section 3.4, respectively.

4. Add support to the Gamera system for using the neural-network-based classifier above. This addition to the Gamera system will be made publicly available.

5. Extend the Gamera `Image` class to provide support for the SDF image format. This addition to the Gamera system will be made publicly available. The SDF image format is discussed in Section 3.5.1.

3.1 Gamera

Gamera provides a Python-based framework for creating document analysis applications. This framework can be extended through plug-ins which are implemented in `C++` and bound to methods on Gamera's Python `Image` class. This binding is accomplished by the Gamera build system. The build system must be provided with meta-data regarding the plug-in and its methods. For the purposes of this project, several methods were added to the `Image` class for extracting the feature data, the image signature, and accessing the image's classification data. Some functions were also added to the built-in Gamera modules, independently of the `Image` class.

3.2 Neural Network

The design choices regarding the neural-network-based classifier and the rationale for them are given in the following sections.

Architecture Selection

A series of two-layer neural-networks is used to implement the classifier. One network is used for each image signature, since the signature determines the number and the semantics of the network inputs. Every unit is completely connected to the units in each layer adjacent to it. Bias inputs are used for units in the middle and output layers. The number of units in the input layer is determined as a configurable function of the image signature. Due to the fact that the number of feature elements per curve is constant, the number of inputs is roughly proportional to the number of curves in the image. The output layer is configured with one output per character class. This allows the network to recognize multiple character classes in a single propagation, which means that computations which are needed for recognizing different characters can be shared within the same network. The overall classifier is made up of multiple networks. Each image is assigned a signature which is used to select the network used to classify it. Image signatures are discussed in Section 3.6.

The hidden layer size can be customized by the user. In order to choose the hidden layer size, the user passes a Python function to the neural network object when it is created. The function accepts parameters which describe properties of the network and it returns a hidden layer size. It's parameters are as follows:

1. *signature*: the image signature; This is defined in Section 3.6.
2. *input size*: the size of the network's input layer
3. *output size*: the size of the network's output layer
4. *training pattern count*: the number of training patterns which will be presented to the network

Training Methods

The basic method used for training the network is back-propagation. Back-propagation essentially involves performing gradient descent in the weight-space of the network in order to minimize the value of the error function. In this project, a stochastic training method is used in which the network weights are adjusted after each training pattern is presented to the network. This is in contrast to batch training, in which the network weights are adjusted only at the end of each training epoch after all patterns have been presented to the network. In comparison to stochastic training, batch training has the advantage that the network weights are being adjusted in response to the true network error function. However, as discussed in Section 3.2, it is more difficult in practice to choose a proper back-propagation step factor for batch training than it is for stochastic training.

Additional Training Techniques

Despite its conceptual simplicity, several additional techniques are needed in order to obtain good performance with back-propagation. The techniques below were implemented according to the guidelines in Section 6.8 of Duda [18]:

1. *input standardization*: It is quite possible that the mean value of certain input elements is much larger than that of others. This means that during back-propagation the weights associated with these input elements will be adjusted at a higher rate than other weights. Since the mean value of different input elements is arbitrary from the perspective of classification, each input is standardized so that the sample mean over the entire training set becomes 0. In addition, the input values are scaled so that their sample variance is equal to 1. This allows the network weights to be initialized in a straightforward way, as described in item 2 below.

2. *initializing the network weights*: The link weights in the network are initialized with random values chosen uniformly from the interval $\left[\frac{-1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right]$, where d is the number of inputs to the unit whose input weights are being initialized. Combined with the fact that the input values are being standardized to a variance of 1, this means that the net activation of each unit will be in the range $[-1,\ 1]$.

3. *stopping criteria*: A pitfall of training is that the network may become overtrained. This is a condition in which the network has very high accuracy on the training set but is unable to accurately classify data that it has not seen before. The stopping criteria used in this project is based on the use of a validation set, on which the neural network is not trained, but which is used to stop training when the network's error on it reaches a minimum. The only additional aspect needed to implement this is a method for determining when the validation error has reached a minimum. This is provided by specifying a maximum iteration count when training is started. If the training process goes through more than this number of iterations without improving the validation error, then training is stopped.

4. *choice of back-propagation step factor*: At each step of the back-propagation algorithm, the gradient of the error function is computed in terms of its partial derivatives with respect to each weight. The magnitude of adjustment for each weight is determined by multiplying the gradient by the scalar *step factor*. Ideally, the step factor is chosen in a way which both reduces the error as quickly as possible and avoids convergence to local minima. A class of methods which rely on the values of the second partial derivatives of the error with respect to the weights are known as *second-order methods*. One such method assumes that the error surface can be approximated by a quadratic function, and chooses the step factor to be $\left(\frac{\partial^2 E}{\partial w^2}\right)^{-1}$ where E is the error function and w

is the weight being adjusted.

Another approach is to choose a fixed step factor which is used for the duration of training. For batch training this method is not generally satisfactory as the size of the weight adjustment vector can be quite large when the number of training examples is large. This strategy can result in a step size that is either too large to allow convergence or which is so small that training is much slower than necessary.

3.3 Neural-Network Based Classifier

Although the neural network provides the core of the functionality needed for classification, there are several additional details that need to be handled that the neural network itself does not implement.

1. The neural network has a series of outputs which are interpreted as confidence values for membership in a particular character class. This mapping is established by the Neural Network Classifier. The classifier builds a dictionary of character classification strings and establishes a mapping between character-class identifiers and numeric values. Character class identifiers are represented as strings. This abstraction allows the classifier to transparently create the appropriate network output patterns during training. It also allows the output of the network to be transparently mapped to a list of (confidence, character-class) pairs after an unknown sample has been classified.

2. As described in Section 3.2, the mean and variance of the training data needs to be standardized in order to allow effective training. This of course requires the input pattern associated with any unknown sample to be transformed in the same way before being presented to the network for classification. The process of mapping from image features to network input patterns during both training and classification is handled by the neural network classifier.

3.4 Fuzzy-KNN Based Classifier

A fuzzy K-nearest neighbors classifier (fuzzy-knn) is similar to a K nearest neighbors classifier (knn) [4, 5] but differs in that each output class is assigned a fuzzy set membership value which is derived from the distance between the query point and the set of its nearest neighbors in the training set. The fuzzy K-nearest neighbors algorithm was introduced by Keller in [11]. Given the set of k training images which are nearest to the query point: $\{x_j : 1 \leq j \leq k\}$, the fuzzy set membership assigned to character class C for the query image Q is denoted by $\mu_C(Q)$ and is defined in Equation (3.1):

$$\mu_C(Q) = \frac{\sum_{j=1}^{k} \mu_C(x_j) \frac{1}{|Q-x_j|^{\frac{2}{m-1}}}}{\sum_{j=1}^{k} \frac{1}{|Q-x_j|^{\frac{2}{m-1}}}} \tag{3.1}$$

The value of $\mu_C(x)$ for $x \epsilon \{x_j : 1 \leq j \leq k\}$ is given by Equation (3.2):

$$\mu_C(x) = \begin{cases} 1 & class(x) = C \\ 0 & class(x) \neq C \end{cases} \tag{3.2}$$

The parameter m in Equation (3.1) ranges over $(1, \infty)$ and controls the "fuzziness" of the distance metric. As m increases, the k neighbors become more evenly weighted regardless of their distance from Q. As m decreases toward 1 the neighbors which are nearest to Q become weighted more and more heavily compared to those which are farther away from Q.

3.5 Training Data Format

The data used for classifier training is stored in a format known as SDF. The SDF format represents a series of bi-tonal images using bit-packing. This format results in reduced storage space requirements as well as in reduced I/O costs for a program analyzing large numbers of images.

3.5.1 SDF File Format

The layout of an SDF file is shown in Table 3.1. The *data_size* function referenced in Table 3.1 is defined as follows:

$data\_size(k) = \lfloor \frac{(width\ of\ image\ k) \cdot (height\ of\ image\ k)}{8} \rfloor + 1$

An SDF file has three sections:

1. *image count*: The number of images contained in the file is stored in the first

Offset	**Byte Count**	**Contents**
0	1	$\lfloor \frac{N}{2^{24}} \rfloor$ *mod* 2^8
1	1	$\lfloor \frac{N}{2^{16}} \rfloor$ *mod* 2^8
2	1	$\lfloor \frac{N}{2^{8}} \rfloor$ *mod* 2^8
3	1	N *mod* 2^8
4	12	image header 0
...	...	...
$12 \cdot k + 4$	12	image header k
...	...	...
$12 \cdot (N-1) + 4$	12	image header $N-1$
$12 \cdot N + 4$	data_size(0)	pixels for image 0
$12 \cdot N + 4 + data\_size(0)$	data_size(1)	pixels for image 1
$12 \cdot N + 4 + data\_size(0) + data\_size(1)$	data_size(2)	pixels for image 2
...	...	...

TABLE 3.1 The SDF File Layout.

four bytes of the file in big-endian format.

2. *image headers*: If the number of images in the file is n, then a sequence of n image headers follows the image count in the file. The layout of an image header is shown in Table 3.2. The first four bytes of the image header are not used. Offsets four through nine store the classification of the sample. Offsets 10 and 11 give the height and width of the image, respectively, in pixels.

3. *image pixel data*: This section contains the bit-packed pixel data for each image. The data for an image with height h and width w occupies $1 + \lfloor \frac{h \cdot w}{8} \rfloor$ bytes. Since only $\lceil \frac{h \cdot w}{8} \rceil$ bytes are required, the additional byte is ignored if present.

Description	**Header Offset**	**Value**
(unused)	0	0
	1	0
	2	0
	3	0
classification	4	1st byte of image class (or 0 if unused)
	5	2st byte of image class (or 0 if unused)
	6	3st byte of image class (or 0 if unused)
	7	4st byte of image class (or 0 if unused)
	8	5st byte of image class (or 0 if unused)
	9	6st byte of image class (or 0 if unused)
image size	10	image height in pixels
	11	image width in pixels

TABLE 3.2 The Image Header Layout

3.5.2 SDF Image Format

The pixels in an SDF image with height h and width w are represented as a sequence of bytes of length $\lfloor \frac{h \cdot w}{8} \rfloor + 1$. The image coordinates run from 0 to $w - 1$ in the left-to-right direction and from 0 to $h - 1$ in the top-to-bottom direction. Let $P\left(x, y\right) \epsilon \left\{0, 1\right\}$ represent the value of the pixel at position (x, y) within an image, with 0 representing a white pixel and 1 representing a black pixel. Let $I(k)$ be the kth byte of the image data. Then the value of $I(k)$ is given by Equation (3.3):

$$I(k) = \sum_{i=0}^{7} 2^{7-\left(i \cdot P\left((8 \cdot k+i) \ mod \ w, \lfloor \frac{8 \cdot k+i}{w} \rfloor\right)\right)} \tag{3.3}$$

This format offers fairly efficient use of storage space for bi-tonal images, while allowing them to be decoded with a minimum of overhead.

3.6 Features

This section describes the feature set used by the classifier. The feature set is derived from the closed curves which bound connected components in an image. The features extracted from an image take the form of a feature vector as well as a "signature". Two images which have the same signature have feature vectors which are the same size. This section introduces some basic terminology in 3.6, the feature vector is described in 3.6, and the image signature is described in 3.6.

Terms

1. *Adjacent Pixels*

   Two pixels are *adjacent* if one is a member of the N_8 set of neighbors of the other.

2. *Connected Pixels*

   Two pixels `a` and `b` are *connected* if they have the same color and:

   (a) they are adjacent or,

   (b) `a` is adjacent to a pixel which is connected to `b`.

3. *Connected Component*:

   A *Connected Component* is a set of black pixels which are pairwise connected.

   Figure 3.1 shows a character image which contains a single connected component.

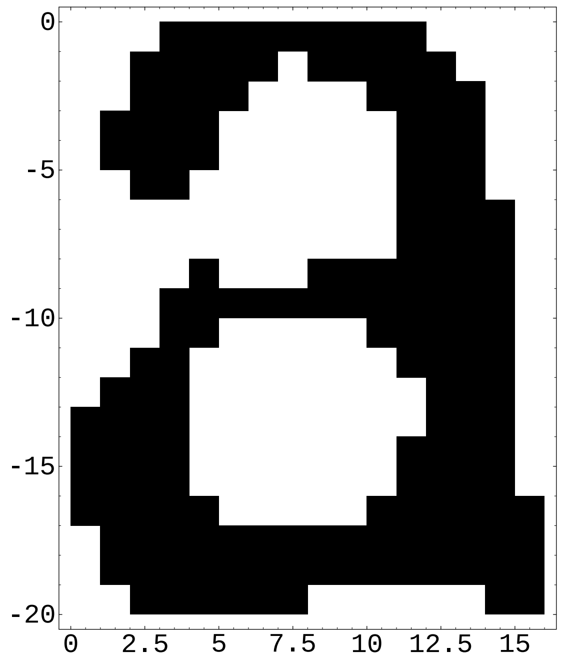

Figure 3.1. An image containing a single connected component

4. *Curve*:

   A *curve* is a polygon which forms a boundary of a connected component. Each segment of the polygon is located either at the boundary between two adjacent pixels that are of different color or at the edge of a black pixel located at the extreme left, right, top, or bottom of the image. Note that the connected component in Figure 3.1 has an inner boundary as well as an outer boundary. Each curve is represented initially as a sequence of (x, y) coordinate pairs, where each coordinate pair locates the corner of a pixel. By convention, the coordinate sequence is ordered counter-clockwise for curves which enclose black pixels. The sequence is ordered clockwise for curves which enclose white pixels.

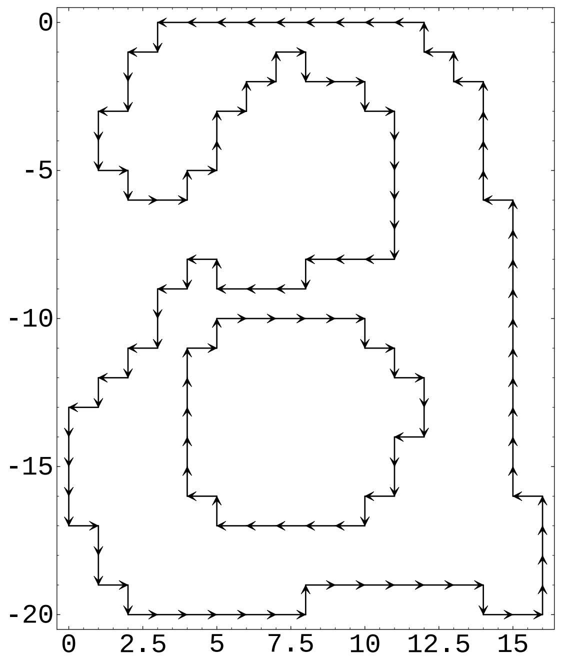

Figure 3.2. The curve representation of the example image

Figure 3.2 shows the curves associated with the image in Figure 3.1. Note that each segment in Figure 3.2 is displayed as an arrow. In addition, the arrows which form the outer boundary of the character point in the counter-clockwise direction around the boundary. The arrows which form the inner boundary of the character point clockwise around the boundary. The initial coordinate sequence for a curve is formed by picking an arbitrary vertex of the polygon, noting its (x, y) coordinates, moving one pixel-width to the adjacent vertex in the direction of the arrow, and repeating this process until all vertices in the polygon have been visited.

5. *Area*:

This section defines the *area* of a curve. For curves which are oriented counter-clockwise, this definition is identical to the standard one. The definition given here has the property that if the orientations of the edges in a curve are all reversed, then its area changes sign. The area of an arbitrary curve, C, is given by (3.4):

$$\frac{1}{2} \sum_{e \epsilon Edges(C)} (StartX(e) + EndX(e))(EndY(e) - StartY(e)) \tag{3.4}$$

where $Edges(C)$ gives the set of edges in C, with each edge having a start and an end point, as in Figure 3.2, and where $StartX(e)$, $EndX(e)$, $StartY(e)$, and $EndY(e)$ give the starting x, ending x, starting y, and ending y coordinates of edge e, respectively. It is not hard to show that for an arbitrary polygon which is oriented counter-clockwise, (3.4) gives the standard definition of the area enclosed by it and that, for a polygon which is oriented clockwise, (3.4) gives the negative of the area enclosed by it.

(3.4) has a positive value for a curve which encloses a black region. Such a curve is defined to be a *positive-area* curve. Analogously, (3.4) has a negative value for a curve which encloses a white region. Such a curve is defined to be a *negative-area* curve.

6. *Centroid*:

The *centroid* of a curve is defined to be the (x, y) coordinate pair which is the average of the coordinates of all vertices of the curve.

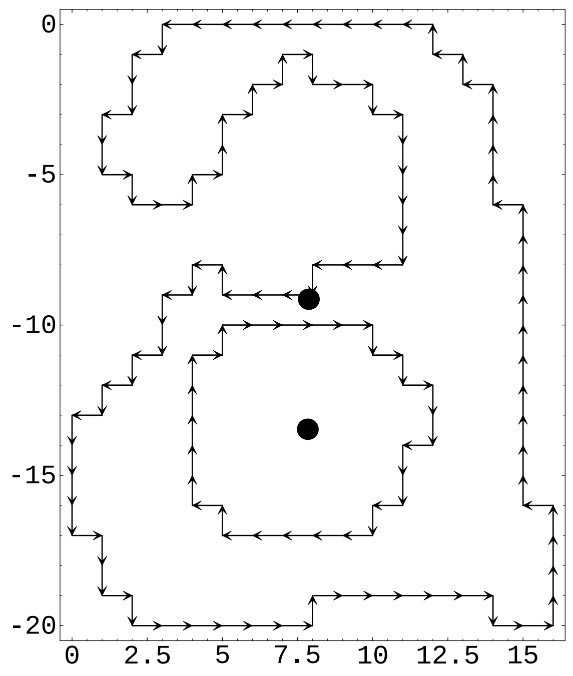

Figure 3.3. The curve representation of the example image, with centroids

Figure 3.3 shows the curves of an example image along with two black dots which indicate the centroid position for each curve. The upper dot marks the centroid of the outer curve and the lower dot marks the centroid of the inner curve.

7. *Curve Ordinal*:

Let `C` be a positive(negative)-area curve with centroid (Cx, Cy). The ordinal of C is a pair of non-negative integers $(xOrd, yOrd)$, where $xOrd$ is the number of positive(negative) area curves in the image whose centroids have an x-component which is smaller than Cx. Similarly, $yOrd$ is the number of positive(negative) area curves in the image whose centroids have a y-component which is smaller than Cy.

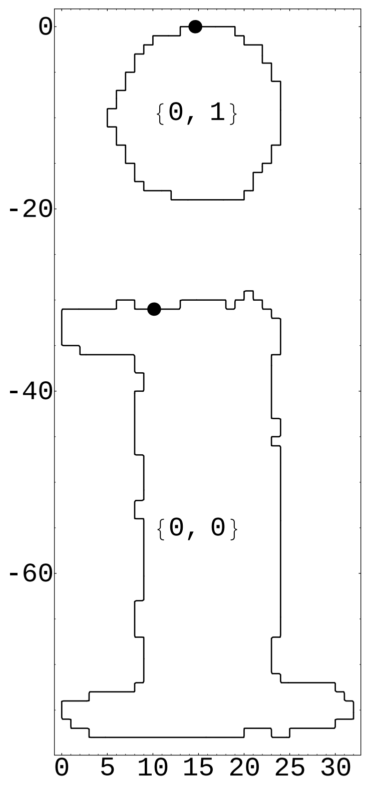

Figure 3.4.: The curve ordinals of the pair of curves associated with a lowercase “i”.

Figure 3.4 shows a lowercase “i” with the ordinal of each curve labeled. Note that the curve associated with the upper “dot” of the “i” has an ordinal of $(0, 1)$, while the lower curve has an ordinal of $(0, 0)$. The $x - coordinates$ of the two ordinals are equal, indicating that their horizontal positions are approximately the same. The $y - coordinates$ of the lower and upper curves are 0 and 1, respectively. This encodes the fact that the lower curve is "below" the upper curve. Taken together, the two curve ordinals encode the positional relationship between the curves in the image.

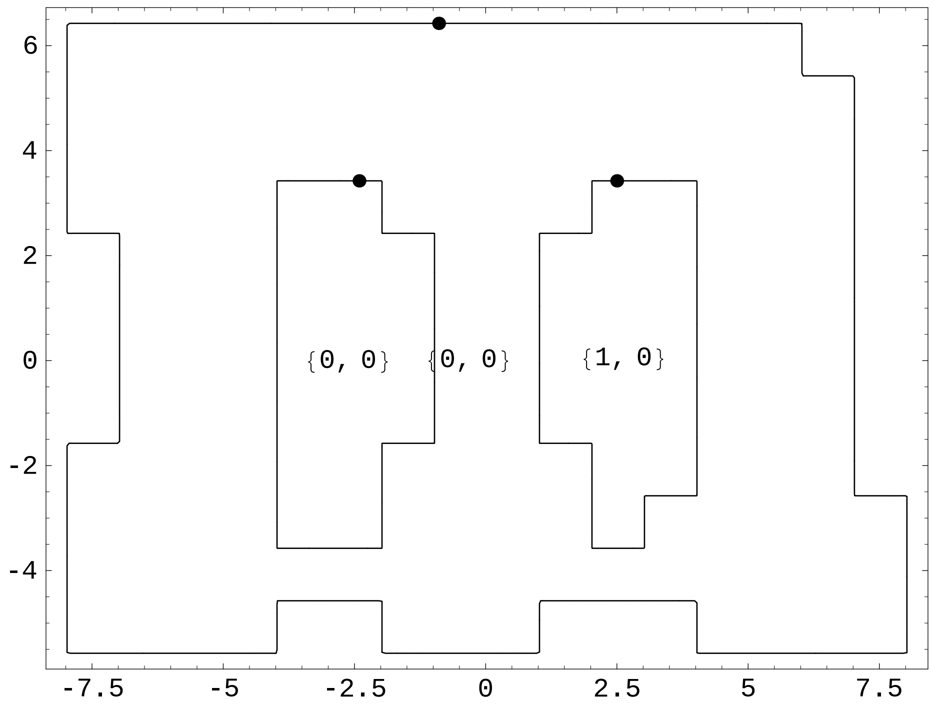

Figure 3.5. The curve ordinals of a pair of curves associated with a lowercase m.

Figure 3.5 shows a lowercase "m" with the ordinal of each curve labeled. Note that this image contains a single positive-area curve as well as two negative-area curves. Note that both the outer curve and the leftmost inner curve have the ordinal $(0, 0)$. This is due to the fact that curve ordinals are determined by comparing the centroids of curves with area of the same sign. Thus, since the two inner curves have negative area, their ordinals are computed without taking into account the presence of the outer positive-area curve.

Feature Vector

The feature vector for an image is constructed using the curves which bound the connected-components in the image. The feature vector is formed by concatenating the following values:

1. centroid difference

   The centroids of the positive area curves in the sample are averaged to form an (x, y) pair. Similarly, the centroids of the negative area curves are averaged. The difference of these two averages is an (x, y) pair which is placed into the feature vector.

2. centroid offset of each curve

   For each curve: If it is positive(negative)-area then the average of the positive(negative)-area curve centroids is subtracted from the curve's centroid. The result is an (x, y) pair which is placed into the feature vector. These curve centroid offsets are placed into the feature vector in lexicographic order by curve ordinal and by whether the curve has positive or negative area.

3. discrete Fourier transform

   In order for the Fourier Transforms of different curves to be directly comparable each curve needs to be defined by the same number of points. In order to accomplish this, linear interpolation is used in order to treat the curve as a periodic, continuous, piecewise-linear parametric function of a single independent variable. This function is then sampled at 2^n points in its domain, where n is a parameter determined before training begins. The same value of n will be used for every curve that the classifier is trained on as well as to unknown images when they are being classified. The implementation of the Fourier Transform being used in this project is a Fast-Fourier-Transform and it requires the number of points sampled on each curve to be a power of 2, which is why this constraint is imposed.

   Each curve is uniquely identified by the sequence of (x, y) coordinates of its vertices. The x and y coordinate sequences can be viewed independently as discrete functions of an independent parameter. Both of these functions are periodic, which makes them well-suited to be analyzed using a Discrete Fourier Transform (DFT). The x-sequence transform of a curve will be called the “x-DFT”. The y-sequence transform will be called the “y-DFT”. Some fixed number of the DFT’s lowest frequency components are preserved while the other higher-frequency components can be discarded. The low-frequency components of the transform will typically contain most of the information about the sample while the high-frequency components often represent noise of various kinds. The ability to ignore high-frequencies in the DFT allows the dimensionality of the feature vector to be reduced without sacrificing information which is likely to be crucial in identifying the sample. For each curve, the feature vector contains the components of the curve's x-DFT followed by the components of its y-DFT.

The curve detection process does not guarantee that the point sequences for two similar curves will begin at similar relative points in their cycles. In fact, the point at which a given curve starts can differ significantly between curves that are only slightly different. In order for two similar curves to be compared accurately, their point sequences need to start at similar positions in their overall cycles. To this end, the components in the Fourier transforms are all effectively shifted in phase by some number of time units, T. The value of T is chosen so that the phase of the component of the y-coordinate spectrum corresponding to the analyzing function $exp\left(\frac{-2\pi it}{N}\right)$ becomes 0 after shifting by T time units. The rationale for choosing this component is that, by definition, it undergoes exactly one cycle during the sequence. Thus, it has exactly one point at which the phase is zero. The value of T is used to shift both the x and y coordinate sequences.

The DFTs of the negative-area curves are placed into the feature vector in lexicographic order by curve ordinal. Similarly, the DFTs of the positive-area curves are placed into the feature vector in lexicographic order by curve ordinal.

Thus, given two images which have the same signature, the DFTs of any pair of curves within the images which have the same curve ordinal and the same area-sign will appear at the same position within their respective feature vectors.

Image Signature

The features described in Section 3.6 are such that the size of the feature vector for an image is dependent upon the number of positive-area and negative-area curves in the image. In order to determine when two different images will have feature vectors that are comparable, a "signature" is assigned to each image. An image signature has the following property: If two images have the same signature then the images have the same arrangement of curves and will consequently have feature vectors that are the same size. The signature for an image is defined to be the ordered pair (N, P), where N is the set of ordinals for the negative-area curves in the image and P is the set of ordinals for the positive-area curves in the image. If the signature of image A is (N_1, P_1) and the signature of image B is (N_2, P_2) then $(N_1, P_1) = (N_2, P_2)$ holds if and only if both of the following conditions hold:

1. N_1 and N_2 contain exactly the same curve ordinals

2. P_1 and P_2 contain exactly the same curve ordinals

Thus, a signature identifies a set of images whose feature vectors can be meaningfully compared to one another. Note that images A and B will have feature vectors of the same size if and only if $|N_1| = |N_2|$ and $|P_1| = |P_2|$. However, if the signatures of

A and B are not equal then the curves which are encoded within their feature vectors will not necessarily have equivalent positional relationships within their respective images.

3.6.1 Pre-Processing

Before constructing the feature vector for a given sample, some preliminary steps are performed:

1. *detect curves*: A curve forms the boundary of a component in the sample. Curves are detected by scanning the sample image in order to find components and their boundaries.

2. *remove curves having sufficiently small area*: Some curves in the sample may represent noise due to printing errors or imperfections in the physical media, etc. The set of curves is filtered by first finding the curve in the image that encloses the largest area. Let this area be A. Any curve whose area falls within the interval $(-0.03A, 0.055A)$ is removed from the curve set on the basis that it is likely to represent noise. The particular interval used here was chosen because it worked well for removing noise from the sample data being used in this project.

3. *find the ordinal for each curve*: Each curve in a sample has a unique centroid which is used to form an approximate description of the curve's position relative

to other curves in the sample. A "curve ordinal" (x-ordinal, y-ordinal) of integer values is assigned to each curve with centroid (cx, cy) using the following calculation: The x-ordinal of the pair is set to the number of other curves in the sample with the same area sign (positive/negative) having centroids whose x-coordinate is less than cx. Similarly, the y-ordinal of the pair is set to the number of other curves in the sample with the same area sign having centroids whose y-coordinate is less than cy. The ordinal pair for a curve thus provides a rough description of the position that it occupies relative to other curves within the sample which have the same area sign. For some characters, such as an upper-case "B", the curve ordinal computation would ideally assign the ordinals $(0, 0)$ and $(0, 1)$ to the two negative-area curves. This accurately captures the conceptual relationship between the two curves, which is essentially that one curve is directly above the other. In practice, however, the sample is likely to contain noise which will cause the x-coordinates of the two negative-area curves to differ. So, this will tend to result in the ordinal assignments $\{(0, 0), (1, 1)\}$ or $\{(1, 0), (0, 1)\}$ for samples which are only slightly different from each other. In order to remedy this situation, the curve centroids are pre-processed before ordinals are assigned to them. For the purposes of determining the curve ordinals, the x-coordinates of the centroids of the curves of positive(negative) area are sorted, and any run of coordinates in which adjacent values differ by less than 10% of the sample width are each replaced with the average value of all elements in the run. The centroid y-coordinates are updated similarly, with the spacing tolerance being 10% of the sample height.

3.7 Reconstruction of Images From Features

The boundaries in an image can be reconstructed using the features described in Section 3.6. Depending on the classification task, this can be helpful for determining both the number of points to sample on each curve as well as the number of Fourier components to save. Choosing a low number of Fourier components will result in lower training and classification times due to the fact that the neural network will have fewer nodes and weights. The use of the Fourier transform described in Section 3.6 has the property that most of the information about a curve is contained in the low-frequency components of the transform. Thus, the amount of information about the original image which is contained in the feature data can be adjusted smoothly by changing the number of high-frequency components which are discarded.

Figure 3.6 shows the unprocessed boundaries that are present in an image of a "B". Figure 3.7 shows a processed version of the same image which was reconstructed from the feature data described in Section 3.6, without discarding any of the Fourier components. There are a few things to note about the reconstruction:

1. The character boundaries show essentially the same level of detail as the original image. Even the pixel edges are visible.

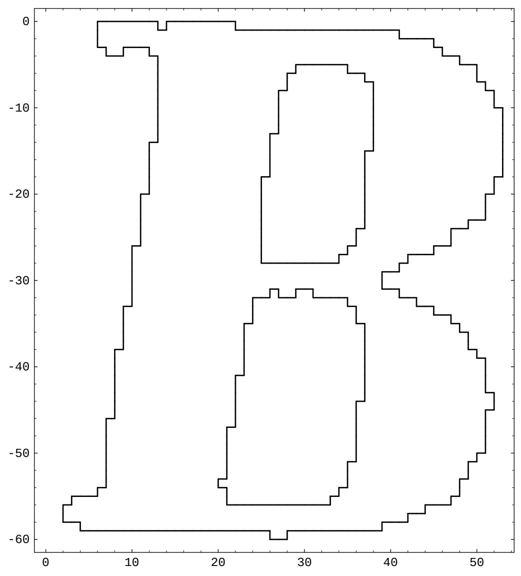

Figure 3.6. Original image boundaries of a “B”

2. Each curve is marked at some point with a small dot. This point represents the zero-phase point for the curve. In order for similar curves to be compared effectively it is important that their coordinate sequences begin at points which represent the “same place” on the two curves, even if the two curves are not identical.

3. Each curve is labeled with a coordinate pair near its centroid, which specifies the curve ordinal. Note also that the curve ordinals of the two inner curves are $(0, 0)$ and $(0, 1)$. This means that the two curves are represented in the feature data as if they were aligned vertically, even though their centroids clearly do not have the same y-coordinate. In addition, note that the ordinals are computed separately for positive-area curves (the outer curve in this image) and

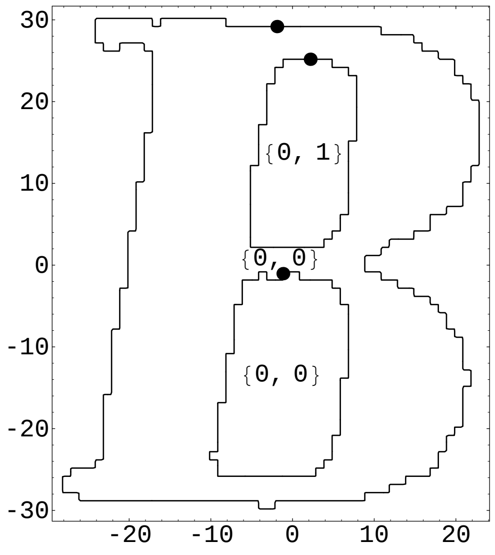

Figure 3.7. Reconstruction of a “B” from high-resolution feature data for negative-area curves (the inner two curves in this image). This is evidenced by the fact that the ordinals of both the outer curve and the lower inner curve are equal to $(0, 0)$.

Figure 3.8 shows a reconstruction of the “B” image from feature data which contains only the first 7 non-zero frequency components of the Fourier transform. This choice of resolution appears to leave enough detail for most of the important image features to be recognizable. The character still retains many of its features. In doing this, the noise associated with pixel boundaries has been removed. This is due to the fact that the 90 degree angle made by a pixel edge is associated with high-frequencies in the spectrum of the curve. Ignoring these frequencies and then reconstructing the image effectively reduces the noise associated with them.

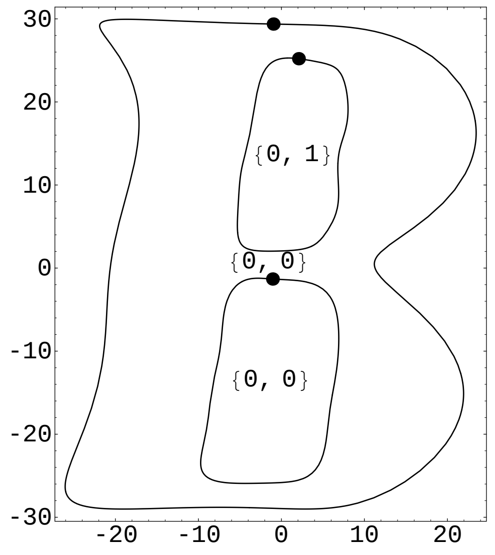

Figure 3.8. Reconstruction of a “B” from medium-resolution feature data

Figure 3.9 shows a reconstruction of the “B” image using the single lowest-frequency component (excluding the constant frequency term) of each curve’s transform. This results in every curve being reconstructed as an ellipse. This level of resolution typically would not be sufficient to allow recognition of general character images on its own, although it could possibly be used in conjunction with other classification methods.

Figure 3.10 shows two reconstructions of different images of “h” characters. The main aspect illustrated by these two images is that their zero-phase markers are located in similar places on each character, even though the images are different. The zero-phase point seems to work fairly well as a way of identifying similar points on different curves.

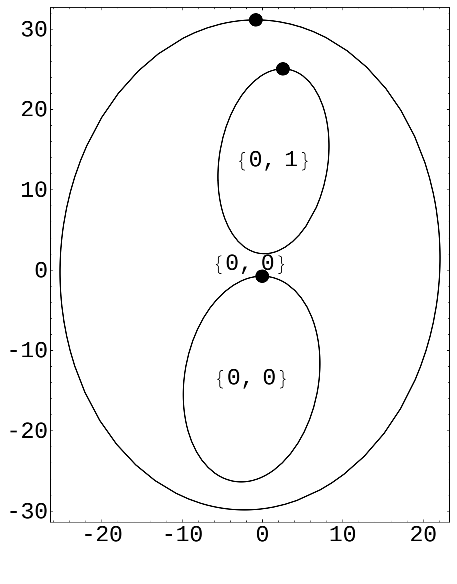

Figure 3.9. Reconstruction of a “B” from low-resolution feature data

3.8 Advantages of Features

The main advantages of the feature set developed in this paper are as follows:

1. *translation-invariance*: The constant term of the Fourier-transform identifies only the centroid of the curve, and can be ignored without losing any information about the shape of a curve. Only the *relative* positions of centroids are stored in the feature data. This means the character can be translated anywhere within the image without changing the features that are generated for it.

2. *noise-robustness*: Curves which have an area below a certain threshold relative to the largest-area-curve in the image can often be ignored on the basis that they represent noise. In addition, since most noise tends to involve high-frequencies,

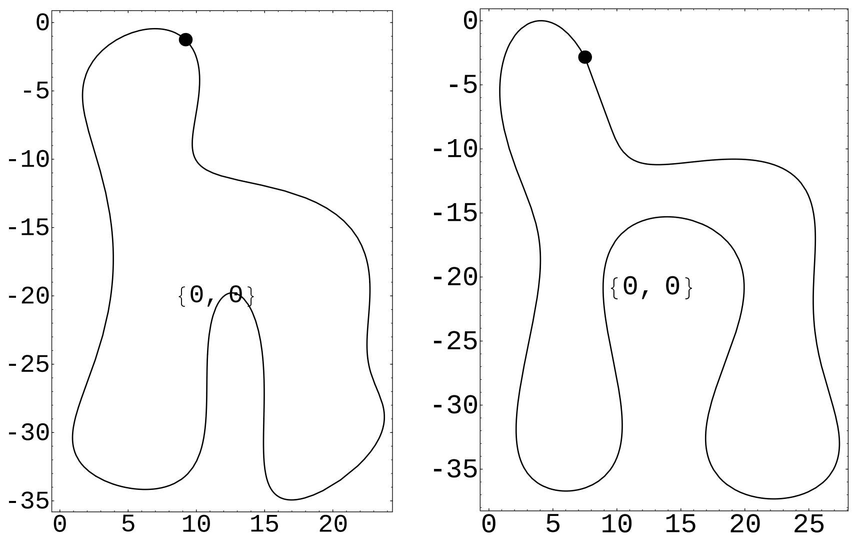

Figure 3.10. Zero-phase points on different instances of the same character class.

ignoring these components of the Fourier-transform simultaneously results in reduction of noise as well as of the dimensionality of the feature vector.

3. *space-efficiency*: Before the image is pre-processed and the high-frequency components of the curve DFTs are discarded, the features contain the same information as the original image. This is due to the fact the original bi-tonal image can be reconstructed exactly from the curves which form the boundaries of its connected components. The curves will typically require less space to represent than a bitmap of the image would. If an image has dimensions $A \times B$, then its curves will typically require $\Theta(A + B)$ space, while a bitmap of the image will occupy $\Theta(AB)$ space. This can allow a reduction in classifier size. For example, a typical approach for a neural-network based classifier is to config-

ure the network with one input per pixel. The ability to represent the same information in less space suggests that a neural-network trained using image boundaries may require fewer neurons to perform the same classification tasks as a classifier which uses one input per pixel.

3.9 Disadvantages of Features

The main disadvantages of this feature set are as follows:

1. Images which differ in only a few pixels can have large differences in feature values. Such images can even have different signatures altogether. For example, two similar samples, one containing a well-formed "O" and one containing an "O" with a small "cut" can have feature vectors that are quite different. The first image would have two boundaries and the other would have only one. This problem can be reduced if the training set contains enough examples of such errors, but given the number of possibilities it may not be possible to get enough training data to cover them all.

2. Some signatures have relatively few representative samples associated with them. Because of this such samples are difficult to classify accurately. In practice, most character images have one of only a few different signatures. Although, noisy images can have signatures that are unique which can make them hard to classify with an exemplar-based approach.

Chapter 4

EXPERIMENT & RESULTS

This chapter discusses the results obtained from evaluating three different classifiers on a test data set. The first two classifiers evaluated are the neural-network and fuzzy-knn classifiers already discussed. The third classifier is a neural-network which uses an 8×8 normalized input window of the image as its feature-set. The estimated generalization accuracy of each classifier is presented and various relationships between the errors made by the classifiers are discussed. Various methods of combining the classifier outputs to increase overall generalization accuracy are discussed as well. In addition, the performance of a method which increases overall generalization accuracy by combining the classifier outputs using weighted-voting is presented.

4.1 Training Data

The generalization accuracy of each classifier was estimated using 10-fold cross-validation. Ten pairs of training and test data sets were produced from a master data set containing character images with classes 0-9, a-z, and A-Z. The master data set contained a total of 312346 distinct images. Each training set contained 280881 images and each test set contained 31209 images. The pairs of training and test data sets were selected from the master set in such a way that the fraction of images from each character class remained as close as possible to the fraction seen in the master set.

The character samples were taken from microfilm copies of The Dallas Morning News and the Chicago Tribune newspapers. The papers were printed in the 1920s, 30s, 40s, and 70s. The character images were taken from all parts of each newspaper and so included varying sizes and styles of print. The images tended to be noisy due to degradation in the media and non-uniform lighting used during the photographic process.

4.2 Experimental Setup

This section describes the experimental configuration including the parameters used to configure the feature set as well as the specifications of each classifier trained.

4.2.1 Feature Extraction Parameters

The feature extraction parameters used are shown in Table 4.1. The vector shown in the “Smoothing Filter” row gives the kernel of the linear filter used to smooth each input curve before sampling it. A set of points, equally-spaced along the curve in terms of arc-length, are then chosen. The number of sample points chosen is shown in the “Sample Points” row of Table 4.1. A Discrete Fourier transform is then applied to the sequence of points. The number of low-frequency components saved from the

Parameter	**Value**
Sample Points	128
DFT Components	16
Smoothing Filter	$\boxed{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}}$

TABLE 4.1 Feature Extraction Parameters

transform is shown in the "DFT Components" row of Table 4.1. The rest of the transform components are discarded.

4.2.2 Fuzzy-KNN Configuration

The fuzzy-knn instances used were configured using two parameters, k and m, which are discussed in Section 3.4. The combinations of (k, m) used for the experiments are given in Equation (4.1). Each fuzzy-knn-based classifier consisted of many fuzzy-knn instances. Each of these instances used the same value of k and m. Thus, a total of 24 separate fuzzy-knn-based classifiers were constructed and tested.

$$(k, m) \,\epsilon\, \{1, 3, 5, 7, 9, 11\} \times \{1.2, 1.5, 2, 2.5\} \tag{4.1}$$

4.2.3 Neural-Network Configuration

Each neural-network-based classifier consisted of many neural networks. The architectures of these neural-networks were determined as follows:

1. input count: Each neural-network was designed to handle images having a certain fixed signature. Each network was given a number of input nodes which was equal to the feature vector length corresponding to its signature.

2. output node count: Each network had a single output node for each character class seen in the training data among images having its particular signature.

3. hidden node count: The number of hidden nodes in each network was chosen to be equal to the average of the number of input and output nodes.

Each layer of the network was fully connected to the immediately preceding layer, with the initial link weights being chosen randomly. The training algorithm used was simple stochastic back-propagation using only first-order methods without momentum. The input training data, taken from images having a given signature, was statistically standardized to $\mu = 0$, $\sigma^2 = 1$ before the network was trained on it. Details of the methods used are given in Section 3.2.

4.3 Training Result

One instance of each of the three classifiers was trained on each of the ten pairs of training and test data sets. The generalization accuracy of each classifier was then estimated as the average of its accuracy over the ten test data sets. The errors made by each classifier in each of the test sets were recorded and compared to those made by the other classifiers on the same test set.

The number of distinct image signatures which resulted from the processed images in each training set ranged from 127 to 136 with the mean value being 129.8. Thus, each of the ten fuzzy-knn-based classifiers consisted of roughly this number of fuzzy-knn instances and each of the ten neural-network-based classifiers consisted of roughly this number of neural networks.

4.3.1 Accuracy

The generalization accuracies of a series of instantiations of the fuzzy-knn classifier are shown in Table 4.27. This classifier has parameters k and m, discussed in Section 3.4, which can be adjusted and which influence its generalization accuracy. The best raw fuzzy-knn accuracy is 0.9468, obtained with $k = 5$ and $m = 1.5$. This table also shows the results of a weighted voting approach which is discussed in Section 4.4. The generalization accuracies of the two neural-network classifiers are shown in Table 4.28.

4.3.2 Error Correlation

Tables 4.3 through 4.26 show various measures of the error correlation between the fuzzy-knn and the first neural-network as well as between the fuzzy-knn and the second neural-network. Table 4.29 shows the same correlation measures between the two neural-networks. The terms used in the tables are as follows:

1. $\mathbf{knn_{k,m}}$: the fuzzy-knn-based classifier, instantiated with parameters k and m; The parameters k and m are discussed in Section 3.4.

2. **nn**: the first neural-network-based classifier, which uses the feature set developed in this paper

3. **nn2**: the second neural-network-based classifier, which uses a feature set described in the introduction to Chapter 4

4. $\mathbf{KNN_{k,m}}$: the set of images which are correctly classified by the $knn_{k,m}$ classifier

5. **NN**: the set of images which are correctly classified by the nn classifier

6. **NN2**: the set of images which are correctly classified by the $nn2$ classifier

7. $\mathbf{P}\,(imageSet)$: the size of $imageSet$ as a fraction of the test-set size

8. **equal** $(classifier1, classifier2)$: the set of images for which $classifier1$ and $classifier2$ both produce the same output

9. **unequal** $(classifier1, classifier2)$: the complement of $equal\,(classifier1, classifier2)$; the set of images for which $classifier1$ and $classifier2$ either produce different output, or for which at least one of the classifiers produces no output

10. **X**: In Tables 4.3 through 4.26 this symbol designates **NN** for the values shown in the "NN" column and it designates **NN2** for the values shown in the "NN2" column.

11. **YuleQ**: Yule's Q statistic [8]. If A and B are sets then Yule's Q statistic gives a measure of correlation between them which is shown in Equation (2.7).

4.4 Weighted Voting

The classifiers tested in this project were voted together to achieve an overall improvement in accuracy. The vote of each classifier was given a weight proportional to its overall accuracy. Table 4.27 shows the accuracy of a classifier obtained by performing a weighted-vote of the *knn*, *nn*, and *nn*2 classifier outputs and then choosing the character class with the most votes. The best weighted-voting accuracy was 0.9536, obtained with $k = 5$ and $m = 1.2$. As shown in the "Fraction Reduced" column of Table 4.27, this represents an error reduction of 15.4 % as compared to the raw fuzzy-knn accuracy using the same parameters and corresponds to an error reduction of 12.7 % as compared to the best overall raw fuzzy-knn accuracy, obtained when $k = 5$ and $m = 1.5$.

In all cases, the accuracy of the fuzzy-knn classifier is larger than that of either neural-network. However, the neural-networks together can out-vote the fuzzy-knn if they both agree on the character class of a sample. Thus, the weighted-voting algorithm can be viewed as a method of using the two neural networks to correct errors made by the fuzzy-knn. An error made by the fuzzy-knn on a given sample will be corrected by weighted-voting exactly when the two neural-networks both correctly classify the sample. The "Errors Corrected" column of Table 4.27 gives the overall fraction of test samples for which this occurs. The errors corrected by weighted-voting can be computed from (4.2).

$$P\left(\overline{KNN} \cap NN \cap NN2\right) \tag{4.2}$$

Conversely, a sample which is correctly classified by the fuzzy-knn will be classified in error by weighted-voting precisely when the two neural networks agree on the sample's class but are both incorrect. The "Errors Introduced" column of Table 4.27 gives the overall fraction of test samples which were classified correctly by the fuzzy-knn but not by weighted-voting. The errors introduced by weighted-voting can be computed from (4.3).

$$P\left(KNN \cap \overline{NN} \cap \overline{NN2} \cap equal\left(nn, nn2\right)\right) \tag{4.3}$$

4.5 Other Techniques

Voting requires the number of available classifiers to be odd. However, given only two classifiers there exist other techniques for combining them to improve generalization accuracy. Dictionary lookup is one such technique and is discussed in this section.

4.5.1 Dictionary Lookup

Assume that classifiers a and b each produce a single class as output when presented with an image. Further, let A and B be the sets of images which are classified correctly by a and b, respectively. Further, assume that $|A| \geq |B|$. One technique

a	**b**	$P\left(\overline{A} \cap unequal\,(a,b)\right)$	$P\,(A \cap unequal\,(a,b))$
$knn_{5,1.2}$	nn	0.0291134	0.0509084
$knn_{5,1.5}$	nn	0.0263482	0.0501458
nn	$nn2$	0.0823993	0.156634

TABLE 4.2 Dictionary Lookup Error Bounds

for combining their output is as follows. Let a_i and b_i be the outputs of classifiers a and b when presented with image i. During training a table can be built which, given (a_i, b_i) for $a_i \neq b_i$, yields the most likely character class of the image. Alternately, the table can contain a weighted list of character classes for each output pair. When a new image is to be classified it is presented to both a and b. If their outputs differ then at least one of the classifiers is in error. The pair of classifier outputs are then located in the table and the result found there is reported instead of either classifier output.

This technique may or may not improve accuracy. An upper bound on the fraction of errors that can be corrected by this technique is given by $P\left(\overline{A} \cap unequal\,(a,b)\right)$. An upper bound on the fraction of errors that could be introduced by this technique is given by $P\,(A \cap unequal\,(a,b))$. These expressions can be looked up from Tables 4.3 through 4.26 and Table 4.29. Table 4.2 shows the values of these two measures taken from Tables 4.11, 4.12, and 4.29.

Measure	$\mathbf{X \rightarrow NN}$	$\mathbf{X \rightarrow NN2}$
$P\left(\overline{KNN_{1,1.2}} \cap \overline{X}\right)$	0.0381364	0.0283284
$P\left(\overline{KNN_{1,1.2}} \cap \overline{X} \cap unequal\left(knn_{1,1.2}, x\right)\right)$	0.0135057	0.0226121
$P\left(\overline{KNN_{1,1.2}} \cap \overline{X} \cap equal\left(knn_{1,1.2}, x\right)\right)$	0.0246307	0.0057163
$P\left(\overline{KNN_{1,1.2}} \cap \overline{X} \mid equal\left(knn_{1,1.2}, x\right)\right)$	0.0270116	0.00737324
$P\left(\overline{KNN_{1,1.2}} \cup \overline{X}\right)$	0.112772	0.230389
$P\left(KNN_{1,1.2} \cap unequal\left(knn_{1,1.2}, x\right)\right)$	0.0509917	0.168608
$P\left(KNN_{1,1.2} \mid unequal\left(knn_{1,1.2}, x\right)\right)$	0.578465	0.750547
$P\left(\overline{KNN_{1,1.2}} \cap unequal\left(knn_{1,1.2}, x\right)\right)$	0.0371495	0.056064
$P\left(\overline{KNN_{1,1.2}} \mid unequal\left(knn_{1,1.2}, x\right)\right)$	0.421535	0.249453
$P\left(\overline{KNN_{1,1.2}} \mid \overline{X}\right)$	0.427838	0.14386
$P\left(equal\left(knn_{1,1.2}, x\right) \mid \overline{KNN_{1,1.2}} \cap \overline{X}\right)$	0.646075	0.201917
$P\left(equal\left(knn_{1,1.2}, x\right) \mid \overline{KNN_{1,1.2}}\right)$	0.398654	0.092618
$P\left(equal\left(knn_{1,1.2}, x\right) \mid \overline{X}\right)$	0.276369	0.0290299
$P\left(X \cap unequal\left(knn_{1,1.2}, x\right)\right)$	0.0236438	0.0334519
$P\left(X \mid unequal\left(knn_{1,1.2}, x\right)\right)$	0.268304	0.148836
$P\left(\overline{X} \cap unequal\left(knn_{1,1.2}, x\right)\right)$	0.0644974	0.19122
$P\left(\overline{X} \mid unequal\left(knn_{1,1.2}, x\right)\right)$	0.731696	0.851164
$P\left(\overline{X} \mid \overline{KNN_{1,1.2}}\right)$	0.617123	0.458734
$YuleQ\left(KNN_{1,1.2}, X\right)$	0.931071	0.589137

TABLE 4.3 Error Correlation for $k = 1$, $m = 1.2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{1,1.5}} \cap \overline{X}\right)$	0.0381364	0.0283284
$P\left(\overline{KNN_{1,1.5}} \cap \overline{X} \cap unequal\left(knn_{1,1.5}, x\right)\right)$	0.0135057	0.0226121
$P\left(\overline{KNN_{1,1.5}} \cap \overline{X} \cap equal\left(knn_{1,1.5}, x\right)\right)$	0.0246307	0.0057163
$P\left(\overline{KNN_{1,1.5}} \cap \overline{X} \mid equal\left(knn_{1,1.5}, x\right)\right)$	0.0270116	0.00737324
$P\left(\overline{KNN_{1,1.5}} \cup \overline{X}\right)$	0.112772	0.230389
$P\left(KNN_{1,1.5} \cap unequal\left(knn_{1,1.5}, x\right)\right)$	0.0509917	0.168608
$P\left(KNN_{1,1.5} \mid unequal\left(knn_{1,1.5}, x\right)\right)$	0.578465	0.750547
$P\left(\overline{KNN_{1,1.5}} \cap unequal\left(knn_{1,1.5}, x\right)\right)$	0.0371495	0.056064
$P\left(\overline{KNN_{1,1.5}} \mid unequal\left(knn_{1,1.5}, x\right)\right)$	0.421535	0.249453
$P\left(\overline{KNN_{1,1.5}} \mid \overline{X}\right)$	0.427838	0.14386
$P\left(equal\left(knn_{1,1.5}, x\right) \mid \overline{KNN_{1,1.5}} \cap \overline{X}\right)$	0.646075	0.201917
$P\left(equal\left(knn_{1,1.5}, x\right) \mid \overline{KNN_{1,1.5}}\right)$	0.398654	0.092618
$P\left(equal\left(knn_{1,1.5}, x\right) \mid \overline{X}\right)$	0.276369	0.0290299
$P\left(X \cap unequal\left(knn_{1,1.5}, x\right)\right)$	0.0236438	0.0334519
$P\left(X \mid unequal\left(knn_{1,1.5}, x\right)\right)$	0.268304	0.148836
$P\left(\overline{X} \cap unequal\left(knn_{1,1.5}, x\right)\right)$	0.0644974	0.19122
$P\left(\overline{X} \mid unequal\left(knn_{1,1.5}, x\right)\right)$	0.731696	0.851164
$P\left(\overline{X} \mid \overline{KNN_{1,1.5}}\right)$	0.617123	0.458734
$YuleQ\left(KNN_{1,1.5}, X\right)$	0.931071	0.589137

TABLE 4.4 Error Correlation for $k = 1$, $m = 1.5$

Measure	$\mathbf{X \to NN}$	$\mathbf{X \to NN2}$
$P\left(\overline{KNN_{1,2}} \cap \overline{X}\right)$	0.0381364	0.0283284
$P\left(\overline{KNN_{1,2}} \cap \overline{X} \cap unequal\left(knn_{1,2}, x\right)\right)$	0.0135057	0.0226121
$P\left(\overline{KNN_{1,2}} \cap \overline{X} \cap equal\left(knn_{1,2}, x\right)\right)$	0.0246307	0.0057163
$P\left(\overline{KNN_{1,2}} \cap \overline{X} \mid equal\left(knn_{1,2}, x\right)\right)$	0.0270116	0.00737324
$P\left(\overline{KNN_{1,2}} \cup \overline{X}\right)$	0.112772	0.230389
$P\left(KNN_{1,2} \cap unequal\left(knn_{1,2}, x\right)\right)$	0.0509917	0.168608
$P\left(KNN_{1,2} \mid unequal\left(knn_{1,2}, x\right)\right)$	0.578465	0.750547
$P\left(\overline{KNN_{1,2}} \cap unequal\left(knn_{1,2}, x\right)\right)$	0.0371495	0.056064
$P\left(\overline{KNN_{1,2}} \mid unequal\left(knn_{1,2}, x\right)\right)$	0.421535	0.249453
$P\left(\overline{KNN_{1,2}} \mid \overline{X}\right)$	0.427838	0.14386
$P\left(equal\left(knn_{1,2}, x\right) \mid \overline{KNN_{1,2}} \cap \overline{X}\right)$	0.646075	0.201917
$P\left(equal\left(knn_{1,2}, x\right) \mid \overline{KNN_{1,2}}\right)$	0.398654	0.092618
$P\left(equal\left(knn_{1,2}, x\right) \mid \overline{X}\right)$	0.276369	0.0290299
$P\left(X \cap unequal\left(knn_{1,2}, x\right)\right)$	0.0236438	0.0334519
$P\left(X \mid unequal\left(knn_{1,2}, x\right)\right)$	0.268304	0.148836
$P\left(\overline{X} \cap unequal\left(knn_{1,2}, x\right)\right)$	0.0644974	0.19122
$P\left(\overline{X} \mid unequal\left(knn_{1,2}, x\right)\right)$	0.731696	0.851164
$P\left(\overline{X} \mid \overline{KNN_{1,2}}\right)$	0.617123	0.458734
$YuleQ\left(KNN_{1,2}, X\right)$	0.931071	0.589137

TABLE 4.5 Error Correlation for $k = 1$, $m = 2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{1,2.5}} \cap \overline{X}\right)$	0.0381364	0.0283284
$P\left(\overline{KNN_{1,2.5}} \cap \overline{X} \cap unequal\left(knn_{1,2.5}, x\right)\right)$	0.0135057	0.0226121
$P\left(\overline{KNN_{1,2.5}} \cap \overline{X} \cap equal\left(knn_{1,2.5}, x\right)\right)$	0.0246307	0.0057163
$P\left(\overline{KNN_{1,2.5}} \cap \overline{X} \mid equal\left(knn_{1,2.5}, x\right)\right)$	0.0270116	0.00737324
$P\left(\overline{KNN_{1,2.5}} \cup \overline{X}\right)$	0.112772	0.230389
$P\left(KNN_{1,2.5} \cap unequal\left(knn_{1,2.5}, x\right)\right)$	0.0509917	0.168608
$P\left(KNN_{1,2.5} \mid unequal\left(knn_{1,2.5}, x\right)\right)$	0.578465	0.750547
$P\left(\overline{KNN_{1,2.5}} \cap unequal\left(knn_{1,2.5}, x\right)\right)$	0.0371495	0.056064
$P\left(\overline{KNN_{1,2.5}} \mid unequal\left(knn_{1,2.5}, x\right)\right)$	0.421535	0.249453
$P\left(\overline{KNN_{1,2.5}} \mid \overline{X}\right)$	0.427838	0.14386
$P\left(equal\left(knn_{1,2.5}, x\right) \mid \overline{KNN_{1,2.5}} \cap \overline{X}\right)$	0.646075	0.201917
$P\left(equal\left(knn_{1,2.5}, x\right) \mid \overline{KNN_{1,2.5}}\right)$	0.398654	0.092618
$P\left(equal\left(knn_{1,2.5}, x\right) \mid \overline{X}\right)$	0.276369	0.0290299
$P\left(X \cap unequal\left(knn_{1,2.5}, x\right)\right)$	0.0236438	0.0334519
$P\left(X \mid unequal\left(knn_{1,2.5}, x\right)\right)$	0.268304	0.148836
$P\left(\overline{X} \cap unequal\left(knn_{1,2.5}, x\right)\right)$	0.0644974	0.19122
$P\left(\overline{X} \mid unequal\left(knn_{1,2.5}, x\right)\right)$	0.731696	0.851164
$P\left(\overline{X} \mid \overline{KNN_{1,2.5}}\right)$	0.617123	0.458734
$YuleQ\left(KNN_{1,2.5}, X\right)$	0.931071	0.589137

TABLE 4.6 Error Correlation for $k = 1$, $m = 2.5$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{3,1.2}} \cap \overline{X}\right)$	0.0381493	0.0272742
$P\left(\overline{KNN_{3,1.2}} \cap \overline{X} \cap unequal\left(knn_{3,1.2}, x\right)\right)$	0.0129161	0.0216572
$P\left(\overline{KNN_{3,1.2}} \cap \overline{X} \cap equal\left(knn_{3,1.2}, x\right)\right)$	0.0252331	0.00561697
$P\left(\overline{KNN_{3,1.2}} \cap \overline{X} \mid equal\left(knn_{3,1.2}, x\right)\right)$	0.0275116	0.0072119
$P\left(\overline{KNN_{3,1.2}} \cup \overline{X}\right)$	0.108049	0.226733
$P\left(KNN_{3,1.2} \cap unequal\left(knn_{3,1.2}, x\right)\right)$	0.0509789	0.169663
$P\left(KNN_{3,1.2} \mid unequal\left(knn_{3,1.2}, x\right)\right)$	0.615566	0.767411
$P\left(\overline{KNN_{3,1.2}} \cap unequal\left(knn_{3,1.2}, x\right)\right)$	0.031837	0.0514531
$P\left(\overline{KNN_{3,1.2}} \mid unequal\left(knn_{3,1.2}, x\right)\right)$	0.384434	0.232589
$P\left(\overline{KNN_{3,1.2}} \mid \overline{X}\right)$	0.427982	0.138502
$P\left(equal\left(knn_{3,1.2}, x\right) \mid \overline{KNN_{3,1.2}} \cap \overline{X}\right)$	0.661531	0.206228
$P\left(equal\left(knn_{3,1.2}, x\right) \mid \overline{KNN_{3,1.2}}\right)$	0.442239	0.0986437
$P\left(equal\left(knn_{3,1.2}, x\right) \mid \overline{X}\right)$	0.283144	0.0285238
$P\left(X \cap unequal\left(knn_{3,1.2}, x\right)\right)$	0.0189208	0.0297959
$P\left(X \mid unequal\left(knn_{3,1.2}, x\right)\right)$	0.228498	0.134683
$P\left(\overline{X} \cap unequal\left(knn_{3,1.2}, x\right)\right)$	0.063895	0.19132
$P\left(\overline{X} \mid unequal\left(knn_{3,1.2}, x\right)\right)$	0.771502	0.865317
$P\left(\overline{X} \mid \overline{KNN_{3,1.2}}\right)$	0.668573	0.478303
$YuleQ\left(KNN_{3,1.2}, X\right)$	0.944873	0.613736

TABLE 4.7 Error Correlation for $k = 3$, $m = 1.2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{3,1.5}} \cap \overline{X}\right)$	0.0385306	0.0268801
$P\left(\overline{KNN_{3,1.5}} \cap \overline{X} \cap unequal\left(knn_{3,1.5}, x\right)\right)$	0.0127367	0.0212279
$P\left(\overline{KNN_{3,1.5}} \cap \overline{X} \cap equal\left(knn_{3,1.5}, x\right)\right)$	0.0257938	0.00565222
$P\left(\overline{KNN_{3,1.5}} \cap \overline{X} \mid equal\left(knn_{3,1.5}, x\right)\right)$	0.0280226	0.00723873
$P\left(\overline{KNN_{3,1.5}} \cup \overline{X}\right)$	0.105319	0.224778
$P\left(KNN_{3,1.5} \cap unequal\left(knn_{3,1.5}, x\right)\right)$	0.0505976	0.170057
$P\left(KNN_{3,1.5} \mid unequal\left(knn_{3,1.5}, x\right)\right)$	0.636305	0.776192
$P\left(\overline{KNN_{3,1.5}} \cap unequal\left(knn_{3,1.5}, x\right)\right)$	0.0289276	0.0490692
$P\left(\overline{KNN_{3,1.5}} \mid unequal\left(knn_{3,1.5}, x\right)\right)$	0.363695	0.223808
$P\left(\overline{KNN_{3,1.5}} \mid \overline{X}\right)$	0.432237	0.136492
$P\left(equal\left(knn_{3,1.5}, x\right) \mid \overline{KNN_{3,1.5}} \cap \overline{X}\right)$	0.669689	0.210657
$P\left(equal\left(knn_{3,1.5}, x\right) \mid \overline{KNN_{3,1.5}}\right)$	0.471682	0.103575
$P\left(equal\left(knn_{3,1.5}, x\right) \mid \overline{X}\right)$	0.289418	0.0287008
$P\left(X \cap unequal\left(knn_{3,1.5}, x\right)\right)$	0.0161908	0.0278413
$P\left(X \mid unequal\left(knn_{3,1.5}, x\right)\right)$	0.203566	0.126984
$P\left(\overline{X} \cap unequal\left(knn_{3,1.5}, x\right)\right)$	0.0633343	0.191285
$P\left(\overline{X} \mid unequal\left(knn_{3,1.5}, x\right)\right)$	0.796434	0.873016
$P\left(\overline{X} \mid \overline{KNN_{3,1.5}}\right)$	0.704359	0.491591
$YuleQ\left(KNN_{3,1.5}, X\right)$	0.953603	0.630109

TABLE 4.8 Error Correlation for $k = 3$, $m = 1.5$

Measure	$\mathbf{X \rightarrow NN}$	$\mathbf{X \rightarrow NN2}$
$P\left(\overline{KNN_{3,2}} \cap \overline{X}\right)$	0.0389567	0.0270467
$P\left(\overline{KNN_{3,2}} \cap \overline{X} \cap unequal\left(knn_{3,2}, x\right)\right)$	0.0127431	0.0213432
$P\left(\overline{KNN_{3,2}} \cap \overline{X} \cap equal\left(knn_{3,2}, x\right)\right)$	0.0262136	0.00570348
$P\left(\overline{KNN_{3,2}} \cap \overline{X} \mid equal\left(knn_{3,2}, x\right)\right)$	0.0284462	0.0073003
$P\left(\overline{KNN_{3,2}} \cup \overline{X}\right)$	0.104688	0.224406
$P\left(KNN_{3,2} \cap unequal\left(knn_{3,2}, x\right)\right)$	0.0501714	0.16989
$P\left(KNN_{3,2} \mid unequal\left(knn_{3,2}, x\right)\right)$	0.639465	0.776947
$P\left(\overline{KNN_{3,2}} \cap unequal\left(knn_{3,2}, x\right)\right)$	0.0283027	0.0488128
$P\left(\overline{KNN_{3,2}} \mid unequal\left(knn_{3,2}, x\right)\right)$	0.360535	0.223053
$P\left(\overline{KNN_{3,2}} \mid \overline{X}\right)$	0.437016	0.13734
$P\left(equal\left(knn_{3,2}, x\right) \mid \overline{KNN_{3,2}} \cap \overline{X}\right)$	0.673056	0.21137
$P\left(equal\left(knn_{3,2}, x\right) \mid \overline{KNN_{3,2}}\right)$	0.481258	0.10498
$P\left(equal\left(knn_{3,2}, x\right) \mid \overline{X}\right)$	0.294137	0.0289608
$P\left(X \cap unequal\left(knn_{3,2}, x\right)\right)$	0.0155596	0.0274696
$P\left(X \mid unequal\left(knn_{3,2}, x\right)\right)$	0.198229	0.125523
$P\left(\overline{X} \cap unequal\left(knn_{3,2}, x\right)\right)$	0.0629145	0.191233
$P\left(\overline{X} \mid unequal\left(knn_{3,2}, x\right)\right)$	0.801771	0.874477
$P\left(\overline{X} \mid \overline{KNN_{3,2}}\right)$	0.715013	0.496445
$YuleQ\left(KNN_{3,2}, X\right)$	0.956314	0.636367

TABLE 4.9 Error Correlation for $k = 3$, $m = 2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{3,2.5}} \cap \overline{X}\right)$	0.039354	0.0272165
$P\left(\overline{KNN_{3,2.5}} \cap \overline{X} \cap unequal\left(knn_{3,2.5}, x\right)\right)$	0.0128617	0.0215034
$P\left(\overline{KNN_{3,2.5}} \cap \overline{X} \cap equal\left(knn_{3,2.5}, x\right)\right)$	0.0264924	0.0057131
$P\left(\overline{KNN_{3,2.5}} \cap \overline{X} \mid equal\left(knn_{3,2.5}, x\right)\right)$	0.0287356	0.0073134
$P\left(\overline{KNN_{3,2.5}} \cup \overline{X}\right)$	0.10455	0.224496
$P\left(KNN_{3,2.5} \cap unequal\left(knn_{3,2.5}, x\right)\right)$	0.0497741	0.16972
$P\left(KNN_{3,2.5} \mid unequal\left(knn_{3,2.5}, x\right)\right)$	0.637754	0.775891
$P\left(\overline{KNN_{3,2.5}} \cap unequal\left(knn_{3,2.5}, x\right)\right)$	0.0282835	0.0490628
$P\left(\overline{KNN_{3,2.5}} \mid unequal\left(knn_{3,2.5}, x\right)\right)$	0.362246	0.224109
$P\left(\overline{KNN_{3,2.5}} \mid \overline{X}\right)$	0.441473	0.138203
$P\left(equal\left(knn_{3,2.5}, x\right) \mid \overline{KNN_{3,2.5}} \cap \overline{X}\right)$	0.673384	0.210336
$P\left(equal\left(knn_{3,2.5}, x\right) \mid \overline{KNN_{3,2.5}}\right)$	0.484054	0.104647
$P\left(equal\left(knn_{3,2.5}, x\right) \mid \overline{X}\right)$	0.297271	0.02901
$P\left(X \cap unequal\left(knn_{3,2.5}, x\right)\right)$	0.0154218	0.0275594
$P\left(X \mid unequal\left(knn_{3,2.5}, x\right)\right)$	0.197555	0.125882
$P\left(\overline{X} \cap unequal\left(knn_{3,2.5}, x\right)\right)$	0.0626358	0.191224
$P\left(\overline{X} \mid unequal\left(knn_{3,2.5}, x\right)\right)$	0.802445	0.874118
$P\left(\overline{X} \mid \overline{KNN_{3,2.5}}\right)$	0.71884	0.497267
$YuleQ\left(KNN_{3,2.5}, X\right)$	0.957457	0.637608

TABLE 4.10 Error Correlation for $k = 3$, $m = 2.5$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{5,1.2}} \cap \overline{X}\right)$	0.0382197	0.0267807
$P\left(\overline{KNN_{5,1.2}} \cap \overline{X} \cap unequal\left(knn_{5,1.2}, x\right)\right)$	0.0124772	0.0212118
$P\left(\overline{KNN_{5,1.2}} \cap \overline{X} \cap equal\left(knn_{5,1.2}, x\right)\right)$	0.0257426	0.00556891
$P\left(\overline{KNN_{5,1.2}} \cap \overline{X} \mid equal\left(knn_{5,1.2}, x\right)\right)$	0.0279818	0.00713481
$P\left(\overline{KNN_{5,1.2}} \cup \overline{X}\right)$	0.105764	0.225012
$P\left(KNN_{5,1.2} \cap unequal\left(knn_{5,1.2}, x\right)\right)$	0.0509084	0.170156
$P\left(KNN_{5,1.2} \mid unequal\left(knn_{5,1.2}, x\right)\right)$	0.636175	0.775489
$P\left(\overline{KNN_{5,1.2}} \cap unequal\left(knn_{5,1.2}, x\right)\right)$	0.0291134	0.0492871
$P\left(\overline{KNN_{5,1.2}} \mid unequal\left(knn_{5,1.2}, x\right)\right)$	0.363825	0.224511
$P\left(\overline{KNN_{5,1.2}} \mid \overline{X}\right)$	0.428775	0.135994
$P\left(equal\left(knn_{5,1.2}, x\right) \mid \overline{KNN_{5,1.2}} \cap \overline{X}\right)$	0.673705	0.208185
$P\left(equal\left(knn_{5,1.2}, x\right) \mid \overline{KNN_{5,1.2}}\right)$	0.469462	0.101682
$P\left(equal\left(knn_{5,1.2}, x\right) \mid \overline{X}\right)$	0.288846	0.028282
$P\left(X \cap unequal\left(knn_{5,1.2}, x\right)\right)$	0.0166362	0.0280752
$P\left(X \mid unequal\left(knn_{5,1.2}, x\right)\right)$	0.207906	0.127886
$P\left(\overline{X} \cap unequal\left(knn_{5,1.2}, x\right)\right)$	0.0633856	0.191368
$P\left(\overline{X} \mid unequal\left(knn_{5,1.2}, x\right)\right)$	0.792094	0.872114
$P\left(\overline{X} \mid \overline{KNN_{5,1.2}}\right)$	0.696886	0.488518
$YuleQ\left(KNN_{5,1.2}, X\right)$	0.951643	0.626111

TABLE 4.11 Error Correlation for $k = 5$, $m = 1.2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{5,1.5}} \cap \overline{X}\right)$	0.0389823	0.0265725
$P\left(\overline{KNN_{5,1.5}} \cap \overline{X} \cap unequal\left(knn_{5,1.5}, x\right)\right)$	0.0122272	0.020901
$P\left(\overline{KNN_{5,1.5}} \cap \overline{X} \cap equal\left(knn_{5,1.5}, x\right)\right)$	0.0267551	0.00567144
$P\left(\overline{KNN_{5,1.5}} \cap \overline{X} \mid equal\left(knn_{5,1.5}, x\right)\right)$	0.0289711	0.00725074
$P\left(\overline{KNN_{5,1.5}} \cup \overline{X}\right)$	0.103249	0.223468
$P\left(KNN_{5,1.5} \cap unequal\left(knn_{5,1.5}, x\right)\right)$	0.0501458	0.170364
$P\left(KNN_{5,1.5} \mid unequal\left(knn_{5,1.5}, x\right)\right)$	0.655565	0.782328
$P\left(\overline{KNN_{5,1.5}} \cap unequal\left(knn_{5,1.5}, x\right)\right)$	0.0263482	0.0474318
$P\left(\overline{KNN_{5,1.5}} \mid unequal\left(knn_{5,1.5}, x\right)\right)$	0.344435	0.217672
$P\left(\overline{KNN_{5,1.5}} \mid \overline{X}\right)$	0.437297	0.134935
$P\left(equal\left(knn_{5,1.5}, x\right) \mid \overline{KNN_{5,1.5}} \cap \overline{X}\right)$	0.686594	0.213779
$P\left(equal\left(knn_{5,1.5}, x\right) \mid \overline{KNN_{5,1.5}}\right)$	0.504158	0.107097
$P\left(equal\left(knn_{5,1.5}, x\right) \mid \overline{X}\right)$	0.300197	0.0288009
$P\left(X \cap unequal\left(knn_{5,1.5}, x\right)\right)$	0.0141209	0.0265308
$P\left(X \mid unequal\left(knn_{5,1.5}, x\right)\right)$	0.184614	0.12175
$P\left(\overline{X} \cap unequal\left(knn_{5,1.5}, x\right)\right)$	0.062373	0.191265
$P\left(\overline{X} \mid unequal\left(knn_{5,1.5}, x\right)\right)$	0.815386	0.87825
$P\left(\overline{X} \mid \overline{KNN_{5,1.5}}\right)$	0.734288	0.500764
$YuleQ\left(KNN_{5,1.5}, X\right)$	0.960318	0.640963

TABLE 4.12 Error Correlation for $k = 5$, $m = 1.5$

Measure	$\mathbf{X \rightarrow NN}$	$\mathbf{X \rightarrow NN2}$
$P\left(\overline{KNN_{5,2}} \cap \overline{X}\right)$	0.0400525	0.0271652
$P\left(\overline{KNN_{5,2}} \cap \overline{X} \cap unequal\left(knn_{5,2}, x\right)\right)$	0.0124323	0.0214425
$P\left(\overline{KNN_{5,2}} \cap \overline{X} \cap equal\left(knn_{5,2}, x\right)\right)$	0.0276202	0.00572271
$P\left(\overline{KNN_{5,2}} \cap \overline{X} \mid equal\left(knn_{5,2}, x\right)\right)$	0.0298751	0.00731906
$P\left(\overline{KNN_{5,2}} \cup \overline{X}\right)$	0.103111	0.223807
$P\left(KNN_{5,2} \cap unequal\left(knn_{5,2}, x\right)\right)$	0.0490756	0.169772
$P\left(KNN_{5,2} \mid unequal\left(knn_{5,2}, x\right)\right)$	0.650075	0.778591
$P\left(\overline{KNN_{5,2}} \cap unequal\left(knn_{5,2}, x\right)\right)$	0.0264155	0.048313
$P\left(\overline{KNN_{5,2}} \mid unequal\left(knn_{5,2}, x\right)\right)$	0.349925	0.221409
$P\left(\overline{KNN_{5,2}} \mid \overline{X}\right)$	0.449315	0.137948
$P\left(equal\left(knn_{5,2}, x\right) \mid \overline{KNN_{5,2}} \cap \overline{X}\right)$	0.689824	0.211101
$P\left(equal\left(knn_{5,2}, x\right) \mid \overline{KNN_{5,2}}\right)$	0.511518	0.106243
$P\left(equal\left(knn_{5,2}, x\right) \mid \overline{X}\right)$	0.309918	0.029061
$P\left(X \cap unequal\left(knn_{5,2}, x\right)\right)$	0.0139831	0.0268705
$P\left(X \mid unequal\left(knn_{5,2}, x\right)\right)$	0.185277	0.123138
$P\left(\overline{X} \cap unequal\left(knn_{5,2}, x\right)\right)$	0.0615079	0.191214
$P\left(\overline{X} \mid unequal\left(knn_{5,2}, x\right)\right)$	0.814723	0.876862
$P\left(\overline{X} \mid \overline{KNN_{5,2}}\right)$	0.74149	0.503069
$YuleQ\left(KNN_{5,2}, X\right)$	0.962567	0.644566

TABLE 4.13 Error Correlation for $k = 5$, $m = 2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{5,2.5}} \cap \overline{X}\right)$	0.0407959	0.0275529
$P\left(\overline{KNN_{5,2.5}} \cap \overline{X} \cap unequal\left(knn_{5,2.5}, x\right)\right)$	0.0127015	0.0218142
$P\left(\overline{KNN_{5,2.5}} \cap \overline{X} \cap equal\left(knn_{5,2.5}, x\right)\right)$	0.0280945	0.00573873
$P\left(\overline{KNN_{5,2.5}} \cap \overline{X} \mid equal\left(knn_{5,2.5}, x\right)\right)$	0.0303769	0.00734389
$P\left(\overline{KNN_{5,2.5}} \cup \overline{X}\right)$	0.103239	0.224291
$P\left(KNN_{5,2.5} \cap unequal\left(knn_{5,2.5}, x\right)\right)$	0.0483322	0.169384
$P\left(KNN_{5,2.5} \mid unequal\left(knn_{5,2.5}, x\right)\right)$	0.64319	0.77516
$P\left(\overline{KNN_{5,2.5}} \cap unequal\left(knn_{5,2.5}, x\right)\right)$	0.0268128	0.0491685
$P\left(\overline{KNN_{5,2.5}} \mid unequal\left(knn_{5,2.5}, x\right)\right)$	0.35681	0.22484
$P\left(\overline{KNN_{5,2.5}} \mid \overline{X}\right)$	0.457651	0.139917
$P\left(equal\left(knn_{5,2.5}, x\right) \mid \overline{KNN_{5,2.5}} \cap \overline{X}\right)$	0.688986	0.208758
$P\left(equal\left(knn_{5,2.5}, x\right) \mid \overline{KNN_{5,2.5}}\right)$	0.51214	0.104893
$P\left(equal\left(knn_{5,2.5}, x\right) \mid \overline{X}\right)$	0.315235	0.0291414
$P\left(X \cap unequal\left(knn_{5,2.5}, x\right)\right)$	0.0141113	0.0273543
$P\left(X \mid unequal\left(knn_{5,2.5}, x\right)\right)$	0.187832	0.125083
$P\left(\overline{X} \cap unequal\left(knn_{5,2.5}, x\right)\right)$	0.0610337	0.191198
$P\left(\overline{X} \mid unequal\left(knn_{5,2.5}, x\right)\right)$	0.812168	0.874917
$P\left(\overline{X} \mid \overline{KNN_{5,2.5}}\right)$	0.743292	0.502171
$YuleQ\left(KNN_{5,2.5}, X\right)$	0.963469	0.643979

TABLE 4.14 Error Correlation for $k = 5$, $m = 2.5$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{7,1.2}} \cap \overline{X}\right)$	0.0385017	0.0266718
$P\left(\overline{KNN_{7,1.2}} \cap \overline{X} \cap unequal\left(knn_{7,1.2}, x\right)\right)$	0.0123618	0.021074
$P\left(\overline{KNN_{7,1.2}} \cap \overline{X} \cap equal\left(knn_{7,1.2}, x\right)\right)$	0.0261399	0.00559774
$P\left(\overline{KNN_{7,1.2}} \cap \overline{X} \mid equal\left(knn_{7,1.2}, x\right)\right)$	0.0283716	0.00716624
$P\left(\overline{KNN_{7,1.2}} \cup \overline{X}\right)$	0.104806	0.224445
$P\left(KNN_{7,1.2} \cap unequal\left(knn_{7,1.2}, x\right)\right)$	0.0506264	0.170265
$P\left(KNN_{7,1.2} \mid unequal\left(knn_{7,1.2}, x\right)\right)$	0.643518	0.778105
$P\left(\overline{KNN_{7,1.2}} \cap unequal\left(knn_{7,1.2}, x\right)\right)$	0.02804	0.0485821
$P\left(\overline{KNN_{7,1.2}} \mid unequal\left(knn_{7,1.2}, x\right)\right)$	0.356482	0.221895
$P\left(\overline{KNN_{7,1.2}} \mid \overline{X}\right)$	0.431944	0.135438
$P\left(equal\left(knn_{7,1.2}, x\right) \mid \overline{KNN_{7,1.2}} \cap \overline{X}\right)$	0.679114	0.210039
$P\left(equal\left(knn_{7,1.2}, x\right) \mid \overline{KNN_{7,1.2}}\right)$	0.482658	0.103506
$P\left(equal\left(knn_{7,1.2}, x\right) \mid \overline{X}\right)$	0.293296	0.0284273
$P\left(X \cap unequal\left(knn_{7,1.2}, x\right)\right)$	0.0156782	0.0275081
$P\left(X \mid unequal\left(knn_{7,1.2}, x\right)\right)$	0.199302	0.125636
$P\left(\overline{X} \cap unequal\left(knn_{7,1.2}, x\right)\right)$	0.0629882	0.191339
$P\left(\overline{X} \mid unequal\left(knn_{7,1.2}, x\right)\right)$	0.800698	0.874364
$P\left(\overline{X} \mid \overline{KNN_{7,1.2}}\right)$	0.710738	0.492697
$YuleQ\left(KNN_{7,1.2}, X\right)$	0.954953	0.631176

TABLE 4.15 Error Correlation for $k = 7$, $m = 1.2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{7,1.5}} \cap \overline{X}\right)$	0.0396905	0.0268256
$P\left(\overline{KNN_{7,1.5}} \cap \overline{X} \cap unequal\left(knn_{7,1.5}, x\right)\right)$	0.0120542	0.021074
$P\left(\overline{KNN_{7,1.5}} \cap \overline{X} \cap equal\left(knn_{7,1.5}, x\right)\right)$	0.0276363	0.00575155
$P\left(\overline{KNN_{7,1.5}} \cap \overline{X} \mid equal\left(knn_{7,1.5}, x\right)\right)$	0.0298791	0.00735168
$P\left(\overline{KNN_{7,1.5}} \cup \overline{X}\right)$	0.102717	0.223391
$P\left(KNN_{7,1.5} \cap unequal\left(knn_{7,1.5}, x\right)\right)$	0.0494377	0.170111
$P\left(KNN_{7,1.5} \mid unequal\left(knn_{7,1.5}, x\right)\right)$	0.658385	0.781746
$P\left(\overline{KNN_{7,1.5}} \cap unequal\left(knn_{7,1.5}, x\right)\right)$	0.0256432	0.047528
$P\left(\overline{KNN_{7,1.5}} \mid unequal\left(knn_{7,1.5}, x\right)\right)$	0.341615	0.218254
$P\left(\overline{KNN_{7,1.5}} \mid \overline{X}\right)$	0.445257	0.136219
$P\left(equal\left(knn_{7,1.5}, x\right) \mid \overline{KNN_{7,1.5}} \cap \overline{X}\right)$	0.696482	0.214721
$P\left(equal\left(knn_{7,1.5}, x\right) \mid \overline{KNN_{7,1.5}}\right)$	0.519	0.108277
$P\left(equal\left(knn_{7,1.5}, x\right) \mid \overline{X}\right)$	0.310087	0.029208
$P\left(X \cap unequal\left(knn_{7,1.5}, x\right)\right)$	0.013589	0.0264539
$P\left(X \mid unequal\left(knn_{7,1.5}, x\right)\right)$	0.181051	0.121474
$P\left(\overline{X} \cap unequal\left(knn_{7,1.5}, x\right)\right)$	0.0614919	0.191185
$P\left(\overline{X} \mid unequal\left(knn_{7,1.5}, x\right)\right)$	0.818949	0.878526
$P\left(\overline{X} \mid \overline{KNN_{7,1.5}}\right)$	0.745137	0.503938
$YuleQ\left(KNN_{7,1.5}, X\right)$	0.963004	0.645145

TABLE 4.16 Error Correlation for $k = 7$, $m = 1.5$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{7,2}} \cap \overline{X}\right)$	0.041331	0.0277452
$P\left(\overline{KNN_{7,2}} \cap \overline{X} \cap unequal\left(knn_{7,2}, x\right)\right)$	0.0126085	0.0219552
$P\left(\overline{KNN_{7,2}} \cap \overline{X} \cap equal\left(knn_{7,2}, x\right)\right)$	0.0287225	0.00579
$P\left(\overline{KNN_{7,2}} \cap \overline{X} \mid equal\left(knn_{7,2}, x\right)\right)$	0.0310188	0.00740779
$P\left(\overline{KNN_{7,2}} \cup \overline{X}\right)$	0.102765	0.22416
$P\left(KNN_{7,2} \cap unequal\left(knn_{7,2}, x\right)\right)$	0.0477971	0.169192
$P\left(KNN_{7,2} \mid unequal\left(knn_{7,2}, x\right)\right)$	0.645506	0.774941
$P\left(\overline{KNN_{7,2}} \cap unequal\left(knn_{7,2}, x\right)\right)$	0.0262456	0.0491781
$P\left(\overline{KNN_{7,2}} \mid unequal\left(knn_{7,2}, x\right)\right)$	0.354494	0.225059
$P\left(\overline{KNN_{7,2}} \mid \overline{X}\right)$	0.463669	0.14089
$P\left(equal\left(knn_{7,2}, x\right) \mid \overline{KNN_{7,2}} \cap \overline{X}\right)$	0.695115	0.209137
$P\left(equal\left(knn_{7,2}, x\right) \mid \overline{KNN_{7,2}}\right)$	0.522976	0.105744
$P\left(equal\left(knn_{7,2}, x\right) \mid \overline{X}\right)$	0.322283	0.0294016
$P\left(X \cap unequal\left(knn_{7,2}, x\right)\right)$	0.0136371	0.0272229
$P\left(X \mid unequal\left(knn_{7,2}, x\right)\right)$	0.184222	0.124577
$P\left(\overline{X} \cap unequal\left(knn_{7,2}, x\right)\right)$	0.0604057	0.191147
$P\left(\overline{X} \mid unequal\left(knn_{7,2}, x\right)\right)$	0.815778	0.875423
$P\left(\overline{X} \mid \overline{KNN_{7,2}}\right)$	0.752292	0.505233
$YuleQ\left(KNN_{7,2}, X\right)$	0.96554	0.647934

TABLE 4.17 Error Correlation for $k = 7$, $m = 2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{7,2.5}} \cap \overline{X}\right)$	0.0421064	0.028197
$P\left(\overline{KNN_{7,2.5}} \cap \overline{X} \cap unequal\left(knn_{7,2.5}, x\right)\right)$	0.0129674	0.0223461
$P\left(\overline{KNN_{7,2.5}} \cap \overline{X} \cap equal\left(knn_{7,2.5}, x\right)\right)$	0.029139	0.00585088
$P\left(\overline{KNN_{7,2.5}} \cap \overline{X} \mid equal\left(knn_{7,2.5}, x\right)\right)$	0.0314648	0.00749097
$P\left(\overline{KNN_{7,2.5}} \cup \overline{X}\right)$	0.10306	0.224778
$P\left(KNN_{7,2.5} \cap unequal\left(knn_{7,2.5}, x\right)\right)$	0.0470217	0.16874
$P\left(KNN_{7,2.5} \mid unequal\left(knn_{7,2.5}, x\right)\right)$	0.636136	0.770921
$P\left(\overline{KNN_{7,2.5}} \cap unequal\left(knn_{7,2.5}, x\right)\right)$	0.0268993	0.0501874
$P\left(\overline{KNN_{7,2.5}} \mid unequal\left(knn_{7,2.5}, x\right)\right)$	0.363864	0.229079
$P\left(\overline{KNN_{7,2.5}} \mid \overline{X}\right)$	0.472365	0.143182
$P\left(equal\left(knn_{7,2.5}, x\right) \mid \overline{KNN_{7,2.5}} \cap \overline{X}\right)$	0.69233	0.207973
$P\left(equal\left(knn_{7,2.5}, x\right) \mid \overline{KNN_{7,2.5}}\right)$	0.520572	0.104861
$P\left(equal\left(knn_{7,2.5}, x\right) \mid \overline{X}\right)$	0.32696	0.0297102
$P\left(X \cap unequal\left(knn_{7,2.5}, x\right)\right)$	0.0139319	0.0278413
$P\left(X \mid unequal\left(knn_{7,2.5}, x\right)\right)$	0.188488	0.127071
$P\left(\overline{X} \cap unequal\left(knn_{7,2.5}, x\right)\right)$	0.0599891	0.191086
$P\left(\overline{X} \mid unequal\left(knn_{7,2.5}, x\right)\right)$	0.811512	0.872929
$P\left(\overline{X} \mid \overline{KNN_{7,2.5}}\right)$	0.751835	0.50376
$YuleQ\left(KNN_{7,2.5}, X\right)$	0.966007	0.646686

TABLE 4.18 Error Correlation for $k = 7$, $m = 2.5$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{9,1.2}} \cap \overline{X}\right)$	0.0387356	0.0266398
$P\left(\overline{KNN_{9,1.2}} \cap \overline{X} \cap unequal\left(knn_{9,1.2}, x\right)\right)$	0.0122625	0.0210452
$P\left(\overline{KNN_{9,1.2}} \cap \overline{X} \cap equal\left(knn_{9,1.2}, x\right)\right)$	0.0264731	0.00559454
$P\left(\overline{KNN_{9,1.2}} \cap \overline{X} \mid equal\left(knn_{9,1.2}, x\right)\right)$	0.0287088	0.0071606
$P\left(\overline{KNN_{9,1.2}} \cup \overline{X}\right)$	0.104364	0.224269
$P\left(KNN_{9,1.2} \cap unequal\left(knn_{9,1.2}, x\right)\right)$	0.0503925	0.170297
$P\left(KNN_{9,1.2} \mid unequal\left(knn_{9,1.2}, x\right)\right)$	0.646862	0.778866
$P\left(\overline{KNN_{9,1.2}} \cap unequal\left(knn_{9,1.2}, x\right)\right)$	0.0274985	0.0483771
$P\left(\overline{KNN_{9,1.2}} \mid unequal\left(knn_{9,1.2}, x\right)\right)$	0.353138	0.221134
$P\left(\overline{KNN_{9,1.2}} \mid \overline{X}\right)$	0.434569	0.135277
$P\left(equal\left(knn_{9,1.2}, x\right) \mid \overline{KNN_{9,1.2}} \cap \overline{X}\right)$	0.683577	0.210173
$P\left(equal\left(knn_{9,1.2}, x\right) \mid \overline{KNN_{9,1.2}}\right)$	0.490625	0.103841
$P\left(equal\left(knn_{9,1.2}, x\right) \mid \overline{X}\right)$	0.297027	0.0284107
$P\left(X \cap unequal\left(knn_{9,1.2}, x\right)\right)$	0.015236	0.0273319
$P\left(X \mid unequal\left(knn_{9,1.2}, x\right)\right)$	0.195624	0.124929
$P\left(\overline{X} \cap unequal\left(knn_{9,1.2}, x\right)\right)$	0.062655	0.191342
$P\left(\overline{X} \mid unequal\left(knn_{9,1.2}, x\right)\right)$	0.804376	0.875071
$P\left(\overline{X} \mid \overline{KNN_{9,1.2}}\right)$	0.717746	0.493998
$YuleQ\left(KNN_{9,1.2}, X\right)$	0.956641	0.632744

TABLE 4.19 Error Correlation for $k = 9$, $m = 1.2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{9,1.5}} \cap \overline{X}\right)$	0.0404755	0.0272486
$P\left(\overline{KNN_{9,1.5}} \cap \overline{X} \cap unequal\left(knn_{9,1.5}, x\right)\right)$	0.0121375	0.0214009
$P\left(\overline{KNN_{9,1.5}} \cap \overline{X} \cap equal\left(knn_{9,1.5}, x\right)\right)$	0.028338	0.00584767
$P\left(\overline{KNN_{9,1.5}} \cap \overline{X} \mid equal\left(knn_{9,1.5}, x\right)\right)$	0.0306142	0.00747691
$P\left(\overline{KNN_{9,1.5}} \cup \overline{X}\right)$	0.102704	0.22374
$P\left(KNN_{9,1.5} \cap unequal\left(knn_{9,1.5}, x\right)\right)$	0.0486526	0.169688
$P\left(KNN_{9,1.5} \mid unequal\left(knn_{9,1.5}, x\right)\right)$	0.654171	0.778895
$P\left(\overline{KNN_{9,1.5}} \cap unequal\left(knn_{9,1.5}, x\right)\right)$	0.0257137	0.048204
$P\left(\overline{KNN_{9,1.5}} \mid unequal\left(knn_{9,1.5}, x\right)\right)$	0.345829	0.221105
$P\left(\overline{KNN_{9,1.5}} \mid \overline{X}\right)$	0.454045	0.138367
$P\left(equal\left(knn_{9,1.5}, x\right) \mid \overline{KNN_{9,1.5}} \cap \overline{X}\right)$	0.700367	0.215016
$P\left(equal\left(knn_{9,1.5}, x\right) \mid \overline{KNN_{9,1.5}}\right)$	0.524622	0.108525
$P\left(equal\left(knn_{9,1.5}, x\right) \mid \overline{X}\right)$	0.317958	0.0296957
$P\left(X \cap unequal\left(knn_{9,1.5}, x\right)\right)$	0.0135762	0.0268032
$P\left(X \mid unequal\left(knn_{9,1.5}, x\right)\right)$	0.182628	0.122939
$P\left(\overline{X} \cap unequal\left(knn_{9,1.5}, x\right)\right)$	0.0607902	0.191089
$P\left(\overline{X} \mid unequal\left(knn_{9,1.5}, x\right)\right)$	0.817372	0.877061
$P\left(\overline{X} \mid \overline{KNN_{9,1.5}}\right)$	0.749032	0.504475
$YuleQ\left(KNN_{9,1.5}, X\right)$	0.964318	0.646339

TABLE 4.20 Error Correlation for $k = 9$, $m = 1.5$

Measure	$\mathbf{X \rightarrow NN}$	$\mathbf{X \rightarrow NN2}$
$P\left(\overline{KNN_{9,2}} \cap \overline{X}\right)$	0.0425935	0.0284309
$P\left(\overline{KNN_{9,2}} \cap \overline{X} \cap unequal\left(knn_{9,2}, x\right)\right)$	0.012977	0.0224743
$P\left(\overline{KNN_{9,2}} \cap \overline{X} \cap equal\left(knn_{9,2}, x\right)\right)$	0.0296165	0.00595662
$P\left(\overline{KNN_{9,2}} \cap \overline{X} \mid equal\left(knn_{9,2}, x\right)\right)$	0.0319644	0.00762808
$P\left(\overline{KNN_{9,2}} \cup \overline{X}\right)$	0.103082	0.225054
$P\left(KNN_{9,2} \cap unequal\left(knn_{9,2}, x\right)\right)$	0.0465347	0.168506
$P\left(KNN_{9,2} \mid unequal\left(knn_{9,2}, x\right)\right)$	0.633476	0.769242
$P\left(\overline{KNN_{9,2}} \cap unequal\left(knn_{9,2}, x\right)\right)$	0.0269313	0.0505912
$P\left(\overline{KNN_{9,2}} \mid unequal\left(knn_{9,2}, x\right)\right)$	0.366524	0.230758
$P\left(\overline{KNN_{9,2}} \mid \overline{X}\right)$	0.477828	0.14437
$P\left(equal\left(knn_{9,2}, x\right) \mid \overline{KNN_{9,2}} \cap \overline{X}\right)$	0.695603	0.209983
$P\left(equal\left(knn_{9,2}, x\right) \mid \overline{KNN_{9,2}}\right)$	0.524335	0.105726
$P\left(equal\left(knn_{9,2}, x\right) \mid \overline{X}\right)$	0.332328	0.0302471
$P\left(X \cap unequal\left(knn_{9,2}, x\right)\right)$	0.0139543	0.0281169
$P\left(X \mid unequal\left(knn_{9,2}, x\right)\right)$	0.189935	0.128241
$P\left(\overline{X} \cap unequal\left(knn_{9,2}, x\right)\right)$	0.0595117	0.19098
$P\left(\overline{X} \mid unequal\left(knn_{9,2}, x\right)\right)$	0.810065	0.871759
$P\left(\overline{X} \mid \overline{KNN_{9,2}}\right)$	0.753677	0.5032
$YuleQ\left(KNN_{9,2}, X\right)$	0.966663	0.646337

TABLE 4.21 Error Correlation for $k = 9$, $m = 2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{9,2.5}} \cap \overline{X}\right)$	0.0435163	0.0289756
$P\left(\overline{KNN_{9,2.5}} \cap \overline{X} \cap unequal\left(knn_{9,2.5}, x\right)\right)$	0.0133679	0.0229645
$P\left(\overline{KNN_{9,2.5}} \cap \overline{X} \cap equal\left(knn_{9,2.5}, x\right)\right)$	0.0301484	0.00601109
$P\left(\overline{KNN_{9,2.5}} \cap \overline{X} \mid equal\left(knn_{9,2.5}, x\right)\right)$	0.032536	0.00770549
$P\left(\overline{KNN_{9,2.5}} \cup \overline{X}\right)$	0.103541	0.22589
$P\left(KNN_{9,2.5} \cap unequal\left(knn_{9,2.5}, x\right)\right)$	0.0456118	0.167961
$P\left(KNN_{9,2.5} \mid unequal\left(knn_{9,2.5}, x\right)\right)$	0.621574	0.764045
$P\left(\overline{KNN_{9,2.5}} \cap unequal\left(knn_{9,2.5}, x\right)\right)$	0.0277804	0.0519177
$P\left(\overline{KNN_{9,2.5}} \mid unequal\left(knn_{9,2.5}, x\right)\right)$	0.378426	0.235955
$P\left(\overline{KNN_{9,2.5}} \mid \overline{X}\right)$	0.488196	0.147135
$P\left(equal\left(knn_{9,2.5}, x\right) \mid \overline{KNN_{9,2.5}} \cap \overline{X}\right)$	0.693073	0.207912
$P\left(equal\left(knn_{9,2.5}, x\right) \mid \overline{KNN_{9,2.5}}\right)$	0.521074	0.104182
$P\left(equal\left(knn_{9,2.5}, x\right) \mid \overline{X}\right)$	0.338308	0.030523
$P\left(X \cap unequal\left(knn_{9,2.5}, x\right)\right)$	0.0144125	0.0289532
$P\left(X \mid unequal\left(knn_{9,2.5}, x\right)\right)$	0.19635	0.131578
$P\left(\overline{X} \cap unequal\left(knn_{9,2.5}, x\right)\right)$	0.0589798	0.190926
$P\left(\overline{X} \mid unequal\left(knn_{9,2.5}, x\right)\right)$	0.80365	0.868422
$P\left(\overline{X} \mid \overline{KNN_{9,2.5}}\right)$	0.751716	0.500679
$YuleQ\left(KNN_{9,2.5}, X\right)$	0.966964	0.644044

TABLE 4.22 Error Correlation for $k = 9$, $m = 2.5$

Measure	$\mathbf{X \rightarrow NN}$	$\mathbf{X \rightarrow NN2}$
$P\left(\overline{KNN_{11,1.2}} \cap \overline{X}\right)$	0.0389151	0.0266814
$P\left(\overline{KNN_{11,1.2}} \cap \overline{X} \cap unequal\left(knn_{11,1.2}, x\right)\right)$	0.0122497	0.0210548
$P\left(\overline{KNN_{11,1.2}} \cap \overline{X} \cap equal\left(knn_{11,1.2}, x\right)\right)$	0.0266654	0.00562658
$P\left(\overline{KNN_{11,1.2}} \cap \overline{X} \mid equal\left(knn_{11,1.2}, x\right)\right)$	0.0289022	0.00719993
$P\left(\overline{KNN_{11,1.2}} \cup \overline{X}\right)$	0.104073	0.224115
$P\left(KNN_{11,1.2} \cap unequal\left(knn_{11,1.2}, x\right)\right)$	0.0502131	0.170255
$P\left(KNN_{11,1.2} \mid unequal\left(knn_{11,1.2}, x\right)\right)$	0.648626	0.779331
$P\left(\overline{KNN_{11,1.2}} \cap unequal\left(knn_{11,1.2}, x\right)\right)$	0.0271941	0.0482329
$P\left(\overline{KNN_{11,1.2}} \mid unequal\left(knn_{11,1.2}, x\right)\right)$	0.351374	0.220669
$P\left(\overline{KNN_{11,1.2}} \mid \overline{X}\right)$	0.436586	0.135489
$P\left(equal\left(knn_{11,1.2}, x\right) \mid \overline{KNN_{11,1.2}} \cap \overline{X}\right)$	0.685449	0.211061
$P\left(equal\left(knn_{11,1.2}, x\right) \mid \overline{KNN_{11,1.2}}\right)$	0.495308	0.104638
$P\left(equal\left(knn_{11,1.2}, x\right) \mid \overline{X}\right)$	0.299197	0.0285737
$P\left(X \cap unequal\left(knn_{11,1.2}, x\right)\right)$	0.0149444	0.0271781
$P\left(X \mid unequal\left(knn_{11,1.2}, x\right)\right)$	0.193052	0.124337
$P\left(\overline{X} \cap unequal\left(knn_{11,1.2}, x\right)\right)$	0.0624628	0.19131
$P\left(\overline{X} \mid unequal\left(knn_{11,1.2}, x\right)\right)$	0.806948	0.875663
$P\left(\overline{X} \mid \overline{KNN_{11,1.2}}\right)$	0.722627	0.495756
$YuleQ\left(KNN_{11,1.2}, X\right)$	0.957811	0.634983

TABLE 4.23 Error Correlation for $k = 11$, $m = 1.2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{11,1.5}} \cap \overline{X}\right)$	0.041033	0.0275081
$P\left(\overline{KNN_{11,1.5}} \cap \overline{X} \cap unequal\left(knn_{11,1.5}, x\right)\right)$	0.0122657	0.0216668
$P\left(\overline{KNN_{11,1.5}} \cap \overline{X} \cap equal\left(knn_{11,1.5}, x\right)\right)$	0.0287673	0.00584126
$P\left(\overline{KNN_{11,1.5}} \cap \overline{X} \mid equal\left(knn_{11,1.5}, x\right)\right)$	0.0310628	0.00747138
$P\left(\overline{KNN_{11,1.5}} \cup \overline{X}\right)$	0.102679	0.224012
$P\left(KNN_{11,1.5} \cap unequal\left(knn_{11,1.5}, x\right)\right)$	0.0480951	0.169429
$P\left(KNN_{11,1.5} \mid unequal\left(knn_{11,1.5}, x\right)\right)$	0.650714	0.776708
$P\left(\overline{KNN_{11,1.5}} \cap unequal\left(knn_{11,1.5}, x\right)\right)$	0.0258163	0.0487423
$P\left(\overline{KNN_{11,1.5}} \mid unequal\left(knn_{11,1.5}, x\right)\right)$	0.349286	0.223292
$P\left(\overline{KNN_{11,1.5}} \mid \overline{X}\right)$	0.460317	0.139689
$P\left(equal\left(knn_{11,1.5}, x\right) \mid \overline{KNN_{11,1.5}} \cap \overline{X}\right)$	0.701287	0.21273
$P\left(equal\left(knn_{11,1.5}, x\right) \mid \overline{KNN_{11,1.5}}\right)$	0.527457	0.10733
$P\left(equal\left(knn_{11,1.5}, x\right) \mid \overline{X}\right)$	0.322798	0.0296646
$P\left(X \cap unequal\left(knn_{11,1.5}, x\right)\right)$	0.0135506	0.0270755
$P\left(X \mid unequal\left(knn_{11,1.5}, x\right)\right)$	0.183382	0.12403
$P\left(\overline{X} \cap unequal\left(knn_{11,1.5}, x\right)\right)$	0.0603608	0.191096
$P\left(\overline{X} \mid unequal\left(knn_{11,1.5}, x\right)\right)$	0.816618	0.87597
$P\left(\overline{X} \mid \overline{KNN_{11,1.5}}\right)$	0.752079	0.504327
$YuleQ\left(KNN_{11,1.5}, X\right)$	0.965286	0.64654

TABLE 4.24 Error Correlation for $k = 11$, $m = 1.5$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{11,2}} \cap \overline{X}\right)$	0.0433881	0.0288987
$P\left(\overline{KNN_{11,2}} \cap \overline{X} \cap unequal\left(knn_{11,2}, x\right)\right)$	0.0132013	0.0229036
$P\left(\overline{KNN_{11,2}} \cap \overline{X} \cap equal\left(knn_{11,2}, x\right)\right)$	0.0301868	0.00599507
$P\left(\overline{KNN_{11,2}} \cap \overline{X} \mid equal\left(knn_{11,2}, x\right)\right)$	0.032568	0.00768242
$P\left(\overline{KNN_{11,2}} \cup \overline{X}\right)$	0.103304	0.225602
$P\left(KNN_{11,2} \cap unequal\left(knn_{11,2}, x\right)\right)$	0.04574	0.168038
$P\left(KNN_{11,2} \mid unequal\left(knn_{11,2}, x\right)\right)$	0.625766	0.765325
$P\left(\overline{KNN_{11,2}} \cap unequal\left(knn_{11,2}, x\right)\right)$	0.0273767	0.0515685
$P\left(\overline{KNN_{11,2}} \mid unequal\left(knn_{11,2}, x\right)\right)$	0.374234	0.234675
$P\left(\overline{KNN_{11,2}} \mid \overline{X}\right)$	0.486769	0.146747
$P\left(equal\left(knn_{11,2}, x\right) \mid \overline{KNN_{11,2}} \cap \overline{X}\right)$	0.696067	0.207802
$P\left(equal\left(knn_{11,2}, x\right) \mid \overline{KNN_{11,2}}\right)$	0.52517	0.104465
$P\left(equal\left(knn_{11,2}, x\right) \mid \overline{X}\right)$	0.338757	0.0304443
$P\left(X \cap unequal\left(knn_{11,2}, x\right)\right)$	0.0141754	0.0286648
$P\left(X \mid unequal\left(knn_{11,2}, x\right)\right)$	0.193787	0.13044
$P\left(\overline{X} \cap unequal\left(knn_{11,2}, x\right)\right)$	0.0589413	0.190942
$P\left(\overline{X} \mid unequal\left(knn_{11,2}, x\right)\right)$	0.806213	0.86956
$P\left(\overline{X} \mid \overline{KNN_{11,2}}\right)$	0.75436	0.502484
$YuleQ\left(KNN_{11,2}, X\right)$	0.967335	0.646168

TABLE 4.25 Error Correlation for $k = 11$, $m = 2$

Measure	**X → NN**	**X → NN2**
$P\left(\overline{KNN_{11,2.5}} \cap \overline{X}\right)$	0.0443366	0.0295171
$P\left(\overline{KNN_{11,2.5}} \cap \overline{X} \cap unequal\left(knn_{11,2.5}, x\right)\right)$	0.0136435	0.023474
$P\left(\overline{KNN_{11,2.5}} \cap \overline{X} \cap equal\left(knn_{11,2.5}, x\right)\right)$	0.0306931	0.00604313
$P\left(\overline{KNN_{11,2.5}} \cap \overline{X} \mid equal\left(knn_{11,2.5}, x\right)\right)$	0.0331204	0.00775349
$P\left(\overline{KNN_{11,2.5}} \cup \overline{X}\right)$	0.103976	0.226605
$P\left(KNN_{11,2.5} \cap unequal\left(knn_{11,2.5}, x\right)\right)$	0.0447916	0.16742
$P\left(KNN_{11,2.5} \mid unequal\left(knn_{11,2.5}, x\right)\right)$	0.611444	0.759227
$P\left(\overline{KNN_{11,2.5}} \cap unequal\left(knn_{11,2.5}, x\right)\right)$	0.0284918	0.0531417
$P\left(\overline{KNN_{11,2.5}} \mid unequal\left(knn_{11,2.5}, x\right)\right)$	0.388556	0.240773
$P\left(\overline{KNN_{11,2.5}} \mid \overline{X}\right)$	0.497411	0.149888
$P\left(equal\left(knn_{11,2.5}, x\right) \mid \overline{KNN_{11,2.5}} \cap \overline{X}\right)$	0.692635	0.205088
$P\left(equal\left(knn_{11,2.5}, x\right) \mid \overline{KNN_{11,2.5}}\right)$	0.519418	0.102454
$P\left(equal\left(knn_{11,2.5}, x\right) \mid \overline{X}\right)$	0.344442	0.0306882
$P\left(X \cap unequal\left(knn_{11,2.5}, x\right)\right)$	0.0148483	0.0296677
$P\left(X \mid unequal\left(knn_{11,2.5}, x\right)\right)$	0.202507	0.134407
$P\left(\overline{X} \cap unequal\left(knn_{11,2.5}, x\right)\right)$	0.0584351	0.190894
$P\left(\overline{X} \mid unequal\left(knn_{11,2.5}, x\right)\right)$	0.797493	0.865593
$P\left(\overline{X} \mid \overline{KNN_{11,2.5}}\right)$	0.749795	0.499261
$YuleQ\left(KNN_{11,2.5}, X\right)$	0.967209	0.643089

TABLE 4.26 Error Correlation for $k = 11$, $m = 2.5$

K	M	Accuracy		Errors		
		Raw	**Voting**	**Corrected**	**Introduced**	**Fraction Reduced**
1	1.2	0.93822	0.951959	0.0160146	0.00227498	22.2395 %
1	1.5	0.93822	0.951959	0.0160146	0.00227498	22.2395 %
1	2	0.93822	0.951959	0.0160146	0.00227498	22.2395 %
1	2.5	0.93822	0.951959	0.0160146	0.00227498	22.2395 %
3	1.2	0.94293	0.953065	0.0123458	0.0022109	17.7587 %
3	1.5	0.945279	0.95336	0.0102438	0.00216284	14.7675 %
3	2	0.945484	0.953132	0.0097536	0.00210516	14.0296 %
3	2.5	0.945224	0.952786	0.00964145	0.00207953	13.8052 %
5	1.2	0.945144	0.953616	0.0106508	0.00217886	15.4439 %
5	1.5	0.946897	0.95359	0.00874748	0.00205389	12.6048 %
5	2	0.945964	0.952581	0.00859688	0.0019802	12.245 %
5	2.5	0.945093	0.951802	0.00866417	0.00195456	12.2199 %
7	1.2	0.94582	0.953574	0.00991381	0.00215963	14.3119 %
7	1.5	0.94672	0.952994	0.00827966	0.00200583	11.7753 %
7	2	0.945032	0.951347	0.00826044	0.00194495	11.4894 %
7	2.5	0.943962	0.950447	0.00840142	0.00191611	11.573 %
9	1.2	0.946028	0.953449	0.00956775	0.00214682	13.7497 %
9	1.5	0.945948	0.952219	0.00820597	0.00193534	11.6012 %
9	2	0.943452	0.95005	0.00843026	0.0018328	11.667 %
9	2.5	0.942071	0.948944	0.0086898	0.00181678	11.8646 %
11	1.2	0.946141	0.953363	0.00932423	0.00210196	13.4095 %
11	1.5	0.945416	0.951706	0.00818995	0.00190009	11.5233 %
11	2	0.942436	0.949108	0.00847192	0.00180076	11.5892 %
11	2.5	0.940815	0.947935	0.00890448	0.00178474	12.0297 %

TABLE 4.27 Fuzzy-KNN Accuracy

Network	**Accuracy**
NN	0.910872
NN2	0.803063

TABLE 4.28 NN Accuracy

Measure	**Value**
$P\left(\overline{NN} \cap \overline{NN2}\right)$	0.0403025
$P\left(\overline{NN} \cap \overline{NN2} \cap unequal\left(nn, nn2\right)\right)$	0.0335736
$P\left(\overline{NN} \cap \overline{NN2} \cap equal\left(nn, nn2\right)\right)$	0.00672883
$P\left(\overline{NN} \cap \overline{NN2} \mid equal\left(nn, nn2\right)\right)$	0.00884197
$P\left(\overline{NN} \cup \overline{NN2}\right)$	0.245762
$P\left(NN \cap unequal\left(nn, nn2\right)\right)$	0.156634
$P\left(NN \mid unequal\left(nn, nn2\right)\right)$	0.655316
$P\left(\overline{NN} \cap unequal\left(nn, nn2\right)\right)$	0.0823993
$P\left(\overline{NN} \mid unequal\left(nn, nn2\right)\right)$	0.344684
$P\left(\overline{NN} \mid \overline{NN2}\right)$	0.204636
$P\left(equal\left(nn, nn2\right) \mid \overline{NN} \cap \overline{NN2}\right)$	0.167133
$P\left(equal\left(nn, nn2\right) \mid \overline{NN}\right)$	0.0755615
$P\left(equal\left(nn, nn2\right) \mid \overline{NN2}\right)$	0.034168
$P\left(NN2 \cap unequal\left(nn, nn2\right)\right)$	0.0488257
$P\left(NN2 \mid unequal\left(nn, nn2\right)\right)$	0.20424
$P\left(\overline{NN2} \cap unequal\left(nn, nn2\right)\right)$	0.190208
$P\left(\overline{NN2} \mid unequal\left(nn, nn2\right)\right)$	0.79576
$P\left(\overline{NN2} \mid \overline{NN}\right)$	0.452173
$YuleQ\left(NN, NN2\right)$	0.597788

TABLE 4.29 Neural Network Error Correlation

4.6 Implementation Results

4.6.1 Hardware Configuration

The classifiers were trained and tested on a Linux beowulf cluster. The cluster consisted of 61 individual nodes connected by a gigabit network with a total of 122 2.4 GHz Intel Xeon processors and a total of 64 GB RAM. The cluster is further described in [21].

4.6.2 Software

The training and testing algorithms both utilized a master-worker model. For training, each worker was responsible for training a single neural-network or fuzzy-knn instance. As described in Section 4.3, each of the 10 neural-network-based and fuzzy-knn-based classifiers consisted of an average of 129.8 neural-networks or fuzzy-knn instances, for an average total of 1298 neural-networks. In addition, there were an average of 1298 fuzzy-knn instances per (k, m) pair, of which there were 24. This yielded a total of 31152 fuzzy-knn instances. Each classifier was trained on a separate cpu.

Chapter 5

CONCLUSION

A feature-set based on Fourier Descriptors was developed. The features employed a new method for handling images which contain multiple curves. In addition, a set of software components were developed which allow these features to be extracted from an image and used for classification. The feature extraction software was made available as an extension to the Gamera open-source document processing framework.

Two classifiers were implemented using the feature-set, one based on a fuzzy-knn and the other based on a neural-network. In addition, software was developed for training and testing the classifiers in parallel on a beowulf cluster. The software was used to evaluate the classifier performance on a data set containing newsprint. The fuzzy-knn classifier was tested using a range of parameter values in order to determine their optimal values. The neural-network classifier was made available through the python api in the Gamera system.

The classifier test results were analyzed in order to determine not only the accuracy of each individual classifier but also the degree to which its errors are correlated with those made by other classifiers. The performance of the fuzzy-knn and neural-network classifiers based on the features investigated in this paper were compared to each other as well as to a third neural-network classifier which used an unrelated feature set.

Lastly, the three classifiers were combined using weighted voting in order to obtain an overall improvement in generalization accuracy.

5.1 Future Directions

There are a number of areas where this work could be extended. The method used for combining the Fourier Descriptors from multiple image curves is not rotation invariant. It may be possible to produce a version of it which has this property. In addition, the work presented in this paper applies only to bi-tonal images which has the drawback that it is susceptible to noise introduced during the binarization process. In principle, the methods could be generalized to operate on gray-scale images which would effectively eliminate binarization noise.

The two-layer neural networks investigated in this paper were chosen so that the nodes in the hidden layer were fully connected to both the input and output nodes. It may be fruitful to investigate other architectures. In particular, architectures which are derived from the image signature may be able to take advantage of relationships between elements of the feature vector in order to improve training and/or generalization performance.

REFERENCES

[1] A. Khotanzad, Y.H. Hong, *Invariant image recognition by Zernike moments*, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 489-497.

[2] C.-S. Lin and C.-L. Hwang. *New Forms of Shape Invariants from Elliptic Fourier Descriptors.* Pattern Recognition, 20(5):535–545, 1987.

[3] C. T. Zahn and R. Z. Roskies. *Fourier descriptors for plane closed curves.* IEEE Transactions on Computers, 1972.

[4] Evelyn Fix and Joseph L. Hodges, Jr. Discriminatory analysis: Nonparametric discrimination: Consistency properties. *USAF School of Aviation Medicine*, 4:261-279, 1951.

[5] Evelyn Fix and Joseph L. Hodges, Jr. Discriminatory analysis: Nonparametric discrimination: Small sample performance. *USAF School of Aviation Medicine*, 11:280-322, 1952.

[6] F.P. Kuhl and Ch.R. Giardina, *Elliptic Fourier Features of a Closed Contour*, CGIP 18, 236-258 (1982)

[7] G. H. Granlund, *Fourier preprocessing for hand print character recognition* IEEE Trans. Comput., vol. C-21, pp. 195–201, 1972.

[8] G. U. Yule. On the association of attributes in statistics. *Phil. Trans.*, A, 194:257-319, 1900.

[9] P C Hew. Geometric and Zernike Moments. Diary, Department of Mathematics, The University of Western Australia, http://maths.uwa.edu.au/~phew/postgrad/diaries/geozermom.ps.Z, October 1996.

[10] J. Flusser, T. Suk, Pattern recognition by affine moment invariants, Pattern Recognition 26 (1993) 167–174.

[11] J. Keller, M. R. Gary, and J. A. Givens. A fuzzy K-nearest neighbor algorithm. *IEEE Trans. Systems, Man, Cybernetics*, Vol. SMC-15, No. 4, pp. 580-585, 1985.

[12] L.I. Kuncheva, C.J. Whitaker, C.A. Shipp, *Limits on the Majority Vote Accuracy in Classifier Fusion.* Pattern Analysis and Applications, 6, 2003, 22-31.

[13] M. K. Hu, *Visual pattern recognition by moment invariants*, IEEE Trans. Inform. Theory, vol. 8, pp. 179-187, Feb. 1962.

[14] O. Trier and A. Jain and T. Taxt, *Feature extraction methods for character recognition - A survey.* Pattern Recognition 29, pp. 641-662, 1996.

[15] T. Pavlidis. *A vectorizer and feature extractor for document recognition.* Compt. Vision Graphics Image Process. Vol. 35, Issue 1, 111-127, 1986.

[16] Rakesh Agrawal and Christos Faloutsos and Arun N. Swami, *Efficient Similarity Search In Sequence Databases.* Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms, Chicago, Illinois: Springer Verlag, pp. 69–84, 1993.

[17] R. L. Cosgriff, "Identification of shape," Ohio State Univ. Res. Foundation, Columbus, OH, Tech. Rep. ASTIA AD 254-792, Dec. 1960.

[18] R. O. Duda, P. E. Hart and D. G. Stork, Pattern Classification. New York: John Wiley & Sons, 2001.

[19] T. H. Reiss, *The Revised Fundamental Theorem of Moment Invariants*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, no. 8, pp. 830-834, Aug. 1991.

[20] Y. Rui and A. She and T. Huang, *Modified fourier descriptors for shape representation – a practical approach.* Proc of First International Workshop on Image Databases and Multi Media Search., 1996.

[21] *Beowulf Cluster Lab.*
`http://cs.boisestate.edu/~amit/research/beowulf/`

[22] Michael Droettboom, Ichiro Fujinaga and Karl MacMillian. *The Gamera Project.*
`http://dkc.jhu.edu/gamera/`

[23] *The Python Programming Language.*
`http://www.python.org/`