CHAPTER 4

Liquid processes and phase transitions
In this chapter we will look at the software level. The lack of self-organization in
hardware steers us into this direction. Software, although it does have mineralized
components such as computer code, acts more like a liquid. It can move from system
to system and it can create moving components on devices such as computer screens
and in loudspeakers. This is the level where the first signs of self-organization occur.
This chapter is about a simulation of the primate visual cortex, which is a flexible
and self-organizing structure. It describes a complete system where several important
aspects come together.

The system described in this chapter uses ITO (Input → Transform → Output) ma-
chine learning mechanisms. Instead of using these methods to find an optimum, they
are used to generate hypotheses which are later verified or falsified by a hominid sort-
ing mechanism. Using a circadian rhythm (a 24h daily cycle) the verified information
is used to generate new knowledge. This knowledge is used by the machine learning
mechanisms to create new and better hypotheses. Once the scaffolding structures were
in place, the machinery was able to improve on itself using iterative self-improving
processes and a little guidance from humans. This self-organizing system is used to
bootstrap knowledge from the ground up. It is a growing system where the history of
the system plays the role of an incremental knowledge production system. The system
creates its own knowledge. During the incremental creation of knowledge first the hu-
mans are being used to generate grounded knowledge and the later the humans function
as a sorting machine on the outcomes of the learning mechanisms. After the initializa-
tion of the system humans are steering the process of knowledge production, leaving
most of it to the interactive and iterative processes. Just as an animal is guided by expe-
rience, so is this handwriting recognition architecture guided by its experiences of the
interactions with the data and human-machine interactions using the internet. The ma-
chinery is constantly evaluating itself in the process, creating an ever improving system
of feedback mechanisms.

Perhaps one of the most important aspects of this research, although the authors
did not fully realize it in the beginning of the project, is that the amount of words
found by the computer show phase transitions. From relatively poor recognition rates,
suddenly these rates shoot upwards and reach high levels of recognition rates, only to
hit another boundary. The ceilings they hit are caused by the environment. In this
case the system had been mined out, which means that there were no more occurrences
of a specific word to be found in the data. These phase transitions are to be expected
from the Generative Artificial Intelligence point of view, although it does not mean that
the recurrent abstract machine is already in place. What is still lacking is an artificial implementation of the next levels of mental machinery. At this moment in time the next levels are still in the minds of the hominids, but as history has shown, the transfer from biological implementations to artificial ones is in progress. There are no theoretical reasons why a machine should not be able to form next levels.

The article was accepted under the name of “Handwritten-word spotting using biologically inspired features” in the IEEE Transactions on Pattern Analysis and Machine Intelligence 30(11) in 2008.
4.1 Introduction

There is a discrepancy between generic text search, where the human user enters typed keywords as a Unicode\footnote{Unicode is a standard encoding method for text data} character string and the methods of search in multi-medial data such as images and sounds. Although many advances have been made in recent years (Vailaya et al., 2001), effective search in image collections is still a difficult problem. Perhaps not surprisingly, this also holds for image collections of Western handwritten historical texts. These documents contain connected-cursive handwritten text in a vast variety of script styles. This data is more difficult to recognize than isolated hand printed characters and ideographic writings such as Japanese and Chinese, since there are no obvious ways to segment and normalize the writings into meaningful chunks, such as words or characters. Common methods of optical character recognition fail miserably on freestyle connected-cursive handwriting, unless dedicated machine learning effort is spent on a particular, well described document of homogeneous layout and writing style. The involvement in human labor is much too costly for institutions in the application domain. Contemporary approaches focus on a combination of language modeling and Hidden-Markov Models (HMMs). Both levels of modeling require massive numbers of labeled word or character instances. In a laboratory condition, character models can be estimated from labeled words using dynamic programming. However, this process takes place under manual supervision. A particular document collection usually requires at least one computer science PhD project of three to four years. This is a big obstacle for creating a search engine for handwritten-cursive text. Cultural heritage institutions are reluctant to digitize manuscript collections, given the poor access methods. Usually, there are only a few annotated text items and meta tags for a huge amount of document-page images. For example, the Dutch National Archive contains about one hundred kilometers of books and boxes filled with handwritten cursive text. The number of documents is overwhelming. The estimated size of the governmental archives in the Netherlands alone is estimated to be approximately 600 kilometers of book shelves, which is twice the diameter of the country. Annotation is scarce and it is unfeasible to annotate even a small percentage of all the items. Finally, there is also a huge discrepancy between the type of semantic annotation that is required for humanities research, and the detailed, precise labeling of image regions of interest that is needed by computer scientists to train classifier algorithms.
This handwriting recognition problem cannot be solved using the ITO paradigm. There are too many different writing styles, too many document types and not enough humans to solve it ITO-style. It becomes essential to not only apply, but also combine, the best of the artificial and the hominid world.

Most challenging is the lack of data to train machine-learning algorithms that are known to perform well (Park and Lee, 1995; Lavrenko and R., 2004; Liwicki and Bunke, 2006; Pal et al., 2006; Caillault and Viard-Gaudin, 2006). This may seem to be a rather practical issue, but in fact there is a profound theoretical problem lurking behind it. The usual way of obtaining labeled data is to manually label the data until a few hundred to a few thousand instances per class have been collected. Next, methods such as Hidden-Markov Modeling or Support Vector Machines can be trained on the pattern classes. Finally, it is possible to start data mining in unfamiliar subsets from the collection. Inconveniently however, obtaining hundreds of labels per word requires an amount of effort that one does not want to repeat for every writing style. It would be more constructive to label a word only once, perform a data mining procedure on the ensemble, and finally to use a fast and convenient user-interaction tool in order to quickly select correct items from an already acceptable list of candidates (Schomaker et al., 1999). The selected candidate patterns can then be used for further supervised training and the process can be repeated reiteratively until the target retrieval accuracy is achieved. This minimizes the time the user has to spend on the training of the handwriting recognition system. Additionally, the expertise of such a system concentrates on common search patterns, satisfying the most frequent user’s needs at an early stage.

This paragraph above shows a shift in the thinking about these type of problems. The shift is triggered by the difficulty of the problem. Only because a real world, and not a toy world, problem is addressed, there is need for a paradigm shift. The old paradigm of ITO is not powerful enough to solve the handwriting problem. When the old paradigm does not suffice anymore, some scientists start looking for a new one (Kuhn, 1962). It is interesting that the machine learning algorithms did not even have to be adapted. They had to be applied in a different context for a different reason. The new methodology had several extra advantages, such as early user satisfaction and the quest for one-shot learning mechanisms (see below).

In order to achieve the aforementioned goal, a one-shot learning algorithm is needed to bootstrap the retrieval system. Currently, the only algorithm that can yield useful pattern retrieval based on a single query item is nearest-neighbor search. Nearest-
neighbor search on normalized pixel images of word patterns, as is shown in section 4.6, already functions reasonably well. The only way to improve this performance, while avoiding expensive parameter search (training), is to use image features which are more powerful than a normalized pixel-based image feature vector. That is, these features need to be more invariant to irrelevant shape deviations. In this study we present a method that uses a small feature vector of limited dimensionality (D=400), which performs very well on a 2100-class problem. Using a relatively small feature vector is important since the data to be processed consists of hundreds of books. The data used here are a subset of automatically segmented word-zone images from a book of 1300 pages of connected cursive handwritten texts. Only word images of classes which have two or more instances are taken into account. Speed is as important as accuracy, because a procedure that is too slow will not be able to process large amounts of data, and a procedure that is not accurate enough will frustrate the user.

It is important to note that this study is motivated by the limited usability of traditional methods in the open domain of historical handwriting retrieval and recognition. A few attempts to use less conventional methods have been successful (LeCun et al., 1998) but the problem is far from being solved. Also the use of Support Vector Machines in the final classification after using convolutional neural nets have been successfully applied to handwriting (Lauer et al., 2007). The results of those studies are difficult to compare to the study here, due to the different datasets and the different methods of gathering the data.

Apart from label-greediness, there is an additional problem with the predominant paradigm of Hidden-Markov modeling. Historical handwriting is characterized by an abundance of loops, guirlands and embellishments which all violate the assumptions of the traditional left-to-right state transitions in Markov modeling. Figure 4.1 shows an example of complex manuscript patterns, which clearly span two dimensions. It is also not trivial to automatically assume that the line of text is completely horizontal. Most of the scanned pages have an offset of a few degrees which can make recognition very difficult. To counteract this, robust feature representations are needed.

The success of Hidden-Markov Modeling is largely due to the robustness of this paradigm to variations in the horizontal position of relevant structural features in a word pattern. Vertical positional information is usually handled by means of baseline and size normalization at the word level. Usually, narrow sliding windows are used (Liwicki and Bunke, 2006) from which basic features are computed that are sometimes as
Figure 4.1: Examples of historical handwriting. The presence of guirlands is a nuisance to many recognition models, but can also be exploited by the use of proper feature-extraction methods. Whereas the curls of the 'd' in *donderdag* may pose a problem for a left-to-right HMM, these structural features represent a robust piece of evidence for the character *d*, if an appropriate feature and classification method were used.
simple as pixel intensities. Although attempts at true 2D-Markov modeling of handwriting have been undertaken (Park and Lee, 1995), these random-mesh approaches are complicated, with an arbitrary image-formation assumption. Commonly, each pixel intensity is conditioned on the upper-left ’preceding’ image rectangle. The larger complexity of such models requires even more labeled data than the usual pseudo-2D (sliding-window) approaches to Hidden-Markov Modeling. Therefore, in order to handle both (1) the difficulties of label scarcity and (2) the problems of feature translation and scaling in two dimensions in historical manuscripts, it may be conducive to look at biologically inspired features from a domain which encounters invariance problems at a large scale, that is, computer vision for poorly and unstructured environments, such as robotics.

This problem occurs more often in contemporary AI: training methods that require a very large amount of data. Often the amount of data required to train the procedures is too big to acquire without superhuman capabilities, or at least one needs an amount of patience that few possess. This learning problem, or perhaps one should state: This lack of learning capabilities problem of most contemporary AI algorithms, arises from the use of artificial data sets and leads many in the AI research community astray.

**Similar methods and rationale for the image-based feature approach** Retrieval of handwritten passages in historical manuscripts has been attempted before. Lavrenko and Rath (Lavrenko and R., 2004; Rath et al., 2004) used dynamic time warping on word contours. The collection consisted of neatly written, well-separated words which lend themselves to whole-word contour extraction. We previously proposed a fragmented-connected component contours approach (Schomaker et al., 2007; Schomaker, 2008a) to allow for a larger range of manuscript styles to be processed. Contour-based approaches allow for invariant shape matching using for example local affine transforms on the coordinates. However, the detection of reliable contours has intrinsic problems when image quality (i.e., contrast) is variable and the ink trace is broken within characters and words. For such data, pure image-based approaches may have an advantage under the condition that invariance can be obtained for irrelevant shape aspects in the 2D plane: Robustness must be obtained for problems occurring in all three dimensions (x, y and luminance).
4.1. Introduction

**The instrument: the Monk web site** Quite often it is not relevant how the data is obtained within the machine learning community. Sometimes however, it is. To understand the problem that our group faces we think it is important to understand our data set, which requires understanding how we gathered the data. This is not only important in order to obtain some insight into the data. It also demonstrates the real-world problems that we face, such as a lack of labeled data, interest by the public in items that are rare such as family and place names, and giving an acceptable answer within a limited time period. As an instrument for harvesting labels of handwritten patterns, we have developed a web-based search engine, dubbed Monk\(^1\), for annotation, search, and retrieval in handwritten-text image collections. Since annotations (labels) are scarce, the focus is on methods that allow for bootstrapping a search engine from a virtual zero-knowledge state to an effective search method in incremental steps under guidance of regular users. It is important that regular users train the system since it will probably have to be repeated for many writers.

One of the outcomes of using GAI principles is that it is difficult to predict the exact outcome of a method or group of methods. Only by running the system it becomes evident what it can and cannot do. The discussion shows that increase of the importance of the steering principle: the hominids steer the system toward a more intelligent state where the search engine becomes better over time. If the humans steer the system in the wrong direction (which means that the hominids were lying to the computer) then the machine learns the wrong categories. In other words: If the badly functioning sorting machine is applied, the system will not function correctly. The steering, using real world data, leads to a grounded system and avoids the symbol grounding problem.

The Monk system does not use character-based recognition. At least hundreds of instances per class would be required to train conventional methods such as HMMs (Liwicki and Bunke, 2006; Pal et al., 2006; Caillault and Viard-Gaudin, 2006) to recognize individual letters. Moreover, characters appear in different styles and allographs, even for a single writer. In a real world, where a system needs to be trained continuously, the number of words and the number of character and word classes both increase over time. Initially, there are only a few labeled instances per class. About 75.0\% of the word classes of the data collected by our research group contain less than ten items and almost half of the classes contain only two or three instances. Using a whole-word

---

\(^1\)http://www.ai.rug.nl/Monk; this is the search engine. One could try ’Groningen’, a place name, or ’Jansen’, a typical Dutch family name, for example.
approach provides at least preliminary access to a manuscript, postponing the training of character-based systems to a later, more label-rich development stage. This concept has the additional advantage that the actual language or script type being fed to the system is irrelevant. The Monk system can be used to bootstrap handwriting retrieval in any language since it does not make assumptions other than requiring a row-based or columnar layout organization of lines with white space between meaningful text shapes in a line. Theoretically, a language with lines being written from the top to the bottom of the page would simply mean that pages require a 90° rotation. However, the current system is rather crude with regard to layout analysis. Apart from automatic line segmentation and the extraction of well-separated word candidate images, the system does not handle non-text items well. The use of explicit layout knowledge can always be added, as presented in (Bulacu et al., 2007). The Monk system is an experimental e-Science project providing access to this data for humanities specialists to collaborate on the annotation and transcription of text images. In layman terms, the Monk system entails: "Specialists training several machine-learning algorithms on a supercomputer using a web browser" (van der Zant, 2008).

Connecting such an e-Science project to GAI could lead to very interesting results. On the other hand, looking at the violent history of hominids and other mammals, perhaps it is better that well-intended individuals bootstrap the system before connecting it to the Internet. At least it shows that many humans can be used in parallel for sorting out preferable sensor interpretations and most likely also for preferable behaviors of autonomous robots.

Since the amount of data is overwhelming, the extraction methods have to be automated as much as possible. An initial seed collection of text is annotated at the line level by volunteers. It would be too tedious to do this at the word level, requiring user actions for each word segmentation. Automated word-segmentation procedures are difficult to implement (Fig. 4.2) without prior knowledge on the text content. During each night the automated procedures try to increase the knowledge-level in the system, exploiting the availability of new labels. By generating hit lists of text objects, the annotation process becomes increasingly effective. Current methods that allow for correlating line-strips (Schomaker, 2007b) detect word presence, or even word-category presence such as "geolocation" or "proper name" (Schomaker, 2008b), in a line. The use of word-based HMMs and especially Support Vector Machines (SVMs) (Joachims, 2005, 2006) has proved to be quite fruitful in the problems where there is a sufficient
number of training samples. The *Monk* system is able to suggest labels for unseen
words, to be validated by human users over the Internet the next day. However, the
feature representations for these experiments have been quite crude, consisting of nor-
malized word images for SVM classification, sliding-window images for word-based
HMMs, and connected-component contour features for line-based HMMs. It was our
impression however, that a performance ceiling had been reached, particularly with
regard to the machine-learning methods. Better features were needed to gain better
performance. Powerful invariant features for handwritten images require both that the
distribution of ink density and the directionality of edges is well preserved.

Above the circadian system is described. Using iterative procedure the knowledge is
not only growing but it is also refined. One cannot expect that these complicated real
world problems have a 'once and for all' solution. Even human experts in the domain of
historical texts do not always agree. It is possible that through refinements the artificial
system becomes as good as its teachers, perhaps even outperforming them.

**A biologically inspired feature for handwriting retrieval** In (Serre, T. et al., 2007) a
new method for object recognition is described. It is biologically inspired and the theo-
ries behind the model are based on the standard model of the visual cortex as described
in (Poggio and Edelman, 1990). Here, we use the methods of Serre as described in
(Serre, T. et al., 2007) and also his implementation of the algorithm. It works very well
on a standard set of computer vision problems with many labeled examples (500) on
in handwriting retrieval is different, however. There are often very few labeled word
examples and the number of word classes is very large. In this study, we deal with
a 2100 class problem, where almost half of the classes contain only two or three in-
stances. We apply the method described in (T. Serre, L. Wolf and T. Poggio, 2005)
and adapt it to the problem of handwritten word classification. It is demonstrated that
this approach performs very well using regular nearest-neighbor methods on a feature
vector of acceptable dimensionality (400). In this paper, experiments are described
that compare learning algorithms on the output of the model of Serre (Serre, T. et al.,
2007; T. Serre, L. Wolf and T. Poggio, 2005) with the same learning algorithms on the
normalized handwritten text images. The feature vector of the normalized images is
4.5 times larger (1800), requires many more computations and performs considerably
less well. The results are shown for class size. The rationale behind the experiments is
to compare the average performance per number of instances of a class. Classes with more instances are expected to perform better in all experiments. In order to provide a comparison with state-of-the-art recognition methods an additional experiment is carried out using an unsheathed version of the model proposed by Serre et al. (Serre, T. et al., 2007).

In a previous comment the danger of using methods that work well on artificial data sets was already stressed. Although one could argue that the neural method described above works well on a natural set, humans created this data set and therefore it still is artificial. That the neural method is applicable to more complex, more real world problems, might be due to the fact that it is derived from a biological implementation (the primate visual cortex) which had to deal with real world problems and did not have the luxury of having a scientist around during the evolutionary history for the configuration and training of its neural patterns.

The rest of this chapter is structured as follows. The data of the Dutch National Archive used for the experiments is described in section 4.2. The Monk system and its automated procedures are described in section 4.3. The neuro-physiological algorithm based on the primate cortex is described in section 4.4 and is referred to below as the Standard Model Features (SMF). The SMF consists of two important groups, L2 (or C1) and L4 (or C2)-features) (Serre, T. et al., 2005, 2007). We will use the L2 and L4 terminology. Section 4.5 explains the nearest-neighbor experiments in detail. The experiments are performed in a real world setting, which means that all the procedures are automated, except for annotation/label validation. In the results section (4.6) the performance of nearest-neighbor experiments are reported for the biologically inspired method, the unsheathed version of the standard model, and for a normalized-image reference method. The last section (4.7) presents a discussion.

The non-automated part of the total process is not a weakness of the system. It does not take away the autonomy the machine has. This non-automated part is the steering part, the adjustment and grounding of the machine intelligence in the world. Just as with humans, the teaching is also essential for the machine intelligence in order to understand the complexity of environment.
4.2 The Data

In the National Archive of the Netherlands, governmental documents are stored and preserved. Some are in book form, usually these are the registers/indices that give access to the rest of the actual, heterogeneous documents (letters, financial administration, declarations of war, etc.). Often, these writings are stored in boxes with a ribbon around them. In total there is about one hundred kilometers of bookshelves filled with this material. The National Archive is specialized in state-related documents such as the laws that have been discussed in the Dutch parliament, documents from the Dutch ministries and the Cabinet of the Queen. The name of this collection is “Het Kabinet der Koningin”, abbreviated as KdK. It comprises about 3 kilometers of boxes on shelves filled with (mostly) handwritten documents. Our focus concerns the index books of the KdK, which consists a few hundred thousand pages. On these pages there are references to documents with royal decrees. The content of the registers is relatively structured. The total archive of the KdK is estimated to be in the range of 25 to 35 million pages. An example of a page is shown in fig. 4.3.

The data used in this study are from a single register/index book. This type of book contains abstracts of royal decrees. It consists of 1146 pages of handwritten text, more or less structured. The book is cut into line strips by an automated procedure (Schomaker, 2007b). These line strips are annotated via a web-based system where specialist volunteers log in and transcribe the line strips. This is a laborious procedure and it has taken more than a year to transcribe one book (29k lines). Note that the main purpose of this transcription was not system training, but to be able to quantitatively measure the performance of retrieval algorithms. In parallel, a word-extraction procedure was used, as described in section 4.3. The rationale for using a large single-writer data collection is that the proposed method should at least work well with such an isogenous (Xiu and Baird, 2008; Sarkar and Nagy, 2005) collection, i.e.,
Chapter 4. Liquid processes and phase transitions

one coming from a single source as regards style and content.

4.3 The Monk system

The Monk system derives its name from the idea of a Monk, from before the invention of the printing press, doing laborious work copying books. The system is implemented on a 200 dual-core cluster. It has a data server consisting of a few hundreds of Terabytes. By the end of the year 2007 it will be accessible for the public as an experimental search engine for a part of the collection of the KdK. Building a system of this size, including the parallel algorithms, requires careful research and several man years of work. There are bottlenecks in both the hard- and software that can slow the system down to a grinding halt (van der Zant, 2008). The Monk system needs this capacity for two reasons; hundreds of users should be able to access the system retaining real-time performance; and processing images, especially image correlation on the KdK line strips, requires for every book several years of computation on a regular computer. An overview of the steps to get black/white line strips (Fig. 4.2) and words out of raw page scan (Fig. 4.3) is given below (Schomaker, 2008b).

1. raw page scan The pages are scanned on a high performance scanner with a resolution of 300dpi.

2. conversions A 300-dpi color line-strip image is converted to bi-tonal (black ink and white background) by de-satiating in HSV color space and thresholding. This is particularly effective in removing the red table lines of the printed ink.

3. cutting line strips The pages are cut into line strips using horizontal ink-pixel density and white pixel run-length histograms. The distribution of line-strip height has its mode and median at 185 pixels.

4. connected components Connected-ink components on the border of the line strip image are deleted if they are too small and do not cover the middle zone of the line strip.

5. unshearing Each line strip is unsheared to obtain an average angle of 90° for the vertical ascenders and descenders in the script.
4.3. The Monk system

Figure 4.3: Example of a page of 'the Cabinet of the Queen', a.k.a. ‘Het Kabinet van de Koningin’. In the early 19th century archives in the Netherlands became more or less standardized. Napoleon Bonaparte started this trend.
6. transcription Line strips are transcribed/labeled at the whole line strip level.

The results of the line-strip processing stage give control over the collection, allowing both keyword-based and pattern-recognition based matching methods to select sub sets of the total list of line strips (29k).

In this study, the next procedure is to try to extract words from the line strips as in (Schomaker, 2008b). The extracted word zones are conservatively-estimated candidates for actual words with the goal to bootstrap classifiers. Using a conservative estimation increases the chance to get clean examples, which increases the performance of the classifiers. A considerable portion of word-zone images contains ascenders and descendents from the neighboring lines. Some of the word zones contain non-text items, such as drawn (underscore) lines, scribbles, large speckles, lineation fragments or isolated ascenders and descendents which survived pruning.

1. system: Create word zones using white space between connected ink components; word zones are not words, but estimations of what might be a word. The word zone has to contain at least 30 pixels of connected ink and the white space on both sides is at least ten pixels.

2. human: Label word zones using a web based interface.

3. system: Train SVMs and HMMs (Rabiner and Juang, 1986) on word zones

4. system: Present word zones derived with the trained SVMs and HMMs

5. human: Select true positives using a web-based interface

6. system: add word zones to labeled set
The cooperation between the artificial and the natural becomes transparent here. Together they form a generator that only has to label one item from a certain class in the best case scenario. The machine uses this label, combined with ITO machine learning techniques, to generate hypotheses, which on their turn are sorted (right or wrong label) by the human sorting machine. The sorting ensures grounding of the artifacts, in this case the word-images classes. The process is repeated many times which is the cyclic flux the generative procedures are in before demonstrating a phase transition. After a while the system becomes useful on the next level, which is a search engine for handwritten texts. This next level has the possibility to become a generator in itself. The next level generator in this chapter is research in history and linguistics. The system is not yet a complete GAI system, since the next levels only exist in the mental machinery of humans. This research demonstrates that all the mechanisms exist to create GAI systems.

This is a combined strategy where humans bootstrap the system by labeling the word zones. Once there are enough word zones of a certain label, a class can be trained. Both the SVMs and the HMMs produce high recognition rates. SVMs work better on short words. HMMs work better on longer words. The words that are harvested by the Monk system are used for recognition by the algorithms working with the SMFs, here. The Monk system extracted, at the time of writing, 100k word-zone instances, of which 45k were labeled by three supervisors and the computer in a period of 200 calendar days. The supervisors were working part time. As a result, there are 2100 different word classes with at least 2 items. As shown in section 4.5, at least 2 instances per class are needed otherwise it is not possible to create a query that can succeed. The remaining unlabeled word instances (100k-45k=55k) consists of many useful, legible words as well as illegible residual items. It becomes progressively more difficult to 'mine out' low-frequency words, which gave an impetus to utilize an improved feature representation.
The last sentence shows that at the moment these hominids were tracking this part of the phylum they did not realize the importance of the different way of working they performed, compared to contemporary AI. At least they realized that the ITO way of working was not really helping the research, but they did not formulate a new methodology at this moment. This does not mean that the importance of their work is diminished, but merely that the theoretical framework was not yet developed. The answer they provided, which was not consistent with the research from the literature at that moment, was a step toward the formulation of GAI. The text is still formulated in Classical AI terminology, which is probably more due to the peer pressure from human reviewers than their own biological mental processes.

Figure 4.4 shows the number of harvested word labels over 200 calendar days via the Monk web site for a selected set of words. Words are sorted from high frequency (‘van’) to low frequency (‘Gouverneur’). The curves are characterized by sudden steep increases: After a critical number of training instances has been harvested for a word, the resulting high-quality hit list can cause a speed up of the labeling process. Thereafter, high-frequency words will remain productive (rich gets richer). Low-frequency words may remain at a dormant level until motivated users have entered the critical minimum amount of labels. The flat lines between day 100 and day 190 are caused by the Dutch summer season 2007, after which labeling by users resumes. Total number of harvested word labels is 45k.

At the moment of writing it is still not clear whether these punctuated equilibria really represent a shift in phase space, or whether they are mere epiphenomenological artifacts. At least it is encouraging to see phase shifts, to see sudden changes which were unexpected. Again the metaphors of dynamical systems theory come into mind: processes which seemed chaotic and unrelated at first transform into coherent (dynamical) structures which were not predicted by theory but emanated from the actual execution of the processes on a real world problem.
4.3. The *Monk* system

![Figure 4.4: Label harvest over a period of 200 days. When a machine learning technique performs acceptably well, human labelers label the data using a single click in the web interface and this causes a steep rise in the amount of labels. By having more labels the machine learning technologies perform even better which can result in even more image labeling. The plateaus can be caused by not having enough labels or because there are no more examples of the class in the image data.](image-url)
Figure 4.5: Distribution of number of instances over classes. 75.0% of the all the word zone classes have less than ten items.
4.4 Neuro-physiological inspired Method for classification

4.4.1 The Standard Model of the Visual Cortex

Initially, visual patterns are represented by means of a mosaic of retinal ganglion cells. In the optic chiasm the representation of the left half of the visual field crosses to the right hemisphere of the brain, and vice versa. Visual signals are then propagated through optic tracts that terminate primarily in the lateral geniculate nuclei (LGN), situated in the thalamus. From this starting point, several stages of visual processing proceed for the left and right half of the visual field in opposite hemispheres. Upon early stages of visual processing it is commonly agreed that it is mostly bottom-up and feed-forward. A representation is created from low-level inputs only. Adaptation and association with previous visual experience occurs in later stages of visual processing.

The terminology used here is in contradiction with a common perspective. This part of the system is feed-forward. The usual manner to discuss these systems is that they perform classification. Classification is a top-down and not a bottom-up procedure. Classification means that there are a limited amount of classifiers and if the input does not trigger a positive outcome of one of the classifiers, then there is no classification or worse, a random and most likely wrong classification. It is difficult to believe that this is the (only) way the biological brain works, otherwise it would be impossible to learn new classifications or categories. If the GAI perspective is applied to this terminology, then the feed-forward system is not only to limit the amount of inputs to the classifiers, but to generate possibilities so the system can learn something new.

From LGN, visual processing splits into parvocellular and magnocellular pathways. These, in turn, branch into two separate streams termed the 'what'- and 'where'-pathway. These ventral and dorsal visual object-processing streams mediate visual object recognition, and object-directed action and spatial analysis (Goodale and Milner, 1992; Ungerleider and Mishkin, 1982), respectively. The stages of low-level visual processing involve a decomposition of visual patterns into primitive elements, achieved by hierarchical processing(i.e., increasing receptive field sizes) within distinct areas of the occipital lobe. The primary visual cortex (V1) processes visual information retinotopically and magnifies the central (foveal) area of the visual field so that centrally
presented objects are represented disproportionately large. After V1, visual information is projected in a sequence to V2, V4, to posterior, and anterior inferotemporal cortex (Ungerleider and Mishkin, 1982). The standard model of visual object recognition is a theory proposed by Riesenuber and Poggio (Riesenuber and T., 1999) that accounts for the first 100-200 milliseconds of ventral feed-forward processing.

The terminology in the neuro-sciences is more consistent with GAI terminology. For example, it is stated above that visual patterns are decomposed into primitive elements and not that features are extracted. The different might be subtle, but it might also prove to be important. Another example from the text above is that visual information is projected, which is a generative action.

V1 initially processes the visual information with alternating simple and complex cells as they have been identified by Hubel and Wiesel (Hubel and Wiesel, 1962). A retinotopically organized mosaic of simple cells in V1 filters the image for the detection of edges and bars, a process that has often been modeled by oriented Gabor filtering. Complex cells, in turn, combine the responses of the these simple cells, representing the image by more complex features. This way, a representation is achieved with: (a) an increased tolerance to the different positions in which an object may appear in the visual field, (b) increased tolerance to the size of the object’s projection, and (c) increased tolerance to objects in the image that are rotated in depth, but more importantly for the application described here, rotated in the 2d-plane of the visual field.

Simple and complex cells in primary visual cortex are tuned to different orientations and have different receptive field sizes. After processing in V1 the visual information is grouped into representations that describe the visual image in terms of different orientation and resolution. Next, after V1, the resulting complex features are propagated further down the stream, being subjected to higher level processing. At the level of V4 and posterior inferotemporal cortex (PIT), these complex features are compared with patterns of activity stored in the synaptic weights of the neural cells: The complex features within a patch of the original retinal image are compared with stored patches from previously seen visual images. At this level of visual processing, the neuronal representation of an image can be seen as a feature vector of patch/appearance based descriptors (Heisele, B. et al, 2001; Weber et al., 2000; Mohan et al., 2001; Ullman et al., 2002) and histogram-based descriptors (Mikolajczyk and Schmid, 2003), combining robustness to variance with template matching.

Recently, Serre et al. (Serre, T. et al., 2007) proposed a computational model based
4.4. Neuro-physiological inspired Method for classification on the standard model of visual processing. This model comprises an algorithm for feature extraction, and is demonstrated to provide excellent feature representations for object detection tasks such as the detection of pedestrians in gray-valued images. The computational model constitutes four processing layers according to the main processing steps of the standard model: Layer 1 models simple cells in V1 by subjecting the image to Gabor-filters of different orientation and size. Layer 2 models subsequent complex cells, achieving locally invariant complex feature descriptions. These complex cells are modeled by applying a MAX-operator locally to the first layer’s results (images, filtered with different orientation and size). Layer 3 creates patch-descriptors for shape recognition. Patches resulting from previously seen shapes subjected to differently sized and orientated filters are matched to all possible crops from the new image by means of radial basis functions (Powell, 1987). Finally, Layer 4 applies a MAX-operator to the result of Layer 3, resulting in a representation that expresses the best comparison with previously seen images.

4.4.2 Implementation of the Standard Model

For clarity, we provide here an algorithmic description of the implementation of the standard model. For further reading we refer to the detailed description provided by Serre et al. in (Serre, T. et al., 2007), whose implementation we used in our experiments.

4.4.2.1 Layer 1: Gabor Functions

The first layer consists of simple cells that take their input from the signals transmitted by the LGN. Neurons in the visual cortex propagate an increased response when specific stimuli are presented in particular parts of their 'visual field'. That is, specific stimuli presented in the receptive field of a neuron invoke increased response. The response of the neurons in the first layer, simple cells, are well modeled by Gabor functions, linear filters whose impulse response is defined by a harmonic function multiplied by a Gaussian function (Jones and Palmer, 1987). They are described by the following equation (Gabor, 1946):

\[ g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(\frac{2\pi}{\lambda} x' + \psi\right) \] (4.1)
Figure 4.6: Overview of visual processing according to the standard model of visual processing. The scheme follows the ventral visual pathway running from the lateral geniculate nuclei to the inferotemporal cortex, via the primary visual cortex (V1), V4, and the posterior inferotemporal (PIT) cortex. V1-simple corresponds to the first layer of the standard model, while V1-complex, V4/PIT, and IT correspond to layers two, three, and four, respectively. Dashed arrows indicate max-operations.
4.4. Neuro-physiological inspired Method for classification

\[
x' = x\cos\theta + y\sin\theta \tag{4.2}
\]
\[
y' = -x\sin\theta + y\cos\theta \tag{4.3}
\]

The filter parameters \(\lambda, \theta, \psi, \sigma\) and \(\gamma\) represent the wavelength, orientation, phase, width, and aspect ratio, respectively (Gabor, 1946). The first layer contains simple cells that implement equation (2.3), using different parametrization regarding width and orientation. Each pixel of the input image is the center of a couple of Gabor filters with different orientations and different sizes (measured in pixels), representing simple cells of different orientations and sizes. The filters are normalized so that they yield an activity between minus one and one.

**Algorithm**

1. For each orientation \(\theta\) and size \(\sigma\) of a Gabor-Filter \(g(\theta, \sigma)\),
   
   (a) Center \(g(\theta, \sigma)\) at each pixel of the input image
   
   (b) Normalize \(g(\theta, \sigma)\) so that it yields a response between -1 and 1

2. Calculate the response of all Gabor-filters \(g(\theta, \sigma)\) at each pixel, this yields filtered \(s1\)-images for all values for \(\theta\) and \(\sigma\)

4.4.2.2 Layer 2: Local Pooling

Complex cells in the second layer take their input from the simple cell afferents of the first layer. The complex cells respond according to the maximum activation among the input. Also complex cells focus their attention in a receptive field, usually twice as large as the receptive fields of simple cells in layer 1. According to Hubel and Wiesel (Hubel and Wiesel, 1962), complex cells respond to oriented bars or edges anywhere in their receptive fields, showing some tolerance to position and size of the presented stimuli.

The receptive fields of the complex cells is restricted to neighboring simple cell afferent responses to similar orientation and similar size. This way, a (limited) position- and scale-invariant representation is realized. In addition, in terms of dimensionality reduction, complex cells perform an important preprocessing step.

First, groups of simple cells are defined, containing filters of a particular size range. Within these scale bands of filters, a range \(r\) is defined that determines the size of a
matrix of neighboring simple cells of all sizes in that scale band. This matrix of simple cell responses is then fed into one of the complex cells of the second layer. Only simple cell filters of the same orientation are fed into the same complex cell in order to preserve feature specificity regarding orientation. The complex cell then performs a max-operation on its input, responding to the strongest simple cell response within the matrix. In addition, complex cells take the absolute value of the simple responses, propagating an activation ranging from zero to one corresponding to the most extreme simple cell response, thereby showing invariance to contrast reversal. Furthermore, the input matrices overlap in such a way that part of the simple cells feeding their response to a complex cell are also feeding their input to a neighboring complex cell. The responses of neighboring complex cells are then grouped together (subsampled) and propagated to the next layer.

**Algorithm**

1. Group all $s1$-images resulting from certain filter sizes together: scale bands

2. Define a pooling grid, a square of $(r \times 2)^2$ pixels, for each scale band

3. Divide the $s1$-images within each scale band into overlapping patches with size corresponding to the appropriate pooling grid

4. For each scale band,

   (a) For each orientation $\theta$,

      i. For each $s1$-image $i$ of orientation $\theta$, within the scale band,

         A. Find the highest pixel-value $\text{MAX}(\text{patch}_i)$ of each patch within $i$

      ii. Find the complex cell responses $\text{MAX}(\text{patch})$

      iii. Combine the complex cell responses into a $c1$-image, while grouping neighboring cell responses together

The Gabor-filters produce stability of the visual input. They do not classify, but transform the visual signal into another signal which on its turn also generates possibilities. The possibilities are somewhat restricted, but it is up to the later parts of the brain to do classification, or in GAI terms, to adjust the sorting machine so that some signals can continue and some not.
4.4. Neuro-physiological inspired Method for classification

4.4.2.3 Layer 3: Radial Basis Functions

Simple cells in the third layer sample the afferent complex cells of the third layer in their receptive fields. Again, retinotopically organized complex cells allow these simple cells to sample the complex cell’s output in a spatial neighborhood. In contrast to the complex cells in the previous layer, the receptive fields of simple cells in the third layer stretch over all orientations, combining bars and edges into more complex shapes.

In addition to their sampling behavior, simple cells in the third layer implement a function similar to a radial basis function (RBF) (Powell, 1987; Poggio and Bizzi, 2004). Radial basis functions are a major class of neural network model, comparing the distance between input and a prototype (Bishop, 1995). For each sampled afferent complex cell activation pattern $\mathbf{X}$, a basis function is introduced and the output is then said to be equal to the weighted sum of the basis functions:

$$ h(\mathbf{X}) = \sum_{i=1}^{n} w_i \exp(-\beta |\mathbf{X}_i - \mathbf{P}_i|^2) $$

(4.4)

$$ \beta = \frac{1}{2\sigma^2} $$

(4.5)

The simple cells in the third layer implement the above equation together, each representing one of the basis functions $\phi = \exp(-\beta |\mathbf{X}_i - \mathbf{P}_i|^2)$, where $\mathbf{P}_i$ is one of the centers of the RBF units learned in the training phase. Where $\sigma$ defines the sharpness of tuning and $\mathbf{P}$ is the center of the RBF, representing the prototype. The weighted sum is replaced by global pooling of the next layer.

During training, random patches of different sizes are drawn from an image at the level of the second layer. Each patch contains all orientations. The third layer compares these patches by calculating the summed Euclidean distance between the patch and every possible crop (combining all orientation) from the image of similar size. This comparison is done separately with each scale-band representation in the second layer.

**Algorithm**

1. During training only:
   
   (a) Select $n$ random $c1$-images
   
   (b) For each of these images,
i. For predetermined different patch-sizes
   A. Define a random position \((x, y)\)
   B. For each scale-band, store the patch \(P\) containing all orientations

2. During normal processing:

   (a) For each stored patch \(P\),

   i. For each possible patch \(X\) (combining all orientations) from a \(c1\)-image within a certain scale-band,

   A. Calculate the response of a \(s2\)-cell \(\exp\left(-\frac{1}{2\sigma^2}|X - P|^2\right)\) for each scale band

4.4.2.4 Layer 4: Global Pooling

Lastly, the fourth layer contains again complex cells, similar to those in layer two. They replace the weighted sum in equation 2.4 with a output response according to the strongest activation (minimum distance) of the simple cell afferents of the third layer in their receptive fields. The receptive fields of these cells stretch over all simple cell scales and positions. A complex cell in this layer will respond according to the most active simple cell of the previous that is selective for the same combination of oriented bars, but regardless of its scale or position.

Algorithm

1. For each \(P\)

   (a) Find the minimum distance within the scale bands, sizes, and positions

   (b) Add this distance to a feature vector
Figure 4.7: Representation of the word-image ‘Feb’ at the level of layer 2. Rows denote different scale-bands, columns denote filter orientation 0deg, 90deg, −45deg, and 45deg, respectively. Note that the actual representation contains 8 different scale-bands as opposed to the three shown here.
Table 1: Standard model Parameterization (Serre, T. et al., 2007)

<table>
<thead>
<tr>
<th>Layer 1</th>
<th>Layer 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>G(σ)</td>
</tr>
<tr>
<td>7 × 7</td>
<td>2.8</td>
</tr>
<tr>
<td>9 × 9</td>
<td>3.6</td>
</tr>
<tr>
<td>11 × 11</td>
<td>4.5</td>
</tr>
<tr>
<td>13 × 13</td>
<td>5.4</td>
</tr>
<tr>
<td>15 × 15</td>
<td>6.3</td>
</tr>
<tr>
<td>17 × 17</td>
<td>7.3</td>
</tr>
<tr>
<td>19 × 19</td>
<td>8.2</td>
</tr>
<tr>
<td>21 × 21</td>
<td>9.2</td>
</tr>
<tr>
<td>23 × 23</td>
<td>10.2</td>
</tr>
<tr>
<td>25 × 25</td>
<td>11.3</td>
</tr>
<tr>
<td>27 × 27</td>
<td>12.3</td>
</tr>
<tr>
<td>29 × 29</td>
<td>13.4</td>
</tr>
<tr>
<td>31 × 31</td>
<td>14.6</td>
</tr>
<tr>
<td>33 × 33</td>
<td>15.8</td>
</tr>
<tr>
<td>35 × 35</td>
<td>17.0</td>
</tr>
<tr>
<td>37 × 37</td>
<td>18.2</td>
</tr>
</tbody>
</table>

4.5 Methods and experiments

The experiments are performed with three types of data; normalized images and SMFs of L2 and L4, all from the same base data set of word images. All classifiers are tested on both data sets using exactly the same parameters. The experimental setup is done with the following configuration (NN refers to "nearest neighbor"):

- The amount of automatically extracted classes with 2 or more instances is 2100
- The amount of instances of word zones in total is 37811
- The amount of word zones used for training the SMFs is 100, which results in a feature vector with a size of 1800 per word zone for the L2 SMFs and 400 per word zone when L4 features are extracted.
There are many classes with a few instances, see figure 4.5. 723 classes have only two instances and 296 classes have only three. For example; 48.5% of the classes contain three or less instances, 62.7% contain five or less instances and 75.0% contain less than ten items.

NN algorithms are used since they are nonparametric methods that are well designed to handle cases where it is difficult to create a model.

The setting of the NNs are the following:

- 1-NN with Euclidean distance
- 1-NN with Hamming distance
- 1-NN with Minkowski distance = \left( \sum_{i=1}^{n} |x_i - y_i|^3 \right)^{\frac{1}{3}}
- WCENT: Standard-deviation weighted Euclidean distance for a feature vector. Every feature value \( f \) is divided by the standard deviation of that feature. During matching the Euclidean distance between two vectors is calculated. \( D = \sqrt{\sum_i d_i^2} \) with \( d_i = \frac{f - M_{\text{class}}}{\sigma_{\text{class}}} \) where \( M_{\text{class}} \) is the average.
- RCENT: The Euclidean distance between a feature vector and the average vector is taken, without weighing.

The NN experiments are 1-against-all. In the case of the classes with only two items this means there are 37710 distractors and only one attractor.

All experiments are averaged over hundreds of runs per class in a leave-one-out setting.

Many researchers who are interested in the content of the collection are interested in the most difficult of items; names of places and names of people. These items have a very low frequency. The research pays attention to this problem by focusing on retrieval systems that can work with few training examples. A 1-against-all setting is the most realistic setting for this type of retrieval if we would like to know the performance in the real world.

In a second experiment, we tested the performance of the standard model in absence of the third and fourth layer. That is, without subjecting the C1/L2-feature representation to radial basis functions. We thereby assess the additive power of representing word-images in terms of a learned feature vocabulary.
The L2-representation consists of a set of matrices, organized in accord with the different scale-bands and filter-orientations. Depending on the size of the input image, these matrices can appear in slightly different sizes. Therefore, as a nearest-neighbor classification task will be performed, these matrices need normalization (see table 2) to obtain equally sized feature vectors. For each image, we normalized the thirty-two matrices to standardized rectangular matrices (one for each scale-band), with some respect to the height- and width-ratio of word-images. Since a primary characteristic of the L2-representation is that it describes an image in different scales and orientations, this characteristic is maintained at the cost of resolving power of the L2-representation matrices for acceptable computational demand. Thus, in addition to normalizing the matrices, they are down-sampled resulting in a representation that contains 6528 feature dimensions. Note that the resulting feature dimensionality is over 15 times larger than the L4-feature representation.

<table>
<thead>
<tr>
<th>Scale Band</th>
<th>Norm. Height (pix)</th>
<th>Norm. Width (pix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
</tbody>
</table>

We used the same bootstrapping method as in the first experiment. However, where we subjected all 2100 classes in each run of experiment 1 (each run consisted of 2100 leave-one-out tests), we now took a random sample of n = 500 different classes in each run each to be tested amidst all 2099 other classes (37711 images). This choice was determined by the computational demands on performing nearest neighbor classification a feature space in which the dimensionality is many times larger as in the first experiment. Again, all experiments are averaged over hundreds of runs.
4.6 Results

The results of the experiments are summarized in the box plots. Performances on the normalized images, the unsheathed standard model implementation (L2), and the SMFs (L4) are shown in table 3, 4, and 5. On the x-axis of the box plot, the number of instances per class is projected and on the y-axis the accuracy. Accuracy is defined as the probability of obtaining class \(X\) as an answer when querying with another instance of class \(X\). As can be seen from the figures more instances implies better performance. Table 3, 4, and 5 show the means of the respective algorithms in comparison, divided over three ranges of amount of instances per class; \(2 - 10\), \(11 - 50\) and \(> 50\).

4.6.1 Summary of results

Table 2: The proportions of correctly classified instances, averaged over classes and k-fold evaluations are shown. The classes are divided into a set with extremely small numbers of instances (\(\mu < 10\)), medium numbers of instances (\(11 \leq \mu < 50\)) and large numbers of instances (\(\mu > 50\)). Best results within such a subset are printed in bold and the overall best is underlined. The L4 feature always outperforms raw image matching. Table 2 clearly shows that the SMF-features perform consistently better than the normalized images. The results of the L2 features show very good results as well when the amount of examples per class is more than 10. However, we aimed to obtain a good feature-representation that allows incremental autonomous labeling from only a few labeled examples. This aim is more likely achieved using L4-features. The additive power of radial basis functions, representing word images in terms of a learned feature-vocabulary, is found when attempting to label recently encountered word-classes. Feature-representations such as the result of the standard model are computationally demanding. They are however, very well suited for an initial boost in the process of incrementally and autonomously labeling a dataset. With the use of past experience newly encountered word-classes can be labeled quite reliably, in contrast with the use of existing feature representations that do not incorporate a learned feature-vocabulary.
It is interesting to notice the performance drop when classes contain 4 to 10 instances. It is uncertain why this happens. But, considering this phenomenon occurs both in the SMF and normalized images setting, we conclude this is not an artifact of the use of the SMFs. The performance values shown in table 2 allow us to conclude that the use of the SMFs is a viable feature extraction method for handwritten text recognition.
Regarding the setting of whole-word recognition among a myriad of classes, the difference in performance between the two tested methods is very promising. For further research on the topic addressed here, where possibly hundreds of thousands of words need to be distinguished, good representations are of utmost importance. SMFs provide a promising method of feature extraction with many open doors for specialization on handwritten text recognition. The SMFs also improve the recognition rate on items with a very low frequency. In general L4 features work better for classes with a small amount of items and L2 features for classes with more than 10 items.

4.7 Discussion

Using the word retrieval methods described in this article it is possible to bootstrap a system for handwriting recognition. Once enough labels are harvested other methods can be trained that operate on the character level. Even then the method is reliable enough to use in an ensemble of multiple classifiers. Especially the observed 89% accuracy of the 1-nearest neighbor with Hamming distance on L2 features is promising. The disadvantage of the L2 features though is that the feature vector is 16.3 times larger than the L4 feature vector and requires a lot more on the hardware side in terms of processing power and memory requirements. This disadvantage is rather large given present day equipment. This means that for bootstrapping we recommend the use of the L4 features. A possible next step in this research is to apply k-nearest neighbor, where k is a number larger than 1. Also using multi-class SVMs or hierarchical SVMs are a possibility. Combining different methods should improve both the overall retrieval capacities of the Monk system and the user satisfaction. Also a larger L4 feature vector could improve the results, but it would also make the system slower.

It can be argued that a comparison of any method to ‘raw’ image matching is an uneven match. However, the raw matching method benefited from a normalization stage that is lacking in the biologically inspired features. Another question which arises is whether support-vector machines or HMMs would perform better. For support-vector machines, we expect a performance which is marginally better than raw matching: the results of such an experiment will be largely dominated by the performance on the classes with irresponsibly low numbers of instances for this method, given the dimensionality of the feature vectors. In a similar vein, the performance of HMMs within this paradigm needs to be explored. We assume that the approach which is presented in
this paper will be a first step toward harvesting the number of labeled instances that the traditional methods require. The intrinsic disadvantage of whole-word (holistic) classification is the large number of classes that need to be trained. In the current approach, there is no re-use of available shape knowledge such as is the case in concatenated character-based HMMs. However, by applying the Viterbi-based character or syllable model construction methods from Hidden-Markov technology in an approach with intrinsically powerful invariant features, a large improvement of handwriting-recognition methodology is expected.

It would be interesting to see whether this generative and flexible system can be expanded further. At the end of the (artificial, but biologically inspired) neurological processes an ITO classification system limits the further development of the system. “How to expand beyond what the system is capable already?” would be a legitimate question for the problems described in this chapter, from a GAI perspective.

This chapter demonstrated an implementation of a single GAI cycle. The cycle was folded onto itself. The cycle starts with a generating procedure where humans annotate handwritten text. Subsequently the computer uses the few annotations it gets applying (ITO) machine learning technology. In contrast to most research the machine learning mechanism is not used as an end state, but as a generative mechanism. It is know up front that the machine learning will produce far from optimal results. Humans are then applied to select the appropriate candidates that the machine learning mechanisms generated. In the next execution of the cycle the machine learning has more data to learn from. This refinement procedure is probably a mechanism which can occur quite often in GAI systems. Such a procedure allows for the continuous improvement of a classification problem starting with little information.

Acknowledgment

The authors wish to thank the Astrophysics department of the University of Groningen and their Astro-Wise research group for making their high performance computing facilities available to AI research. Special thanks go to Edwin Valentijn and his research group for assisting us in building a search engine for handwritten text. Also the authors wish to thank Marco Wiering of Utrecht University for his assistance during the writing of the article. This work is supported by two separate research grants from the NWO
(Dutch National Scientific Organization), combined value over 1m$. 

Chapter 4. Liquid processes and phase transitions

Figure 4.8: 1-NN with Hamming distance on the normalized image.

Figure 4.9: 1-NN with Hamming distance on the L2 SMFs.
4.7. Discussion

Figure 4.10: 1-NN with Hamming distance on the L4 SMFs.

Figure 4.11: 1-NN with Euclidean distance on the normalized image.
Figure 4.12: 1-NN with Euclidean distance on the L2 SMFs.

Figure 4.13: 1-NN with Euclidean distance on the L4 SMFs.
Figure 4.14: RCENT: Negative correlation on the raw centroid match, on the normalized image.

Figure 4.15: RCENT: Negative correlation on the raw centroid match, on the L2 SMFs.
Figure 4.16: RCENT: Negative correlation on the raw centroid match, on the L4 SMFs.