Cognitive Developmental Pattern Recognition: Learning to learn DRAFT NOT FOR PUBLICATION

Tijn van der Zant[1]
Lambert Schomaker[1]
Marco Wiering[2]
Axel Brink[1]
[1]Department of Artificial Intelligence
University of Groningen
The Netherlands
{tijn,schomaker,a.a.brink}@ai.rug.nl
[2]Faculty of Computer Science
Utrecht University
The Netherlands
marco@cs.uu.nl

***Abstract*— It can be very difficult to create software systems which capture the knowledge of an expert. It is an expensive and laborious process that often results in a suboptimal solution. This article proposes an approach which is different from 'manual' knowledge construction. The described system is relevant and usable for the end user, from the beginning of its development. It is continuously being trained by the experts while they are using it. This paper shows an example and explains how the computer learns from the experts. In order to provide a general mechanism, the principle of learning to learn is proposed and applied to the problem of handwriting recognition. The study pays attention to the development of the hypotheses that the developing system uses as it adapts to feedback it receives from its own actions on itself, on objects, on the users of the system and the reflection on this feedback. The ultimate goal is the creation of a system that learns to 'google' through handwritten documents, starting from scratch with a pile of raw images.**

I. INTRODUCTION

In the creation of systems that capture the knowledge of the users quite often the focus is on how the information should be presented and on how the creator of the system can extract the information from the users. The current study aims for a different approach. The focus is to identify mechanisms by which the system may determine itself what which information should be presented to the user in order to:

- minimize the interaction (e.g., as measured in amount of mouse clicks)
- maximize the information gain for the learning algorithms

These two aspects go hand in hand. How to solve them is difficult and there is no abundant literature teaching us how to combine these two questions at once. The proposed methodology is based on current machine learning techniques to solve these problems, however, with the goal of embedding such techniques within a general and comprehensive architecture. An example is given in section V-C.

This paper does not make any claims about how humans solve these kind of problems. On the other hand the research is loosely based on current theories of human cognition. The theories from Jean Piaget seem to be an approach to the problems of constructing a view on the world by learning and adaptation in infants [13]. Since the intelligence of the current systems is not even close to humans, it is worthwhile to obtain inspiration from natural cognition.

Our goal is to develop a system which is able is to search through massive collections of connected-cursive handwritten text. Though this is not a typical cognitive task of infants, the complexity of the problem is so enormous that conventional approaches typically fail. Therefore, it may be conducive to look at the machine, as if it were a learning child. Our research does not focus only on the technical aspects of pattern recognition in hybrid intelligent systems [11], but also on the subjective part of the system itself. We call this *Cognitive Developmental Pattern Recognition*, following the naming from robotics [1]. Since much of the research discussed in this paper is ongoing, we only discuss some examples that are already working.

The rest of the document is structured as followed: In section II a brief description of our problem in handwriting recognition is given and our approach toward the problem of getting a computer to understand handwritten documents. In III an introduction is given on how humans might solve the complex task of developing hypotheses and testing them against experiences. Modern variants of the theory of Piaget [13] form the base of how the human infant constructs its cognitive procedures. Also a second core concept is

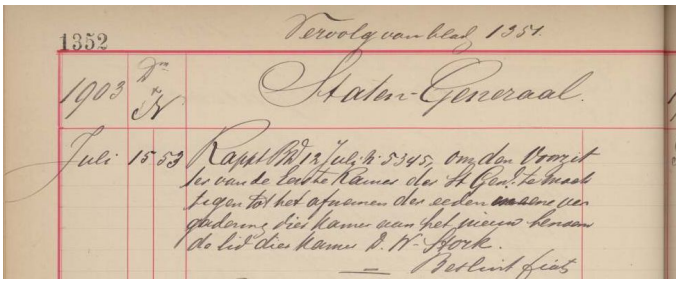

Fig. 1. An example of a part of a handwritten document from the Dutch National Archive

introduced: *behavior-based learning*, which forms the basis of how a computer can learn its own learning architecture. In section V these views are combined to show how they can be utilized to construct the necessary conditions on how a cognitive system with a specialized function can develop. An example is given on how to do machine learning on human-machine interaction and how this can be generalized for other problems.

II. HANDWRITING RECOGNITION

At the Dutch National Archive there are eight floors with handwritten documents which bookshelves measure about one hundred kilometers, all filled with historical documents. It is almost impossible to search in the archive, and it is only one of the many in the Netherlands. Together all archives, mostly of the Dutch state, form a collection of about seven hundred kilometers of bookshelves[1]. There are about 180.000 people in the Netherlands doing research into their historical backgrounds, which is about one percent of the population of the Netherlands, and who are desperately in need for digitized and searchable archives. We are developing 'googling' procedures for the collections of the Dutch National Archive, as part of a large national project (NWO/CATCH and MORPH) for improving the access to cultural heritage information.

For handwriting recognition (HWR) systems it is still not possible to read a randomly selected connected-cursive piece of text in an open context. The successful systems that do work act on simpler forms of script, on restricted content (addresses and cheque writings), or on a single writer. Our goal is to develop generic mechanisms that provide a reasonable to good performance on a wide range of script types.

Usually, machine-learning techniques such as Hidden-Markov Models [14] or a combination of smart preprocessing and template matching [12] are used for HWR research. The idea put forward in this paper is *not to design a system with a single optimized function, but to design a system that learns optimal strategies*. Today, each PhD student addresses a particular problem in HWR, and the field does not benefit from the specific solutions due to the lack of generic methodologies. Our system should learn how to learn to solve the problem at hand. Given the stereotypical patterns which are present in the process of PhD projects in HWR, it should be possible to lift up machine reading to the next level. The examples used in this paper come from HWR research, but the ideas should be applicable to more types of pattern-recognition research.

There are many problems in HWR research. An obvious one is the preprocessing of the image to a binarized version containing the essential information. Subsequently, the lines have to be cut out which raises the question of what is part of a line at all. In boxed isolated-character scripts (e.g. Chinese, Korean) this is often not an issue, but in western style scripts the 'g' for example can be connected to the character below. To recognize to which line the piece of ink belongs to one has to recognize the word, but to recognize the word one has to know where the piece of ink belongs to. For humans this is a relatively easy task, but for image-processing algorithms this is a very difficult and unsolved problem, requiring precise parameter values, thresholds and manual tuning.

After line-segmentation the connected components of ink are identified [15], which in their turn are clustered on the Blue Gene supercomputer in Groningen, the Netherlands, by means of meta-clustering. The meta-clustering algorithm looks for stable pairs that are grouped together by different clustering algorithms. These clusters can then be labeled by an expert. The labeled clusters are used for training classifiers, for the processing of unseen pages. In this paper we mainly focus on the preprocessing.

To explain Cognitive Developmental Pattern Recognition we sometimes use the metaphor of teaching a child how to play soccer. Instead of teaching the child how to kick the ball and optimize this process, it is easier for the tutor to tell the kid how it can learn for itself. Instead of telling every single move of the leg and foot one explains the concrete geometrical configuration of the a ball and a goal. The child should practice and maximize the percentage of balls hitting the goal. The tutor can give the child some advice while the child is practicing, but most of the learning is done autonomously. This way the tutor can train more individuals at the same time. The property of learning to learn, to decrease the training time, is paramount within a project as complex as HWR of connected cursive scripts. It is evident that tight-loop supervision with accurate and detailed error feedback, such as in back-propagation training of neural networks, is complicated and in any case costly when it comes to the training of the preprocessing stages.

III. PSYCHOGENESIS

Psychogenesis can be described as dealing with the origin and development of psychological processes, personality, or behavior. We do not make claims about biological plausibility. What interests us are the cognitive processes underlying the development of the behavior of intelligent systems. An essential aspect of cognitive systems is that internal processes may, may not or may not yet change the external behavior. These processes are regarded as psychological states of adaptation, searching and reflection on previous actions.

[1]Information from an informal CATCH meeting on the preservation of the Cultural Heritage in the Netherlands

As an example, rather than programming a fixed order of preprocessing steps in a fixed pipeline, we consider the actions of each processing module as part of a behavioral pattern which is controlled by parameters, which in turn are being optimized using cost evaluation.

The trainers of the system, i.e., the human users, are essential in the developmental autonomous process within the system. An important part of the internal state can be regarded as consisting of hypotheses on the outside world. The developing system has to improve on them and expand the hypothesis space with new hypotheses, which are tested on whether they are an improvement or not. The actions of the users determine whether the system assumes it is on the right track or whether is has to undertake actions to review, refine or totally change the way it interpreted its input. At the moment of writing this paper the actions of the prototypical system are fairly limited. But as the system matures, more hypotheses are generated and have to be taken into consideration.

To view a system in this way is not new, if the system is a human, the term cognitive constructivism [13] is often used. Cognitive constructivism is based on the work of Jean Piaget. The main parts of the theory are an ”ages and stages” component that predicts what children can and cannot understand at different ages, and a theory of development that describes how children develop cognitive abilities. The theory of cognitive development claims that humans cannot be ”given” information, which they immediately understand and use. Instead, humans ”construct” their own knowledge[9]. They build their knowledge through experience. Experiences enable them to create schemas (mental models in their minds). These schemas are changed, enlarged, and made more sophisticated through the complementary processes of assimilation and accommodation.

This paper proposes a way of developing such schemas, by providing the necessary building blocks that the system can use to get to the correct solution. Since the problem that is focused on is HWR research some of the building blocks might not be good for a more general solution of the learning to learn problem. One such mechanism is selective attention. Although the general mechanism of selective attention could be applicable to different research domains, our implementation will involve steps such as more processing time for the segmentation of a part of the image, the use of more clustering algorithms for meta-clustering etc.

IV. BEHAVIOR-BASED LEARNING

The overall architecture for the system is loosely based on the subsumption architecture of Brooks [3]. Brooks developed a layered control architecture where a layer is a single debugged behavior. In [4], instead of a layer consisting of a single behavior, every layer get several actions it can choose from. The goal of the system is to minimize the amount of times that the user pushed a so-called ’rubbish’ button. This button, as explained later in section V-C, is an essential aspect of the system. The creators of the system only have a vague idea of what might work and the knowledge that

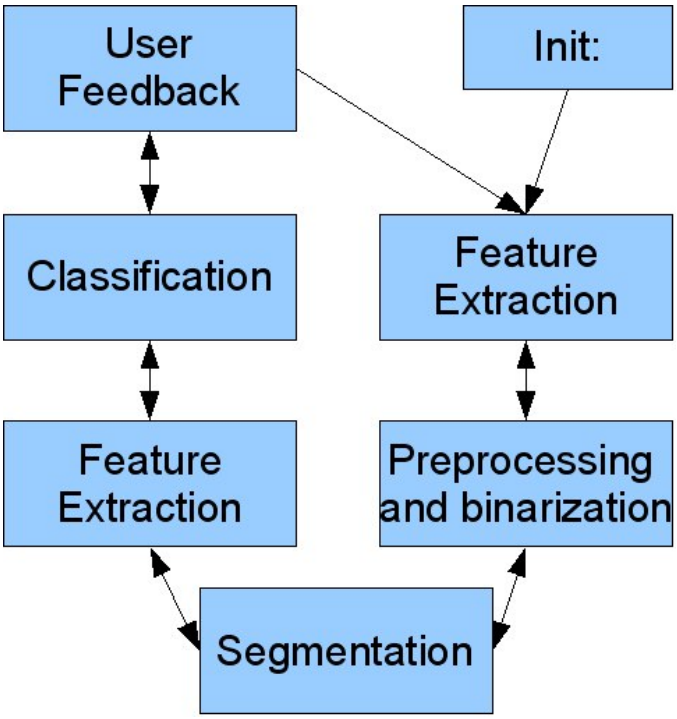

Fig. 2. The main in the software. Feedback can go through the entire system, as described in [4]

the system is capturing is very difficult if not impossible to quantify. Often the user can not explain why an answer is good or bad. But the user does know when an answer is bad. The system therefore is improving itself by getting rid of the bad answers.

The entire system is split into different behavioral layers, where a behavior can be anything ranging from performing a simple operator to a meta-clustering on the Blue Gene supercomputer we use in our research. A behavior can also draw attention to a certain region for more processing. Since some words are easier to recognize than others, and some might have features that hint at an important word (such as a capital), not all parts of a scanned document receive the same amount of processing. The layers are shown in figure 2

In the beginning of a developmental process there is no feedback yet on the end product (a correct classification of a word), but the system is bootstrapped by incremental improvements of the layers. One layer that is explained in this article is the preprocessing layer. This layer is trained by humans for increased performance on the data which appear within the operating context at hand.

In the beginning the features are created by hand to bootstrap the system. In this phase expert knowledge is put into the system, but the system decides on whether it uses it or not. In this phase of development we give the system possibilities to explore. If the system would be tabula rasa then it seems unlikely that it will come up with good features and methods in a reasonable amount of time, even with the aid of a supercomputer. The system tries several methods

and the most promising methods are explored with the use of the interval estimation algorithm [10], [4]

V. A NEW APPROACH TO CREATING EXPERT SYSTEMS

Traditionally, expert systems are created by an expert system builder who interviews experts, extract and models knowledge, and who delivers a database query system for the expert to use. Quite often it is not possible to extract the knowledge, due to problems such as that the expert does not even know what steps he or she is performing to come to a solution. It might also be that the nature of the input is too difficult to capture in a database system. The HWR problem is such a problem. Although most humans are able to learn how to read, there is no known algorithm or procedure that captures the knowledge. Also the knowledge about how humans actually read is fairly limited, from the point of view of the designer of a *Universal Reading Machine*.

A different approach is, instead of building a once-and-for-all system, to create a system where the experts give advice to the system and incrementally train it by actually using the system. There are several steps in this process.

- *Creating hypotheses:* Most parts of the system are considered as hypotheses that can be true, false of have a confidence measure. An example is the preprocessing of an image. With a random combination of filters it is likely that there are bad outcomes, such as a total black picture. Instead of programming a maximum amount of pixels that can be black before rejecting the picture, the borders are learned where a picture is always discarded by the user, but also when it is always accepted and in between. Since the system is using confidence intervals [10], [4] statements such as: 'there is a 95% chance that this picture is rejected' become possible. Some hypotheses could be rubbish, and after passing a (learnable) threshold they are always discarded by the system.
- *Bootstrapping/initial training:* After the creation of hypotheses, which is a form of modeling the expert knowledge, the system is bootstrapped either on known examples or, as in our case, by training the system in parts. The users train the system incrementally. Instead of having the computer figuring out from scratch, or from a tabula rasa state, what to do, it already got hints on how to solve the problem. The actual filling in of specific situation, parameters and actions are learned by the computer. The computer is *always* in the learning mode.
- *Using the system* After bootstrapping the system is ready to be used. It is not functioning perfectly, but it already works. Since the user can always push a button that states that he/she did not agree with the results of the system, there is always feedback if it took a wrong decision. These are key learning moments for the system and the user does not have to specify why he/she did not like the results. He/she might not even know why, but the computer will re-evaluate hypotheses and offers the next best possibility.
- *Computer generated hypotheses* Once the system has gathered enough samples it can start to generate hypotheses by itself and see if it can improve on its own performance. The cases that are the most important are the rejects by the user that are not classifiable by the hypotheses it already has. For this we plan to use reinforcement learning, partly initialized by the good hypotheses.

A. USABLE SOFTWARE FOR EXPERTS THE EXTREME PROGRAMMING WAY

Developing software is a complex process. In Extreme Programming (EP)[2] it is customary to have continuous feedback during the development of the software. The feedback ensures that unfortunate design choices are caught where they do not harm: right after the making of such an unfortunate choice. It takes much less effort to fix the choice early on. Our research takes EP to the extreme by creating a software that is using the continuous feedback from the user to improve on itself. It is better to catch a mistake from the learning software early on and use the feedback to improve continuously than to build a complete system only to find out that it did not exactly do what was expected.

Also in the development of scientific software many of the principles of EP hold [5]. One of those principles is short development cycles, in the order of a few weeks. Due to short development cycles both in the development of the functionality of the software and the hypotheses the system becomes more robust against errors, strange data, wrong usage, etc.

Other important principles are: simple design, test-driven software development, collective code-ownership, continuous integration and steering. The simple design ensure that the software remains understandable. and in our cases our c code is ready to use on the Blue Gene. In the case of scientific software test-driven development is mandatory. If you are working in a team of scientist, then you will want to check whether the software is functioning as it is meant to do, and by looking at tests and creating them one can make certain that the software is doing the right thing. The ideas of test-driven development has aided in the conception of capturing hypothesis driven expert knowledge. Collective code-ownership can only be done with the tests in place but is useful for rapid development. Continuous integration makes certain that there are no modules that require a laborious integration (which often takes as long as the development of the module). Steering of the development process is because there is no 'grand' design plan. We trust that we are smart enough to deal with the problems as they come. This does not mean there is no research plan, just that we do not plan more than a couple of weeks ahead. With the tests in place it is always easy to refactor (adapt) the software [7]. The flexible way of working with the creation of the software is what we also try to capture in the creation of knowledge.

B. MACHINE LEARNING ON HUMAN-MACHINE INTERACTION

The idea so far is to learn on the human-machine interaction (HMI) and to minimize the amount of clicks from the human. The criterion is clear (minimizing the interaction) but the way to achieve this is less clear. In the next section a method is explained how we have implemented an expert task, by implementing the possible actions an expert might undertake to solve a problem, but we let the computer figure out what exactly has to be done in this situation.

C. AN EXAMPLE: PREPROCESSING IMAGES BY FORENSIC INVESTIGATORS

1) WRITER IDENTIFICATION: At the "Nederlands Forensisch Instituut" (NFI, forensic research institute in the Netherlands) forensic researcher are looking for similarities between handwritten documents. They do writer identification. In [15] an example is given on the automated process of writer identification after binarization.

The preprocessing of an image into a binarized image can be done with image processing software such as Photoshop or the Gimp. However, it takes a lot of experimentation and knowledge of the software to get a correct binarized image[8]. Some specific filters that are applied to the input image might not even be available in these general image processing products. For example, one of the filters an expert might use deals with finding local parameters, which on their turn rely on a certain size of the matrix. This knowledge is put in the system, in the form of algorithms/filters that process the data (an image).

The filters might have some parameters, for example the size of a matrix, a threshold etc. These parameters are usually adjusted by the expert to get to the correct image. The only way the expert knows how to do this is because he/she had training and has experience. Our system only has some knowledge (the operations/filters it can perform) but is clueless to which one it should apply in what order.

Once there is a binarized image it is put into the writer identification system and that delivers a list of possible writers. They need a binarized image for this, but lack the knowledge to create one in a easy to do manner.

2) APPLYING FILTERS GENERATED BY A SPECIALIZED EVOLUTIONARY ALGORITHM: The evolutionary algorithm (EA)[6] generates sequences of filters (including parameters) to apply on the input image. Since there is no clear way to rank the individuals of the population the only viable option is tournament selection. With tournament selection a certain amount of individuals are chosen randomly from the parent population and after some form of competition the fittest goes to the next generation. In our case we first had a tournament selection size of two, but after a short while we introduced a selection size of three since many individuals deliver poor performance in the early stages of the EA. A bigger tournament size gives rise to a higher evolutionary pressure and faster convergence, but might also limit the search space.

The user sees the original picture, and three pictures that are processed by the system. Usually with tournament selection there is always one winner. After a while we realized that if all the output images are bad, then there is no use of putting these individuals in the next generation. Instead we created an "All images are bad" button that the user can press if he/she does not like the results. Whenever that button is pushed three new individuals are created and put in the parent population. However, the more the user clicks on an image (because it looked reasonable) the chance that these generated individuals are random diminishes, and the chance that the new individuals are created from the child population (with already reasonable individuals) increases.

Here the idea is captured that human feedback is important (all individuals are deleted or a single one goes through) and that the user does not need to know what is going on (mathematical operations on images) or why he/she likes what he/she sees. An interesting aspect is that the individuals (mostly university students) at first made some very poor choices. We asked them to click on a processed image if: the text was black, the background white and the written text is horizontal (not all are horizontal due to scanning variations). We did not understand why they made poor choices. But when we asked someone if he/she could also pay attention to whether he/she thinks it looks good or not the results were much better. The total amount of clicks for a certain image were reduced with approximately two thirds by these procedures.

D. LEARNING TO LEARN

The basic system and functionality are in place. The algorithm does what it has to do: minimizing the interaction between a human and a computer for binarization of an image without having expert knowledge. Now this is in place it is already ready to use by the people in the field, in this case at the forensic institute. The next steps are to let people use it, collect data and let the computer learn how it can improve on its learning.

The next steps are to use already fit individuals from previous binarizations as a seed point for the EA. This is a natural way of speeding up the system. A less obvious step is to use reinforcement learning [16] to train on the input and output images and let the reinforcement learning algorithm update individuals. This second step already comes closer to the 'learning to learn' paradigm followed by our research. Also we have just started with the creation of hypothesis spaces where we give the machine an idea on what might be good to focus on for deciding whether the processing it is doing makes sense.

A concrete example is the percentage of black pixels in the image. There is a certain percentage, unknown beforehand, where above it is certainly not text. But the exact percentage has to be learned. There might also be a minimum which has to be learned. More difficult is it for the percentages in between these two limits. For this we will use the interval estimation algorithm [10], [4] which give us a (for example) 95% confidence interval. By combining different

trained hypotheses the system knows when the chance for acceptance get below a certain threshold (a few percent) and the image is not shown to the user at all. Also the hypotheses can tell the EA that the first x steps were correct according to the hypotheses but that all steps afterward were not good (again with a degree of certainty). The EA will then only keep the first x steps, the image of those first steps is shown to the user in a tournament selection, and if chosen it gets into the child population.

Another option is the learning of the filters by means of evolutionary programming [17], [8]. If there is a baseline in place the system can train itself where improvement of the baseline is the goal. This could be regarded as a form of reflection on its own actions. The system uses knowledge from past experiences to improve its future actions.

VI. CONCLUSIONS AND FUTURE WORK

The research is still preliminary, but the results are promising. Within two months after the start there already is usable software. This software will be distributed to users who generate data needed for machine learning. Both groups (researchers and users) profit from this way of working.

Once there is enough feedback from the users the 'learning to learn' algorithms come into action, which should decrease the interaction between humans and computers even more. The ultimate goal is that the computer does not even have to ask anymore because its conclusions are certain enough.

Our future work will continue in this line with the aid of a supercomputer to go through many hypotheses and to try many different combinations of learning algorithms. A step we want to undertake this year is the use of reinforcement learning to learn which (learning-)procedures to apply for improving the system.

REFERENCES

[1] Minoru Asada, Karl F. MacDorman, Hiroshi Ishiguro, and Yasuo Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous System*, Vol. 37:185–193, 2001.

[2] Kent Beck. *Extreme Programming Explained.* Addision Wessley, 2000.

[3] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, April 1986.

[4] T. Van der Zant, M. Wiering, and J. Van Eijck. On-line robot learning using the interval estimation algorithm. *Proceedings of the 7th European Workshop on Reinforcement Learning*, pages 11–12, 2005.

[5] Tijn Van der Zant and Paul Plöger. Lightweight management - taming the RoboCup development process. *Proceedings of the RoboCup 2005 conference*, 2005.

[6] D.B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Trans. on Neural Networks: Special Issue on Evolutionary Computation*, 5:3–14, 1994.

[7] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addision Wessley, 1999.

[8] Katrin Franke. *The Influence of Physical and Biomechanical Processes on the Ink Trace*. PhD thesis, University of Groningen, 2005.

[9] H Hendriks-Jansen. *Catching Ourselves in the Act: Situauted Activity, Interactive Emergence, Evolution, an Human Thought*. MIT Press, 1996.

[10] L. P. Kaelbling. *Learning in Embedded Systems*. MIT press, 1993.

[11] Patricia Melin and Oscar Castillo. *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*. Springer, Berlin Heidelberg, Germany, 2005.

[12] Hiroto Mitoma, Seiichi Uchida, and Hiroaki Sakoe. Online character recognition using eigen-deformations. *Proceedings of the IWFHR*, pages 3–8, 2004.

[13] Sue Taylor Parker, Jonas Langer, and Constance Milbrath. *Biology and Knowledge Revisited: From Neurogenesis to Psychogenesis*. Lawrence Erlbaum Associates, London, 2005.

[14] L.R. Rabiner. A tutorial on HMM and selected applications in speech recognition. *Proc. IEEE*, Vol. 77 no. 2:257–286, 1989.

[15] Lambert Schomaker and Marius Bulacu. Automatic writer identification using connected-component contours and edge-based features of upper-case western script. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), IEEE Computer Society*, vol. 26, no. 6:787–798, 2004.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambride, MA, 1998.

[17] Astro Teller and Manuela Veloso. PADO: A new learning architecture for object recognition. *Symbolic Visual Learning*, pages 81–116, 1996.