ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 10 (2017) 1272–1279

Classifier Optimized for Resource-constrained Pervasive Systems and Energy-efficiency

Niklas Karvonen[1], Lara Lorna Jimenez[1], Miguel Gomez Simon[1], Joakim Nilsson[1], Basel Kikhia[2], Josef Hallberg[1]

[1] *Computer Science Division, Lulea University of Technology, 971 87 Lulea, Sweden*

E-mail: niklas.karvonen@ltu.se, lara.lorna.jimenez@ltu.se, miguel.gomez.simon@ltu.se, joanil@ltu.se, josef.hallberg@ltu.se

[2] *Faculty of Health and Sport Sciences, University of Agder 4879 Grimstad, Norway*

E-mail: Basel.Kikhia@uia.no

Received 27 February 2017

Accepted 15 May 2017

Abstract

Computational intelligence is often used in smart environment applications in order to determine a user's context. Many computational intelligence algorithms are complex and resource-consuming which can be problematic for implementation devices such as FPGA:s, ASIC:s and low-level microcontrollers. These types of devices are, however, highly useful in pervasive and mobile computing due to their small size, energy-efficiency and ability to provide fast real-time responses. In this paper, we propose a classifier, CORPSE, specifically targeted for implementation in FPGA:s, ASIC:s or low-level microcontrollers. CORPSE has a small memory footprint, is computationally inexpensive, and is suitable for parallel processing. The classifier was evaluated on eight different datasets of various types. Our results show that CORPSE, despite its simplistic design, has comparable performance to some common machine learning algorithms. This makes the classifier a viable choice for use in pervasive systems that have limited resources, requires energy-efficiency, or have the need for fast real-time responses.

Keywords: Classifier, Energy-saving, Parallel computing, FPGA, Microcontroller, Embedded

1. Introduction

Pervasive devices are often kept small in order for them to be unobtrusive. Reducing the physical size of a system, however, typically also reduces computational resources (e.g. memory and CPU) which affects the choices of hardware and software. It is also common for pervasive devices to be battery-powered which enforces even harder constraints upon their hardware and software design. These resource-constraints can be challenging when creating context-aware devices or services, since such systems often rely on machine learning (ML). ML algorithms often require significant computational resources (e.g. memory, CPU) that makes them difficult to implement in devices such as Field-Programmable Gate Arrays (FPGA), Application-Specific Integrated Circuits (ASIC) and low-level microcontrollers. Many such algorithms (e.g. neural networks, Support Vector Machines) also rely on

floating point operations, which further complicate such implementations[1]. This is unfortunate, since low-level components such as FPGA:s are typically highly useful for pervasive computing due to their small size, energy efficiency, fast response times, and high throughput (i.e. when pipelined).

In this work we extend our previous efforts on creating a classifier specifically targeted at low-level and resource-constrained devices[2]. We propose a Classifier Optimized for Resource-constrained Pervasive Systems and Energy-efficiency, CORPSE for short. The classifier is based on Elementary Cellular Automata (ECA) which can show complex behaviour emerging from seemingly simple rules[3]. This makes it an interesting underlying model for a classifier, since it exhibits nonlinear behaviour and it can work at the edge of chaos; a powerful realm for computations[4]. Additionally, ECA has been proven capable of universal computation (Rule 110)[5] which shows that the model holds extensive expressive power.

Another interesting property of ECA is that it does not rely on a global state for making computations. This makes it suitable for parallel processing, which can be utilized for reducing power consumption or decreasing real-time response times in pervasive computing. Furthermore, ECA requires only a binary lattice and a set of boolean algebra rules to perform its computations which allows for efficient implementations on low-level devices such as FPGA:s and ASIC:s.

Given the attractive properties of ECA combined with the challenges of performing context-recognition on low-level devices, the research questions addressed in this paper are:

RQ1: *Can Elementary Cellular Automata be extended to create a computationally inexpensive classifier by allowing varying neighborhoods across space and time?*

RQ2: *Which are the important parameters in terms of setup and training to optimize the performance of such an algorithm?*

This paper is structured as follows: section 2 reviews related work, section 3 describes the CORPSE classifier and explains how it is trained using a Genetic Algorithm, section 4 shows the evaluation of the algorithm, section 5 shows the results, followed by discussion in section 6, and conclusions and future work in section 7.

2. Related Work

Being able to run machine learning algorithms on resource-constrained devices allows for richer context-aware applications, such as determining behavioural patterns of a user[6,7]. Several efforts have been made to adapt existing machine learning algorithms for such systems. Lee et. al[8] stated that using machine learning offer promising tools for analyzing physiological signals, but that the computations involved are not well handled by traditional DSP:s. In their work they propose a biomedical processor with configurable machine-learning accelerators for low-energy and real-time detection algorithms.

Kane et. al[9] state that the computational process of Support Vector Machine classification suffers greatly from a large number of iterative mathematical operations and an overall complex algorithmic structure. Their work proposed a fully pipelined, floating point based, multi-use reconfigurable hardware architecture designed to act in conjunction with embedded processing as an accelerator for multiclass SVM classification. Other work such as DianNao[10] and PuDianNao[11], has also focused on hardware accelerating the most resource-consuming computations of common machine learning algorithms.

While most research on machine learning for use on resource-constrained devices has been focused on adapting and optimizing existing algorithms for implementations in FPGA:s, the classifier proposed in this paper is, however, designed specifically for this purpose. The simple principles of CORPSE yields a low computational complexity while using only rudimentary binary operations[2]. These properties makes it easy to implement on digital circuits and depending on the configurations of the classifier it can also be fully or partially pipelined.

The use of ECA for classification on resource-constrained devices is not a novel approach. Sev-

eral researchers have utilized the attractive properties of ECA in order to create computationally efficient classifiers. Chaudhuri et al.[12] did extensive work on additive cellular automata. Among their work we find a multiclass pattern classifier using Multiple Attractor Cellular Automata (MACA)[13]. Ganguly et. al[14] performed pattern recognition with CA:s using Generalized Multiple Attractor Cellular Automata (GMACA). Their study confirmed the potential of GMACA to perform complex computations, like pattern recognition, at the edge of chaos.

While these pattern recognition algorithms using (G)MACA:s are highly computationally efficient, they act as content-addressable memories and do not necessarily create generalized models over data. They also require that patterns belonging to the same class must be close to each other in terms of Hamming Distance[14]. This can be problematic for use in pervasive systems since data from e.g sensors or other peripherals first need to be encoded into binary patterns in a way that upholds this requirement (e.g. Gray code[15], thermometer encoding[16] or scatter code[17]). Our proposed algorithm does not require any such encoding so the input data can be fed straight from any sensors to the classifier.

Cellular Automata Neural Networks (CANN) are another example of where principles of cellular automata are used to create classifiers[18]. CANN:s have shown to be successful in, for example, image processing[19], fluid dynamics[20] and statistical physics[21]. Unfortunately, CANN:s still rely on floating point values to perform its computations, which complicates the implementations in digital logic. However, the CORPSE classifier is not limited in this way.

3. CORPSE Algorithm

3.0.1. CORPSE working principles

CORPSE's basic principle is similar to an ECA with the difference that each cells neighborhood can vary spatio-temporally (See figure 1). The rationale behind this is that it creates a function similar to a sparse neural network where each neuron has three incoming connections of unit weight and uses the ECA rule as an activation function. This creates a novel combination of neural networks and cellular automata that aims to add powerful computational properties from neural networks to ECA. Since the ECA neighborhood varies in space, it enables a cell A to shortcut its interactions with another cell B in the automata without having to propagate changes through the cells between them during multiple timesteps. By allowing the neighborhood to also connect across timesteps, cell A can also interact with any previous state of cell B. This enables to have temporal dependencies between cells in the network. Each cell also has the possibility to connect to bias cells which have a constant value of zero or one.

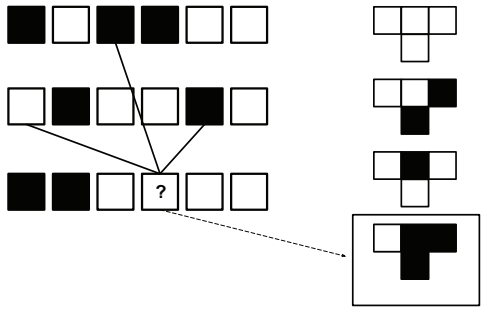

Fig. 1. The principle of determining neighborhood and updating of a CORPSE neuron/cell.

Since most researchers are likely more familiar with neural networks than with cellular automata, we will introduce the notion of CORPSE as a binary neural network. CORPSE can be viewed as a sparsely connected binary neural network where each neuron has three connections to neurons in previous layers. Each connection has unit weight, i.e. only copies the value from the connected cell. The activation function used for the network can be any ECA rule.

The CPU operations required to calculate a neuron's state, consists of evaluating a boolean expression (the activation function). E.g. for rule 110 that is used in this work:

$$q = (\neg a \wedge b) \vee (b \wedge \neg c) \vee (\neg b \wedge c) \qquad (1)$$

Where q is the next state for a neuron with incoming connections a, b, c. Each update thus requires 8 instructions when implemented on a CPU

(assuming AND, OR and NOT are part of the CPU instruction set and are atomic).

To perform a classification, all features' bit patterns are loaded onto the input layer. Then the network is run in a forward-propagation manner which will produce a bit pattern at the output layer. This output pattern is then compared to a set of bit patterns that corresponds to each class. The comparison between the output pattern and the classes bit patterns is performed using a binary similarity measure. The class with the highest similarity to the output pattern is chosen.

3.0.2. CORPSE Hyper parameters

There are five different setup parameters for CORPSE:

1. The number of network layers.
2. Whether to allow neurons' connections to skip over layers (i.e. connect to a neuron in any previous layer)
3. The bit pattern for each class (that the output pattern will be compared to).
4. The similarity measure used to compare the output pattern to the classes bit patterns.
5. The cellular automata rule to use as activation function for the network.

Using a deeper network model is assumed to allow for a higher degree of expressiveness for the classifier model. A deeper model, however, results in a larger search space for the Genetic Algorithm (GA) to explore during training. The number of layers also affect the computational resources required for running the algorithm.

Since CORPSE is based on ECA, the computational expressiveness could be limited when only a small number of layers is used. By allowing neurons to connect to a neuron in any previous layer and not just the adjacent (previous) layer, the expressive power of the classifier could increase due to introducing the ability of having temporal dependencies between neurons. It should be noted, however, that using this setting complicates a fully pipelined implementation of the classifier.

The bit patterns assigned to the classes interacts with the GA fitness model and the output similarity measure. Together, these parameters make up the discriminative function of the classifier. In order to achieve optimal results, these parameters should be tuned to fit the classification problem at hand.

There exists 256 different rules for ECA that can be used for CORPSE. While these rules have known characteristics in ECA, it is not clear how they affect the behaviour of CORPSE when the neighborhood configurations are different. Most likely, the activation rule should be chosen with regards to the comparative bit patterns, the output similarity measure, and the dataset.

3.0.3. Training

Training is performed using a GA where an individual represents all connections between cells in the network. The fitness of an individual is the total amount of correct classifications the individuals network gets on the training set. Regeneration is performed using tournament selection and elitism is used to prevent degeneration. The least fitted individuals are culled and replaced with random individuals. Uniform crossover is performed by randomly picking connections from the parents and assigning these to the offspring. The probability for an individual to mutate is based on the mutation rate. If an individual is chosen to mutate, a selected amount of its connections will be randomly changed (with the constraint that each cell must connect to a cell in a previous layer).

4. Evaluation of CORPSE

4.1. Evaluation setup for the CORPSE algorithm

4.1.1. Implementation

In order to evaluate the algorithm, CORPSE was implemented in Java to work with the WEKA machine learning suite[22]. This choice allowed the algorithm to be compared with other machine learning algorithms whose implementation and performance is

known within the research community. Five classifiers of different types (Bayesian, Tree, Neural network, Ensemble) were chosen to be used for comparison with CORPSE's results. The choice of having multiple algorithms as a baseline was based on the fact that some datasets are better classified with a certain type of classifier. Using several algorithms for comparison reduces the effect of this while also giving readers a better bases for further reasoning about the performance of CORPSE.

4.1.2. GA Training Parameters

In an effort to find well-performing parameters for the GA, a series of 192 tests were performed varying: number of generations, elite size, tournament size, crossover type(uniform, single-point), and mutation rate. Average fitness plots were used to validate convergence of the GA. From these experiments, the best overall performing parameters were selected:

- 400 generations
- 100 population size
- 85 tournament selected individuals
- 2 elites
- 13 individuals culled
- Uniform crossover
- 90 percent mutation rate (amount of connections changed in a mutation was randomly chosen between 1-5)

4.2. Datasets and Feature Selection

Due to our earlier work and findings[2], this article focuses on binary classification problems. 8 different datasets (See Table 1.) were used for evaluating the proposed algorithm. Datasets 1-7 were chosen since they included with WEKA, which makes it easy to reproduce our results. They also allow for a comparison with our previous efforts[2]. For more information about these datasets, please refer to the WEKA documentation. Dataset 8 was chosen based on that it consists of an activity recognition task using accelerometer data from a wrist-worn device[23]. This was considered a good use-case of the proposed algorithm. It is also significantly larger than datasets 1-7. In order to avoid overfitting and to put a resource-constraint on the feature selection, the three highest ranked features for each dataset were selected using InfoGain[24].

4.3. Evaluating hyper parameters

A series of tests were performed varying different hyper parameters of CORPSE to analyze its performance. All these tests were performed using 10-fold cross validation. Following are the the different tests and their results.

4.3.1. Allowing connections to skip layers

This test aimed at exploring if an increased expressive power of the classifier could enhance performance. Results showed that allowing neurons to connect (skip) over layers did, in fact, have a positive effect on the performance.

4.3.2. Using bias neurons

Since the GA can choose whether to connect "regular" neurons to bias neurons or not, the hypothesis in this test was that using bias neurons should always yield a better result. Our tests, however, showed that using bias could actually degrade performance slightly.

4.3.3. Using different bit patterns for classes

Two different models were used for generating the bit patterns for each class. The first one was to randomly choose the bit pattern belonging to each class. This choice was based on the assumption from hyper-dimensional computing that two random binary vectors is approximately orthogonal if their dimensionality is in the thousands. Although the dimensionality in this paper only reaches the hundreds, this model was still considered an interesting setup since the two patterns will still likely be discriminative in hamming space.

The second choice was to use a equidistant and maximized hamming distances between each pattern. This was achieved by dividing the total pattern length in two and filling the first half with ones for

class A and the second half with ones for class B. This setup was also chosen to make the two patterns be discriminative in hamming space.

The test's results indicates that the difference between using maximum hamming distance patterns compared to using random patterns, is very small. This parameter gave an inconclusive result and also had an varying effect depending on the dataset.

4.3.4. Using different output compare models

Two different similarity measures, Hamming Distance and Jaccard Similarity Coefficient, were used in order to determine which class the output layer is closest to. Surprisingly, the difference between using the two output compare models showed little differences in our tests.

4.3.5. ECA rule (activation function)

There was no evaluation of how different rules affect the performance of the algorithm. Instead, the ECA rule was kept fixed to rule 110 throughout this paper. This choice was based on the fact that rule 110 is the only ECA rule that has been shown capable of universal computing.

4.3.6. Varying the amount of network layers

In this last test, the best found hyper parameters from the previous tests were used while varying the network layer size between 2-6 layers (See table 2). The hypothesis was that a larger amount of layers could yield a better performance due to a higher expressive power for the classifier. As can be seen from the results, however, the optimal number of layers changes with the dataset and no clear conclusion can be drawn from this.

5. Results

The comparison between CORPSE and the comparative classifiers (see Table 1) was performed using the following hyper parameter setup found by the hyper parameter experiments:

- Allowing connections across layers
- Using bias nodes
- Using random bit patterns for class comparisons
- Output compare model set to Jaccard similarity coefficient

Results show that CORPSE is performing with an accuracy and kappa statistics similar to the comparative algorithms. In all datasets, CORPSE is outperforming one or more of the comparative algorithm. For dataset 1, CORPSE is even outperforming all of the comparative algorithms. It should be noted, however, that all classifiers used a standard setting and were not setup in any way to fit each dataset. The results for the classifiers in this paper, including CORPSE, should therefore not be considered to be optimal.

Given the results, it is clear that an extended ECA can be used in order to create a computationally inexpensive classifier (RQ1). Allowing neurons to connect across layers showed an improvement in performance, while adding bias nodes sometimes reduced performance slightly. No clear conclusion could be drawn from how the number of layers, the output patterns and the output compare models affect the performance. (RQ2).

6. Discussion

In the light of how simplistic the CORPSE algorithm's working principle is, its results in comparison with other more advanced ML algorithms are remarkable. It should also be noted that the performance could be further increased by using other output compare models, bit patterns for classes, or fitness functions. These could also be tailored to fit the underlying classification problem.

Results also show that our different choices of output compare models and classes' bit patterns, had a negligible effect on the results. This result is counter-intuitive but could possibly be explained by that the discriminative ability of each of these choices are similar. E.g. if two sets of points are sufficiently far apart, it makes little difference if one uses a line or a curve to separate them.

Table 1: The classification results for the algorithms on the different datasets.

Datasets	**CORPSE**	Naive Bayes	Bayes Net	Multi-layered Perceptron	K-star	J48	Random Forest
1 Breast Cancer	**73.77%**	71.8%	72.08%	70.14%	73.36%	71.86%	69.30%
2 Labour	**78.94%**	85.17%	78.5%	84.37%	81.90%	82.33%	84.30%
3 Vote	**94.71%**	94.71%	94.71%	95.08%	94.71%	95.63%	95.24%
4 Diabetes	**73.82%**	76.39%	75.57%	76.08%	73.52%	74.65%	72.21 %
5 Ionosphere	**84.61%**	86.82%	89.15%	87.56%	89.66%	90.17%	90.06%
6 Supermarket	**65.24%**	63.71%	63.71%	63.71%	63.71%	63.71%	63.71%
7 German-Credit	**73.60%**	73.94%	73.01%	73.5%	72.86%	72.06%	70.97 %
8 Strong and Light	**76.45%**	75.59%	79.23%	77.86%	81.30%	82.41%	81.68%

Table 2: **CORPSE**: Results of Accuracy / Kappa statistic from Layer2 - Layer6

Datasets	LAYER2	LAYER3	LAYER4	LAYER5	LAYER6
1 Breast Cancer	73.37%/0.2379	72.72%/0.2446	73.42%/0.2522	70.97%/0.2055	**73.77%/0.2080**
2 Labour	**78.94%/0.5378**	64.91%/0.2297	78.94%/0.5270	64.91%/0.2470	70.17%/0.3377
3 Vote	**94.71%/0.8896**	94.71%/0.8896	94.71%/0.8896	94.71%/0.8896	94.48%/0.8846
4 Diabetes	73.82%/0.3749	72.91%0.3747/	**73.82%/0.3985**	73.43%/0.3933	73.43%/0.3845
5 Ionosphere	**84.61%/0.6610**	84.04%/0.6434	82.33%/0.6121	84.33%0.6504	83.76%/0.6389
6 Supermarket	**65.24%/0.119**	65.16%/0.1181	65.16%/0.1181	65.24%/0.119	65.24%/0.119
7 German-Credit	72.40%/0.2367	**73.6%/0.2731**	73.40%/0.2756	72.70%/0.2573	72.70%/0.2573

Another counter-intuitive result is that using bias nodes actually could degrade the performance. The reason for this might be that the bias nodes could be utilized by the GA to quickly achieve a relatively high fitness, but ending up in a local optima. This reveals a possible flaw in the fitness function since, theoretically, the addition of bias nodes should not degrade the performance. If the fitness function and GA works properly, it should simply disregard connecting neurons to the bias nodes if it is not beneficial for the final solution.

In our results no clear conclusion could be drawn from how the amount of layers affect the performance of the algorithm. A reason for this could be that the fitness function of the GA was too coarse. We can consider an example where there exists two different candidates which both would increase the number of correct classifications by an equal amount. One of these could, however, be closer to the global optima but our fitness function would fail to recognize this and consider them equal. If the numbers of layers are few, the effect of this could be less significant due to the amount of candidates that give rise to equal fitness are potentially fewer. With this in mind, a different fitness function should be designed in order to properly evaluate the possibilities of using deeper network structures.

7. Conclusion

In this paper we present the CORPSE algorithm, a computationally inexpensive classifier that is targeted for resource-constrained devices. We have shown that it has comparable performance to other well known algorithms using eight different datasets. The computational properties of the algorithm are interesting and it is possible that the algorithm could be used also for regression, filtering, or feature extraction. Our experiments also revealed a possible flaw in the fitness function of the genetic algorithm used for training. Future work should therefore primarily focus on designing a better fitness function or finding alternative training methods. This could possibly improve performance by, for example, better utilizing deeper network structures.

References

1. Xie Zhen-zhen and Zhang Su-yu. A non-linear approximation of the sigmoid function based fpga. In *Proceedings of the 2011, International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19–20, 2011, Melbourne, Australia*, pages 125–132. Springer, 2012.
2. Niklas Karvonen, Basel Kikhia, Lara Lorna Jiménez, Miguel Gómez Simón, and Josef Hallberg. A computationally inexpensive classifier merging cellular automata and mcp-neurons. In *Ubiquitous Computing and Ambient Intelligence: 10th International Conference, UCAmI 2016, San Bartolomé de Tirajana, Gran Canaria, Spain, November 29–December 2, 2016, Part II 10*, pages 368–379. Springer, 2016.
3. Palash Sarkar. A brief history of cellular automata. *ACM Computing Surveys*, 32(1):80–107, Mar 2000.
4. Chris G Langton. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1-3):12–37, 1990.
5. Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40, 2004.
6. Basel Kikhia, Thanos G Stavropoulos, Stelios Andreadis, Niklas Karvonen, Ioannis Kompatsiaris, Stefan Sävenstedt, Marten Pijl, and Catharina Melander. Utilizing a wristband sensor to measure the stress level for people with dementia. *Sensors*, 16(12):1989, 2016.
7. Anders Hedman, Niklas Karvonen, Josef Hallberg, and Juho Merilahti. Designing ict for health and wellbeing. In *International Workshop on Ambient Assisted Living*, pages 244–251. Springer, 2014.
8. Kyong Ho Lee and Naveen Verma. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE Journal of Solid-State Circuits*, 48(7):1625–1637, 2013.
9. Jason Kane, Robert Hernandez, and Qing Yang. A reconfigurable multiclass support vector machine architecture for real-time embedded systems classification. In *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, pages 244–251. IEEE, 2015.
10. Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ACM Sigplan Notices*, volume 49, pages 269–284. ACM, 2014.
11. Daofu Liu, Tianshi Chen, Shaoli Liu, Jinhong Zhou, Shengyuan Zhou, Olivier Teman, Xiaobing Feng, Xuehai Zhou, and Yunji Chen. Pudiannao: A polyvalent machine learning accelerator. In *ACM SIGARCH Computer Architecture News*, volume 43, pages 369–381. ACM, 2015.
12. Parimal Pal Chaudhuri. *Additive cellular automata: theory and applications*, volume 1. John Wiley & Sons, 1997.
13. Pradipta Maji, Niloy Ganguly, Sourav Saha, Anup K. Roy, and P. Pal Chaudhuri. *Cellular automata machine for pattern recognition*, pages 270–281. Cellular Automata. Springer, 2002.
14. N. Ganguly, P. Maji, B.K. Sikdar, and P.P. Chaudhuri. Design and characterization of cellular automata based associative memory for pattern recognition. *IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics)*, 34(1):672–678, Feb 2004.
15. James R. Bitner, Gideon Ehrlich, and Edward M. Reingold. Efficient generation of the binary reflected gray code and its applications. *Communications of the ACM*, 19(9):517–521, Sep 1976.
16. Paul Crook, Stephen Marsland, Gillian Hayes, Ulrich Nehmzow, et al. A tale of two filters-on-line novelty detection. In *Robotics and Automation, 2002. Proceedings. ICRA 02. IEEE International Conference on*, volume 4, pages 3894–3899. IEEE, 2002.
17. D. Smith and P. Stanford. A random walk in hamming space. pages 465–470 vol.2. IEEE, 1990.
18. Leon O Chua and Lin Yang. Cellular neural networks: Applications. *IEEE Transactions on circuits and systems*, 35(10):1273–1290, 1988.
19. Alper Basturk and Enis Gunay. Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm. *Expert Systems with Applications*, 36(2):2645–2650, Mar 2009.
20. Sndor Kocsardi, Zoltan Nagy, Arpad Csik, and Peter Szolgay. Simulation of 2d inviscid, adiabatic, compressible flows on emulated digital cnn-um. *International Journal of Circuit Theory and Applications*, 37(4):569–585, May 2009.
21. M. Ercsey-Ravasz, T. Roska, and Z. Nda. Statistical physics on cellular neural network computers. *Physica D Nonlinear Phenomena*, 237(9):1226–1234, Jul 2008.
22. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.
23. Basel Kikhia, Miguel Gomez, Lara Lorna Jiménez, Josef Hallberg, Niklas Karvonen, and Kåre Synnes. Analyzing body movements within the laban effort framework using a single accelerometer. *Sensors*, 14(3):5725–5741, 2014.
24. Veronica Bolon-Canedo, Noelia Sanchez-Marono, and Amparo Alonso-Betanzos. A review of feature selection methods on synthetic data. *Knowledge and information systems*, 34(3):483–519, 2013.