A Comparison of Evolvable Hardware Architectures for Classification Tasks

Kyrre Glette[1], Jim Torresen[1], Paul Kaufmann[2], and Marco Platzner[2]

[1] University of Oslo, Department of Informatics,
P.O. Box 1080 Blindern, 0316 Oslo, Norway,
{kyrrehg,jimtoer}@ifi.uio.no
[2] University of Paderborn, Department of Computer Science,
Warburger Str. 100, 33098 Paderborn, Germany
{paul.kaufmann,platzner}@upb.de

Abstract. We analyze and compare four different evolvable hardware approaches for classification tasks: An approach based on a programmable logic array architecture, an approach based on two-phase incremental evolution, a generic logic architecture with automatic definition of building blocks, and a specialized coarse-grained architecture with pre-defined building blocks. We base the comparison on a common data set and report on classification accuracy and training effort. The results show that classification accuracy can be increased by using modular, specialized classifier architectures. Furthermore, function level evolution, either with predefined functions derived from domain-specific knowledge or with functions that are automatically defined during evolution, also gives higher accuracy. Incremental and function level evolution reduce the search space and thus shortens the training effort.

1 Introduction

Evolvable hardware (EHW) has a variety of applications, one of which is classifier systems. A number of studies report on the use of EHW for classification problems such as character recognition [1], prosthetic hand control (PHC) [2], sonar return classification [3], and face image recognition [4]. These studies have demonstrated that EHW classifiers can outperform traditional classifiers such as artificial neural networks (ANNs) in terms of classification accuracy. For the electromyographic (EMG) signal classification, [5] showed that EHW approaches can perform close to the modern state-of-the-art classification methods such as support vector machines.

Furthermore, there are other performance metrics, such as classification speed, training speed, and resource requirements, where EHW might excel over traditional methods. Classifiers can benefit from online adaptation and are thus an interesting target for studying online adaptive EHW schemes. An important aspect of online adaptive systems is the adaptation time, which for EHW systems equates to evolution time. Important factors contributing to evolution time are the size and the complexity of the search space. Approaches to address these

challenges include variable length chromosomes [6], incremental evolution [3], function-level building blocks [7] and automatic definition of building blocks [5].

The novel contribution of this paper is the analysis and comparison of four EHW classifier architectures. The architectures differ mainly in the building blocks used for evolution and their overall structure. We measure qualities such as classification accuracy and evolution speed. Generally, it is difficult to compare performance figures for EHW classifiers from literature, since data sets and evaluation schemes differ. We therefore propose and rely on a common data set obtained from the classification of EMG signals for PHC.

The paper is structured as follows. Section 2 describes approaches to EHW classification, especially for the domain of PHC. The selected EHW approaches are detailed in Section 3. The experimental results are given and discussed in Section 4. Finally, Section 5 concludes the paper.

2 EHW Classifiers

2.1 Related Work

An early use of EHW for classification was reported in [1]. Originally, the architecture was applied for character classification but later on used for classification in a prosthetic hand controller [2,8]. The classifier architecture is a programmable logic array (PLA)-like structure of AND gates followed by OR gates. The configuration of the architecture was evolved using a genetic algorithm (GA) implemented on the same chip as the classifier, resulting in an online adaptable system. The system had 16 input feature bits, could classify six different categories, and its classification accuracy was shown to be competitive to an ANN.

Experiments on two-phase incremental evolution of an EHW architecture applied to PHC were presented in [9]. The two-phase approach consists of first evolving category subsystems separately and then assembling them in a second phase. The results showed that the approach can lead to a better generalization performance than both traditional direct evolution and ANNs.

An online EHW architecture for classification tasks was proposed in [10,11]. The architecture was applied to multiple-category face image recognition and sonar return classification. The evolution part of the system is implemented on an FPGA, where fitness evaluation is carried out in hardware and the evolutionary algorithm (EA) runs on an on-chip processor. The architecture employs function level modules as well as a method of dividing the evolution into several smaller tasks. Later, the same architecture was also applied to PHC and compared to another approach based on embedded cartesian genetic programming (ECGP) [5]. The ECGP-based approach uses automatic definition of sub-functions, and achieves good classification accuracies despite the fact that evolution is performed on a more general architecture with lower level primitives.

EHW classification architectures applied to domains other than PHC include, for example, the function level evolution of [7]. This architecture was applied to typical ANN applications, however, with fewer inputs and outputs, and attained

accuracies comparable to ANNs. A different EHW pattern classification system, Logic Design using Evolved Truth Tables (LoDETT), was presented in [3,4]. LoDETT allows for high accuracy classification on problems with a much higher number of inputs and outputs. An example is face image recognition with 512 inputs and 40 different categories. Although providing high classification accuracy, also outperforming an ANN, the system lacks the ability of online evolution and relies on synthesis in software before the circuit is implemented on a field-programmable gate array (FPGA). The approach utilizes incremental evolution, i.e., sub-circuits are evolved separately before being assembled into a final system.

2.2 EMG Reference Data Set

We have defined an EMG data set for PHC as common reference for testing all EHW classifier approaches. A test subject has been equipped with EMG sensors and asked to perform different hand movements repeatedly. We have collected signals from four sensor channels. The signals are categorized into the eight different movements: *open*, *close*, *flexion*, *extension*, *ulnar deviation*, *radial deviation*, *pronation* and *supination*. The raw EMG signals have been preprocessed following the approach of Kajitani et al. [8] which smoothes the signal using the RMS method and averages the amplitudes over one second. Since we measure four channels, the feature vector for every movement is represented by a tuple of four numbers. Overall, we have recorded 20 data sets per category, and repeated the experiment on three consecutive days. The experimental setup for the signal acquisition and preprocessing are described in detail in [5].

3 Selected Architectures

This section presents the selected EHW classifier approaches and discusses their characteristics. We have chosen to compare a PLA-based architecture, an architecture using increasing complexity evolution, an ECGP-based system, and the functional unit row (FUR) approach. It would also have been interesting to compare with the LoDETT system [3]. However, LoDETT has not yet been applied to PHC.

3.1 PLA-based Architecture

In [2] a self-contained EHW system running an on-chip GA was presented. The genome's representation model is based on a PLA architecture, as shown in Figure 1. The PLA approach implements a sum of products representation: Each of the circuit's primary inputs and their negates can be connected to a set of AND gates. Accordingly, the AND gates' outputs can then be connected to a set of OR gates. The sum of products, which corresponds to the detection of a single category, is then represented by the output of a single OR gate. A genome from the population memory configures such a PLA by defining the fuse

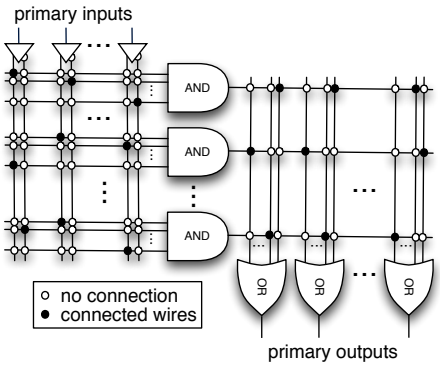

Fig. 1. PLA-based architecture.

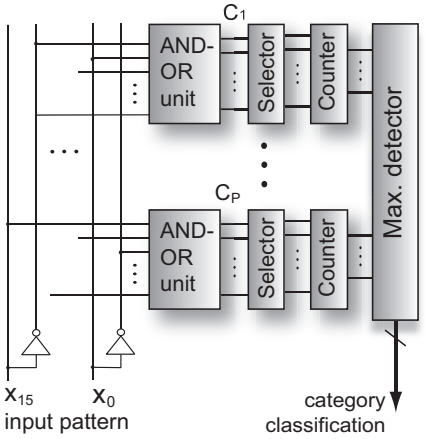

Fig. 2. Increased complexity evolution architecture.

settings at the intersections of primary input lines and inputs of the AND gates, and output lines of the AND gates and inputs of the OR gates, respectively. A classifier system is then constructed by assigning a primary output to a category. Having this, the simultaneous and thus erroneous recognition of multiple hand movements is solved by selecting the category with the lowest index.

We have reconstructed this PLA-based architecture from [2] which supposes a genome length of 2048 bits, and from [8] which determines a possible input/output configuration as 4 channels of 4 bits, each coding for 6 different categories. This gives 32 lines of primary and negated inputs to the AND gates. Given that there are 6 OR gates, one can have a maximum number of 53 AND gates from a genome of 2048 bits. We have expanded the PLA-based architecture to 8 OR gates in order to be able to classify 8 categories. Keeping the same number of input lines and AND gates, we result at a genome of $32 \times 53 + 8 \times 53 = 2120$ bits. This reconstruction will for the rest of the paper be referred to as the "PLA" architecture. The fitness is measured on the number of correct outputs from the classifier after being presented to all training vectors.

3.2 Increased Complexity Evolution (ICE) Architecture

The increased complexity evolution approach (also referred to as incremental evolution) was introduced by Torresen in [9]. The experiments were based on software simulations but a hardware implementation of the architecture, either offline or online, would be straightforward since the number of digital building blocks and their locations are fixed. An overview of the architecture can be seen in Figure 2. The classifier system is divided into subsystems, one for each classification category, which are connected to a maximum detector. Each subsystem is in turn divided into an AND-OR unit, a selector unit and a counter. The

AND-OR unit consists of a layer of AND gates followed by a layer of OR gates. Each gate has a fixed number of inputs which are programmable: The AND gates can connect to any bit from the input data and the OR gates can connect to any output from the AND gates. The outputs from the OR gates are fed into the selector unit, which selects which of these outputs are to be counted by the counter. The number of asserted lines from the selector are thus passed to the maximum detector, which in turn decides which of the subsystems, that is, which category, has the highest value. In the case of a tie, the subsystem with the lowest index is chosen.

The system is evolved in two phases. First, the AND-OR subsystems are evolved separately, one at a time. All training vectors for all categories are applied to each subsystem, and fitness is measured on the outputs of the OR layer. If the category of the training vector corresponds to the category of the subsystem, a high number of activated OR gates is rewarded by adding the number of activated gates to the fitness value. In addition, the number of activated gates are multiplied by a given emphasize value of 4 in order to emphasize the activation of the current subsystem, which in turn was found to speed up evolution. In the opposite case, where the categories do not correspond, not activated OR gates are rewarded in a similar manner, but without the emphasize value. In the second step, the subsystems are assembled and evolution is performed on the selector units, measuring the outputs from the complete system. Now, the fitness value is incremented in the cases where the system output corresponds to the category belonging to the applied training vector. In addition, in [9] experiments were made which showed that including only half of the OR gate outputs for the fitness evaluation in the first phase gives better generalization ability.

We have selected the same system parameters as in [9], with 3 inputs to each gate, 32 gates in each AND and OR layer, and 16 "floating" OR-gates. With an input of 32 lines, one then has $5 \times 3 \times 32 \times 2 = 960$ genome bits for each subsystem in the first step of the evolution. The second phase for evolving the selectors gives, with 8 categories, $32 \times 8 = 256$ genome bits. This architecture will for the rest of the paper be referred to as the "ICE" architecture.

3.3 The Embedded Cartesian Genetic Programming (ECGP) Architecture

Figure 3 shows the structure of the ECGP architecture which consists of a number of category detection modules (CDMs), summation blocks and a maximum detector. The CDMs split into a number of category detectors and are responsible for the classification of one category. In essence, each category detector represents an independent classifier structure. The single category detector is a digital circuit evolved within the ECGP model. ECGP is an extension of the popular FPGA-oriented cartesian genetic programming (CGP) model [12]. CGP is a structural hardware model that arranges logic cells in a two-dimensional geometric layout. An evolved circuit consists of a number of primary inputs, a number of logic blocks, and a number of primary outputs. ECGP extends CGP by relaxing the strict geometric layout constraints and by adding the automatic definition

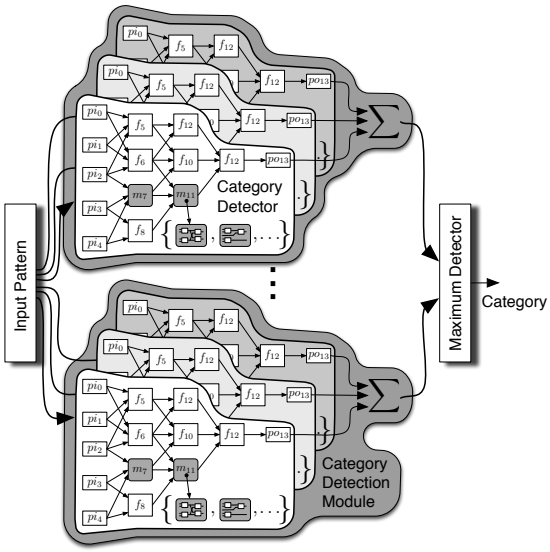

Fig. 3. Embedded cartesian genetic programming architecture

and reuse of sub-functions (modules) [13]. While primitive nodes correspond to basic gate functions, modules are defined as compositions of primitive nodes. The size of a module is restricted, which also restricts the maximal genome size.

As a fitness metric for the training phase we use the reciprocal square error distance to the predictions of a perfect classifier. Similar to [13], we have chosen a standard $1 + 4$ evolutionary strategy (ES) as the optimization algorithm. The population is initialized randomly with circuits that comprise 10 logic blocks. Depending on the created modules, the chromosome of a single category classifier is allowed to grow up to 250 logic blocks. While the chromosome's size can vary between 560 and 17.685.000 bits, starting with the shortest configuration the chromosomes grow on average up to 678 bits.

Our ECGP-based classifier system evolves twelve classifier circuits for each movement (category). For each feature vector and category, we calculate the maximum of activated classifier circuits and take the category with the most activations as a result. In case of a tie, the CDM with the lowest index is chosen. For the rest of the paper, this architecture will be referred to as the "ECGP" architecture.

3.4 The Functional Unit Row (FUR) Architecture

Like the PLA architecture, the following architecture is designed for online evolution. To facilitate online evolution, the classifier architecture is implemented as a circuit whose behavior and connections can be controlled through configuration registers. By writing the genome bitstream produced by the GA to these registers, one obtains the phenotype circuit which can then be evaluated.

The classifier system consists of CDMs, one for each category to be classified. These equate to the subcircuits in the ICE architecture as well as to the CDMs in the ECGP architecture, and are connected to a maximum detector. The input data to be classified is presented to each CDM concurrently on a common input bus. The CDM with the highest output value will be selected by the maximum detector, and in the case of a tie, the category with the lowest index is chosen. Each CDM consists of M FURs – "rules" or rows of functional units (FUs). See

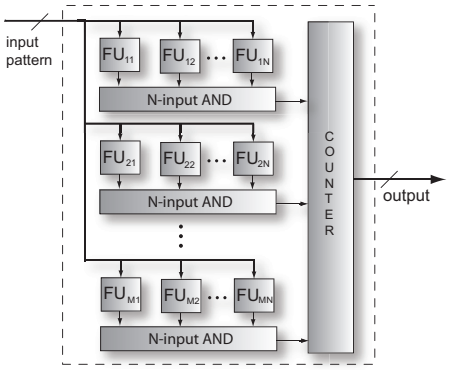

Fig. 4. Category detection module

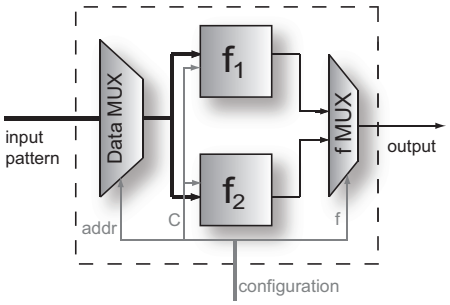

Fig. 5. Functional unit

Figure 4. These FURs equate to the category detectors evolved by the ECGP approach. Each FUR consists of N FUs. The inputs to the circuit are passed on to the inputs of each FU. The 1-bit outputs from the FUs in a row are fed into an N-input AND gate. The 1-bit outputs from the AND gates are connected to an input counter which counts the number of activated FURs. As the number of FURs is increased, so is the output resolution from each CDM. Each FUR is evolved from an initial random bitstream, which ensures a variation in the final evolved FURs.

The FUs are the reconfigurable elements of the architecture. As seen in Figure 5, each FU behavior is controlled by configuration lines connected to the configuration registers. Each FU has all input bits to the system available at its inputs, but only one data element (e.g., one byte) of these bits is selected from the input bits, depending on the configuration lines. This data is then fed to the available functions, which, for the EMG classification, will be detailed below. In addition, the unit is configured with a constant value, C, used together with the input data element to compute the output from the selected function.

The functions chosen for to FU elements are summarized as follows, with I being the selected input value, O the output, and C the constant value:

f	Description	Function
0	Greater than	$O = 1$ if $I > C$, else 0
1	Less than or equal	$O = 1$ if $I \leq C$, else 0

Further, the architecture parameters $N = 4$ FUs per row and $M = 10$ rows per CDM have been used.

The EA is written to be run on a PowerPC or MicroBlaze core in a Xilinx FPGA. Each FU is encoded in the genome string with 2, 1, and 8 bits for the *feature address, function type,* and *constant*, respectively. This gives a total of 11 bits for each unit. The total amount of bits in the genome for one FUR is then, with $N = 4$, 44 bits. Like with the two previously described approaches, an incremental approach is chosen for the evolution, such that one FUR can be evolved at a time. Each FUR is fed with all the training vectors V_t, and fitness is based on the row's ability to give a positive (1) output for vectors v belonging to its own category ($C_v = C_p$), while giving a negative (0) output for the rest of the vectors ($C_v \neq C_p$). In the case of a positive output when $C_v = C_p$, the value A is added to the fitness sum. When $C_v \neq C_p$ and the row gives a negative output (value 0), 1 is added to the fitness sum. The fitness function F for a row can then be expressed in the following way, where o is the output of the FUR:

$$F = \sum_{v \in V_t} x_v \qquad \text{where } x_v = \begin{cases} A \cdot o & \text{if } C_v = C_p \\ 1 - o & \text{if } C_v \neq C_p \end{cases}$$

For the experiments, a value of $A = 4$ has been used. This architecture will for the rest of the paper be referred to as the "FUR" architecture.

3.5 Characteristics of the EHW Classifier Architectures

All of the approaches except the PLA architecture feature a way of having a graded output for each of the categories, which is then connected to a maximum detector. This can be seen as a way of having several different "detection rules" for each category, which in turn should reduce the effect of overfitting. A parallel could be drawn to stochastic models such as random decision forests [14]: where *single* decision trees (DTs) can be prone to overfitting, having a *collection* of slightly different DTs for one category can significantly reduce this effect. In ECGP and FUR these detection rules are evolved separately, while in the ICE approach the rules are all drawn from the same set of AND-OR gates, which could make them more interdependent.

Another difference from the PLA architecture is that all of the other architectures use incremental evolution, i.e. evolving different subsystems one at a time. The concept of dividing the system into subsystems gives smaller genomes and a simpler search, which in turn should reduce the total evolution time. While the ICE architecture evolves a subsystem for an entire subcategory at a time, the ECGP and the FUR approaches further subdivide this. An advantage of this further subdivision is the ability of starting classification before all of the sub-circuits are evolved, i.e., in the case of the FUR architecture, one could start classifying once one FU row for each category is evolved.

The last two approaches, ECGP and FUR, further employ high-level building blocks in addition to, or instead of, gate-level components. The rationale for this is to reduce the search space for the EA. While the ECGP approach extracts

building blocks automatically, and thus is a very general approach, the FUR architecture uses a priori knowledge in form of pre-defined building blocks found to be good for classification.

4 Experimental Evaluation

To evaluate the different classifier approaches with respect to classification accuracies, we rely on our data set from EMG signal classification. We have chosen this application domain because this is one of the intended applications for all of the compared EHW architectures. The complexity of the problem is close to the problems reported in the original publications and, thus, it is likely that we select suitable parameters in the recreation of the previously proposed architectures. However, we have not tried to optimize the parameters of the PLA and ICE architectures and therefore it could be that they would perform better with other parameters. Each of the classifier architectures is provided with the same feature vectors as input data and has to compute a classification result. We use 3-fold cross-validation, where data from two days (320 4-tuples) is used for training and the third day (160 4-tuples) for testing. This scheme is repeated three times, such that every day once provides test data. In addition, we perform this experiment for ten times and average the classification accuracies.

4.1 Results

Table 1 presents the results of our experiments. It shows the training and test accuracies for the different architectures over the number of fitness evaluations. The number of fitness evaluations measures the effort for training the classifiers. Since the EHW architectures differ in their complexity, also the effort for evaluating one specific instance of a classifier can differ. To account for that, we have defined as one basic fitness evaluation step the evaluation of an entire classifier system for the PLA architecture, the evaluation of only one category sub-circuit for the ICE architecture, one FU row for the FUR architecture, and one single category detector for the ECGP architecture. This makes it possible to compare the computational efforts required to reach a certain classification accuracy, given that now a single evaluation would take roughly the same amount of time for all of the EHW approaches. The results for all of the approaches have been evolved using a 1+4 ES.

4.2 Discussion

Table 1 shows the learning abilities (training accuracy) and the generalization abilities (test accuracy) for the EHW classifiers. From that data, we can derive the following observations:

The PLA architecture requires a high number of fitness evaluations in order to reach its maximum classification accuracy. At 2^{20} evaluations the fitness value and thus the training accuracy is near 100%. Also the maximum test accuracy of

Table 1. Average accuracies (in %) achieved for given numbers of evaluations.

# of evaluations	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
PLA-train	72.0	85.3	92.5	94.7	96.5	97.8	98.5	99.0
PLA-test	52.9	65.3	70.2	70.4	71.7	73.0	72.9	**74.1**
Difference	13.0	20.0	22.3	24.3	23.5	24.8	25.6	24.9
ICE-train	83.3	89.0	91.9	92.6	92.7	93.0	92.9	93.7
ICE-test	65.0	72.7	76.4	77.4	78.0	**79.8**	79.8	78.5
Difference	23.3	16.3	15.5	15.2	14.7	13.2	13.1	15.2
ECGP-train	82.2	82.2	84.3	87.3	89.7	92.6	95.6	97.7
ECGP-test	44.4	44.5	52.4	62.7	75.4	84.1	88.6	**90.3**
Difference	37.8	37.8	31.9	24.6	14.4	8.4	7.1	7.5
FUR-train	90.2	94.3	95.8	96.4	96.4	96.7	96.7	96.6
FUR-test	84.9	88.7	89.2	**90.0**	89.7	89.0	88.4	88.1
Difference	5.3	5.6	6.6	6.4	6.7	7.7	8.3	8.5

74.1% is reached at 2^{20} evaluations. The slow convergence compared to ICE and FUR could be the result of the genome having a high number of bits and thus the search uses more time to converge than the incremental approaches which divide the genome into smaller parts. Despite the high ability to fit the training set, there is a significant difference between the training accuracy and the test accuracy, indicating a low generalization ability.

The ICE architecture reaches lower training set accuracies, but still manages to achieve a higher test accuracy than the PLA approach, indicating a better generalization ability. The slightly worse ability to fit the training set could stem from the fact that the number of AND-OR gates is rather limited for one category which might make it difficult to accommodate the whole training set.

The ECGP architecture is a very general approach and therefore features a large chromosome compared to the other, more domain-specific, architectures. The large chromosome size results in a slow initial convergence rate. However, the ECGP architecture delivered the best test accuracy among all architectures. Moreover, the tendencies in the training and test accuracies let us assume that the ECGP architecture has the best generalization behavior among all architectures for this particular benchmark.

The FUR architecture gives, like the ECGP approach, good test set accuracies, and also small gaps between the accuracies for the training and test sets, indicating good generalization abilities. Further, the number of evaluations required before a high accuracy is achieved is significantly lower than for the other approaches. The use of predefined high-level building blocks selecting logical features of the inputs significantly reduces the genome size and thus the search space, in addition to making the search simpler. Combined with fine-grained incremental evolution the genome becomes very short and high evolution speed is achieved. On the other hand, the FUR architecture seems to overfit slightly when the evolution is run for too long.

The PLA, ICE and to some extent the ECGP architecture as well are rather fine-granular, while the FUR architecture definitely uses more complex func-

tional blocks. In the first group the test accuracies increase from PLA over ICE to ECGP. This can be explained by the rising complexity of the according hardware representation models: The PLA architecture is limited by its two-level logic structure. The ICE architecture can be viewed as multi-PLA that applies several sum-of-products for each category detection unit. The ECGP architecture's basically has the same structure as the ICE architecture, a set of classifiers followed by summation blocks and a maximum detector. However, compared to ICE, ECGP is neither restricted to single or multiple two-level logic expressions nor to a limited set of logic block functions. The drawback of this general architecture is the increased computational effort.

5 Conclusions

In this paper, we have compared four EHW classifier approaches using a common data set. Two of the architectures, ECGP and FUR are our own recent developments while the other two, PLA and ICE, have been proposed earlier. The experimental results show that both our approaches outperform the previous architectures. By introducing incremental evolution combined with high level functions, one obtains higher classification rates and significantly faster training times than earlier EHW approaches. The ECGP architecture relies on a very general hardware representation model and requires a high number of fitness evaluations to fit the training data set, but excels at classifying the test data. The FUR approach, on the other hand, relies on a domain-specific hardware representation model which on average allows to evolve classifiers with high classification accuracies much faster.

The ECGP and FUR approaches have earlier shown to work on a higher number of inputs in [5], although in that case the FUR approach obtained better overall accuracies. The FUR approach has also been shown to handle a larger number of categories in [10]. In contrast, the experiment presented in this paper used only of few input features and a rather moderate number of categories. The investigation of how the different architectures scale with other applications having a higher number of inputs and categories is subject of further work.

Acknowledgment

This work was supported by the German Research Foundation under project numbers PL 471/1-2 and SI 674/3-2 within the priority program *Organic Computing* and the Research Council of Norway through the project *Biological-Inspired Design of Systems for Complex Real-World Applications* under project number 160308/V30.

References

1. Higuchi, T., Iwata, M., Kajitani, I., Iba, H., Hirao, Y., Manderick, B., Furuya, T.: Evolvable Hardware and its Applications to Pattern Recognition and Fault-

Tolerant Systems. In: Towards Evolvable Hardware: The evolutionary Engineering Approach. Volume 1062 of LNCS. Springer (1996) 118–135
2. Kajitani, I., Hoshino, T., Nishikawa, D., Yokoi, H., Nakaya, S., Yamauchi, T., Inuo, T., Kajihara, N., Iwata, M., Keymeulen, D., Higuchi, T.: A Gate-Level EHW Chip: Implementing GA Operations and Reconfigurable Hardware on a Single LSI. In: Proceedings 2nd International Conference on Evolvable Systems (ICES). Volume 1478 of LNCS., Springer (1998) 1–12
3. Yasunaga, M., Nakamura, T., Yoshihara, I.: Evolvable Sonar Spectrum Discrimination Chip Designed by Genetic Algorithm. In: Systems, Man and Cybernetics. Volume 5., IEEE (1999) 585–590
4. Yasunaga, M., Nakamura, T., Yoshihara, I., Kim, J.: Genetic Algorithm-based Design Methodology for Pattern Recognition Hardware. In: Proceedings 3rd International Conference on Evolvable Systems (ICES). Volume 1801 of LNCS., Springer (2000) 264–273
5. Glette, K., Kaufmann, P., Gruber, T., Torresen, J., Sick, B., Platzner, M.: Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control. In: Submitted to 3rd NASA/ESA Conference on Adaptive Hardware and Systems (AHS). (2008)
6. Kajitani, I., Hoshino, T., Iwata, M., Higuchi, T.: Variable Length Chromosome GA for Evolvable Hardware. In: International Conference on Evolutionary Computation (ICEC), IEEE (1996) 443–447
7. Murakawa, M., Yoshizawa, S., Kajitani, I., Furuya, T., Iwata, M., Higuchi, T.: Hardware Evolution at Function Level. In: Proceedings 4th Parallel Problem Solving from Nature (PPSN). Volume 1141 of LNCS., Springer (1996) 62–71
8. Kajitani, I., Sekita, I., Otsu, N., Higuchi, T.: Improvements to the Action Decision Rate for a Multi-Function Prosthetic Hand. In: Proceedings 1st International Symposium on Measurement, Analysis and Modeling of Human Functions (ISHF). (2001) 84–89
9. Torresen, J.: Two-Step Incremental Evolution of a Digital Logic Gate Based Prosthetic Hand Controller. In: Proceedings 4th International Conference on Evolvable Systems (ICES). Volume 2210 of LNCS. Springer (2001) 1–13
10. Glette, K., Torresen, J., Yasunaga, M.: An Online EHW Pattern Recognition System Applied to Face Image Recognition. In: Proceedings Applications of Evolutionary Computing (EvoWorkshops2007). Volume 4448 of LNCS. Springer (2007) 271–280
11. Glette, K., Torresen, J., Yasunaga, M.: Online Evolution for a High-Speed Image Recognition System Implemented On a Virtex-II Pro FPGA. In: Proceedings 2nd NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Los Alamitos, CA, USA, IEEE CS Press (2007) 463–470
12. Miller, J., Thomson, P.: Cartesian Genetic Programming. In: Proceedings 3rd European Conference on Genetic Programming (EuroGP), Springer (2000) 121–132
13. Walker, J.A., Miller, J.F.: Evolution and Acquisition of Modules in Cartesian Genetic Programming. In: Proceedings 7th European Conference on Genetic Programming (EuroGP). Volume 3003 of LNCS., Springer (2004) 187–197
14. Ho, T.K.: Random Decision Forests. In: Proceedings 3rd International Conference on Document Analysis and Recognition (ICDAR). Volume 1., IEEE (1995) 278