Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control

Kyrre Glette, Jim Torresen
University of Oslo, Norway
{kyrrehg,jimtoer}@ifi.uio.no

Thiemo Gruber, Bernhard Sick
University of Passau, Germany
{grubert,sick}@fim.uni-passau.de

Paul Kaufmann, Marco Platzner
University of Paderborn, Germany
{paul.kaufmann,platzner}@upb.de

***Abstract*— Evolvable hardware has shown to be a promising approach for prosthetic hand controllers as it features self-adaptation, fast training, and a compact system-on-chip implementation. Besides these intriguing features, the classification performance is paramount to success for any classifier. However, evolvable hardware classifiers have not yet been sufficiently compared to state-of-the-art conventional classifiers. In this paper, we compare two evolvable hardware approaches for signal classification to three conventional classification techniques: k-nearest-neighbor, decision trees, and support vector machines. We provide all classifiers with features extracted from electromyographic signals taken from forearm muscle contractions, and try to recognize eight different hand movements. Experimental results demonstrate that evolvable hardware approaches are indeed able to compete with state-of-the-art classifiers. Specifically, one of our evolvable hardware approaches delivers a generalization performance similar to that of support vector machines.**

I. Introduction

Prosthetic hand controllers (PHCs) are usually operated by the signals generated by contracting muscles – namely electromyographic (EMG) signals. Such signals consist of a transient phase and a steady state phase. While some investigations have been done on PHC using transient bursts [1], the classical approach for the classification of muscle contractions extracts feature vectors from the signals' steady state.

There are several goals in PHC design. For example, while classic PHCs drive actuators with constant force, a new and rather ambitious goal is to create a proportional prosthetic hand controller [2]. Then, traditional PHCs only cover two motions – *open* and *close*. Here it is highly desirable for users to have improved control over motions using prostheses with more degrees of freedom. To that end, PHCs classifying multiple movements are necessary. Finally, having access to PHCs which adapt themselves to the changes in the user's EMG signal patterns would be a great advantage. The EMG patterns are influenced by parameters such as muscle fatigue, skin conductivity, and age. Currently, users are required to adapt to pre-defined EMG patterns, partly supported by periodic re-training sessions.

In this context evolvable hardware (EHW) becomes an interesting approach, providing possibilities for self-adaptation, fast training, and compactness. The combination of genetic algorithms (GAs) and reconfigurable hardware makes it possible to have hardware systems which can be automatically constructed and then adapt their structure to specification changes.

Kajitani et al. presented such an EHW-based adaptive PHC [3]–[6]. A structure of AND gates followed by OR gates is evolved using a GA which is implemented on the same chip, resulting in a very compact PHC system. The controller is trained with feature vectors extracted from EMG data where one input signal consists of four channels at a resolution of four bits. The classifier distinguishes between six different movements. The classification performance was computed by dividing the EMG data into two halves and using one half as training data and the second half as test data. Although the results show a competitive classification rate for evolved circuits compared to artificial neural networks (ANNs), it was noted that the size of the used data set might be insufficient which is underlined by the strongly varying classification rates. As a result of having the GA implemented entirely in hardware and on the same chip, the learning time (800ms) for the EHW approach is significantly shorter than for the ANN. Short training times are important for the user-friendliness of a PHC, especially if online adaptation is applied.

Using similar EMG data, Torresen conducted experiments on incremental evolution of an EHW architecture in [7]–[9]. The results showed that a two-step incremental approach can lead to a better generalization performance than both traditional one-step evolution and ANN. In addition, combining the best subsystems from different runs further increased the average generalization performance. The total evolution time was also shortened by applying the incremental approach.

A large-input EHW pattern classification system, Logic Design using Evolved Truth Tables (LoDETT), was proposed by Yasunaga et al. [10]. Although providing high classification accuracy and speed, the system lacks the ability of online evolution.

A major challenge is to actually map evolved circuits to reconfigurable hardware at runtime, as for todays field-programmable gate arrays (FPGAs) there exist no commercial tools that support the online reconfiguration at the fine-granular level of logic gates and wires. An alternative EHW approach to online reconfigurability on FPGAs is the Virtual Reconfigurable Circuit (VRC) method proposed by Sekanina in [11]. This method obtains virtual reconfigurability by chang-

ing register values and multiplexor control signals in the user circuit. The advantages of this technique are faster reconfiguration and applicability to all SRAM-based FPGAs. However, the method potentially requires much logic resources.

Using the virtual reconfiguration technique, Glette et al. proposed an online evolvable EHW architecture for classification tasks [12]–[14]. The architecture was applied to multiple-category face image recognition and sonar return classification. The evolution part of the system has been implemented on an FPGA, where fitness evaluation is carried out in hardware and the evolutionary algorithm runs on an on-chip processor.

Non-EHW adaptable controllers for PHC-related tasks have also been proposed. An example is [15] where a 6-class controller is developed. More recently, support vector machines (SVMs) have been applied to PHCs. Successful experiments include an 8-class controller controlling a robotic arm [16] and a controller allowing for finger movements [17].

Given the possibilities for self-adaptation, fast training, and compact classifier circuits, we continue the line of research on EHW-based PHCs. In this paper we investigate the classification performance of EHW approaches for a multi-movement problem and see if they are competitive to conventional classifier approaches. For this we have collected a new EMG data set with 8 different motion classes spanning three days which can be used as a common reference. We present two different EHW approaches and compare their performance to three conventional approaches, one of which is SVM, as this is regarded as one of the most powerful classifier methods existing today.

The paper is structured as follows. Section II describes the setup of the EMG sensor system and the signal processing applied to obtain feature vectors. The tested conventional classifiers as well as the two EHW approaches are detailed in Section III. The experimental results are given and discussed in Section IV. Finally, Section V concludes the paper.

II. EMG Sensor System and Feature Extraction

For EMG data acquisition, we use a measurement system comprising four components: EMG sensors (Tyco Arbo*, Ag/AgCl, 35 mm), amplifiers (Biovison [18]), A/D converters (N.I. [19]), and a standard computer. Our system continuously monitors four sensor channels with 14 bit resolution at a sampling rate of 6 kHz. Two important requirements for such a measurement system are the reduction of noise in the analog signal domain and a reproducible biomechanical experiment setup.

To reduce noise, we employ an optical bridge (Sonowin [20]) to galvanically decouple the signal amplifiers and the A/D converters from the computer that accumulates the data. A separate battery provides a stable power supply to the amplifiers and A/D converters. Moreover, the amplifiers have been placed as near as 10 cm to the skin-attached electrodes in order to minimize parasitic inductance of a significant level.

We have placed the four electrode pairs on the top, bottom, medial, and lateral sides of the forearm as shown in Figure 1 with the reference at the wrist. The exact electrode positions

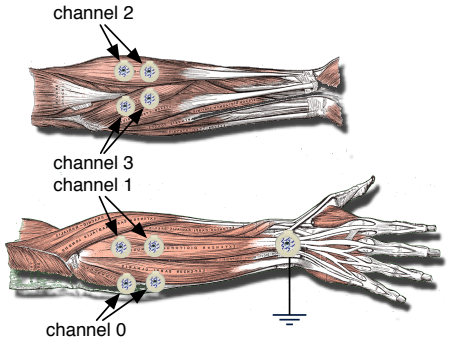

Fig. 1. Sensor placement (muscle anatomy taken from [21]).

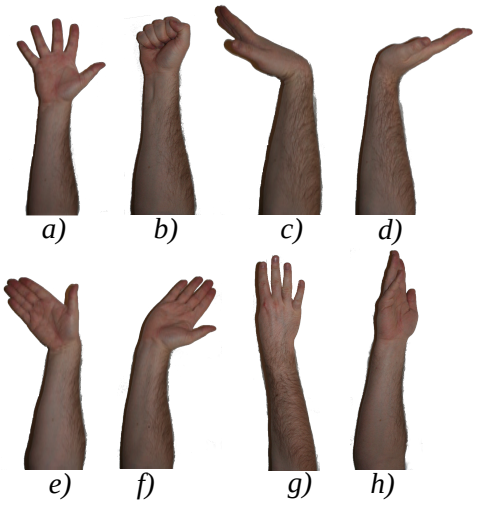

Fig. 2. Motion classes: *a)* open, *b)* close, *c)* flexion, *d)* extension, *e)* ulnar deviation, *f)* radial deviation, *g)* pronation and *h)* supination.

have been determined specifically for each test subject to obtain pronounced signals. After this initial calibration, the electrode positions have been marked to be able to re-establish the experimental setup on different days.

In a single experiment run, the test subject had to perform 20 iterations of a sequence of eight different movements. These movements are *open, close, flexion, extension, ulnar deviation, radial deviation, pronation,* and *supination*, and are depicted in Figure 2. A single movement consists of two phases: a 9 seconds relaxation part and an 11 seconds contraction part. The EMG signal for the contraction part divides into a three seconds phase at the onset of the contraction containing the transient components of the EMG signal, and an eight seconds steady state phase which corresponds to a constant force contraction. A part of this steady state phase has been used for classification. An example for a complete EMG signal is presented in Figure 3.

Signal preprocessing and feature extraction is done completely in the digital domain. Following the approach presented by Kajitani et al. in [5], we extract the features in four steps:

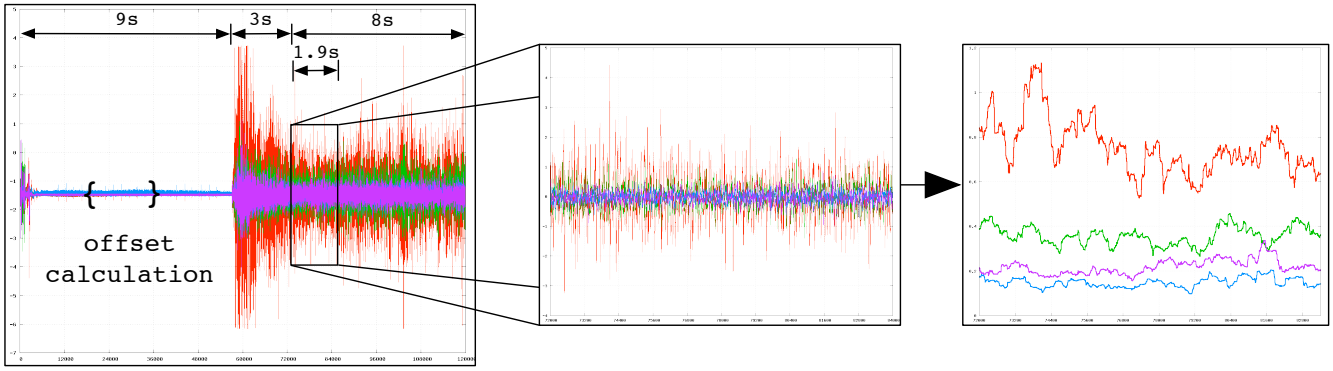

Fig. 3. EMG signal preprocessing. The left figure shows the raw signals for all four channels, consisting of a nine seconds relaxation phase, a three seconds transient phase with intensified activity, and an eight seconds steady state contraction phase. The center figure presents the DC offset-compensated first 1.9 seconds of the steady state phase, and the right figure the RMS smoothed signals from which the features are extracted.

1) For every channel $k, k = 1, \ldots, 4$, and movement $p, p = 1, \ldots, 8$, the sensor DC offset o_{kp} is calculated as the mean value of all signal samples between the third and the fifth second of the signal relaxation phase.
2) The steady state signal d_{ikp} is DC offset-compensated and smoothed by a root mean square (RMS) method with a window size of $w_s = 600$. The first 1.9 seconds ($11'400$ samples at 6 kHz) of the rectified and smoothed signal d'_{jkp} are calculated by

$$d'_{jkp} = \left[\frac{1}{w_s} \sum_{i=j}^{j+w_s-1} (d_{ikp} - o_{kp})^2 \right]^{\frac{1}{2}},$$

with $j = 1 \ldots 11'400$.
3) For this signal, a logarithm-transformed moving average is computed with a window size of $w_f = 6'000$ samples and a shift amount of $s_f = 600$ samples. The non-normalized feature thus consists of 10 values and is defined as

$$f_{l_m kp} = -\log \left(\frac{1}{w_f} \sum_{j=l_m}^{l_m+w_f-1} d'_{jkp} \right),$$

with $l_m = 1 + (m-1) \cdot s_f$, and $m = 1, \ldots, 10$.
4) Finally, the features are normalized for each channel separately:

$$g_{lkp} = \frac{f_{lkp} - \min_{l,p}(f_{lkp})}{\max_{l,p}(f_{lkp}) - \min_{l,p}(f_{lkp})}$$

Taking all $k = 4$ channels into account, the feature vector for a single movement consists of 10×4 values. These 40 values are fed into the classifiers.

III. Classifiers, Models, and Architectures

In this section the different classification approaches to the multi-movement PHC problem are described. First, approaches based on conventional classification techniques are presented, followed by two EHW approaches.

A. Conventional Classification Techniques

In order to compare the results of the evolvable hardware classifiers to some conventional classifier paradigms, we have selected three paradigms which realize different forms of decision boundaries between classes.

k-nearest-neighbor (kNN) is a very simple data-based approach for classification [22]. We regard kNN as a kind of baseline method. Here, the number of neighbors is set to $k = 5$.

Decision trees (DTs) can be used for the classification of numerical as well as categorical data. A DT realizes a set of well-interpretable *if-then* rules. In a tree structure, each leaf node represents a classification decision, each non-leaf node evaluates the corresponding attribute. An input sample is classified by successive tests from the root of a DT down to a leaf. Here, we use the C4.5 algorithm [23] to build the DT. C4.5 selects the next attribute (based on a greedy principle) according to an information gain measure. Pruning techniques such as subtree raising are applied to reduce an overfitting of the classifier to the training data set.

SVMs basically use a hyperplane to separate two classes [24]. For problems that can not be linearly separated in the input space, SVMs find a solution using a nonlinear transformation of the original input space into a high-dimensional so-called feature space, where an optimal separating hyperplane is determined. Those hyperplanes are called optimal that have a maximal margin, where margin means the minimal distance from the separating hyperplane to the closest (mapped) data points (so-called support vectors). The transformation is usually realized by nonlinear kernel functions, e.g., Gaussian kernels. ν-SVMs, which are used here, introduce slack variables – being subject to minimization as well – to allow a certain degree of misclassification. The key advantage of SVMs is the principle of structural risk minimization which typically yields very good generalization performance compared to other classifier paradigms.

The experiments with kNN, DT, and SVM have been conducted with the data mining framework RapidMiner [25].

B. The ECGP EHW Model

The first EHW-based classifier relies on a variant of the *Embedded Cartesian Genetic Programming* (ECGP) model. ECGP is an extension of the popular FPGA-oriented cartesian genetic programming (CGP) model [26]. CGP is a structural hardware model that arranges logic cells in a two-dimensional geometric layout. In Figure 4, a parameterized CGP model is shown. The model consists of $n_c \times n_r$ combinational logic blocks, n_i primary inputs, and n_o primary outputs. A logic block has n_n inputs and implements one out of n_f different logic functions of these inputs. While the primary inputs and outputs can connect to any logic block input and output, respectively, the connectivity of the logic block inputs is restricted. The input of a logic block at column c may only connect to the outputs of blocks in columns $c-l,\ldots,c-1$ as well as to the primary inputs. The levels-back parameter l restricts wiring to hardware-friendly local connections. More importantly, as only feed-forward connections are allowed the creation of combinational feedback loops is avoided. The logic block genes are mapped to the array in order of their position within the chromosome. Consequently, a CGP configuration implicitly encodes the placement of logic blocks but no routing.

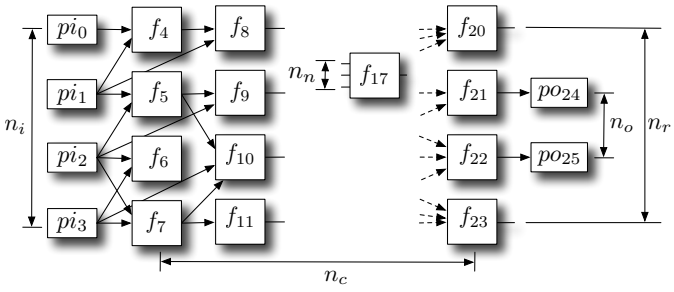

Fig. 4. Cartesian Genetic Programming Model.

ECGP extends CGP by configuring it as a single line of functional nodes ($n_r = 1, l = n_c$) and by adding the automatic definition and reuse of sub-functions (modules) [27]. Modules are defined as compositions of primitive nodes, see Figure 5. The size of a module is restricted, which also restricts the maximal chromosome size. The functional set contains all boolean functions having $n_n = 4$ inputs. We have improved the ECGP model with an age-based module creation technique. Age-based module creation aggregates primitive nodes that have persisted in the chromosome for a higher number of generations. The rationale is that aged nodes are likely to contribute directly or indirectly to an individual's success and should therefore be preferred over randomly selected nodes for module creation.

Our EHW classifier system evolves six classifier circuits for each movement (class) C_p, with $p = 1,\ldots,P$, and $P = 8$. Each of the 40 elements of a feature vector is linearly quantized to a 4-bit representation and input to a 1-out-of-16 encoder. The resulting 40×16 bits are then fed to a classifier

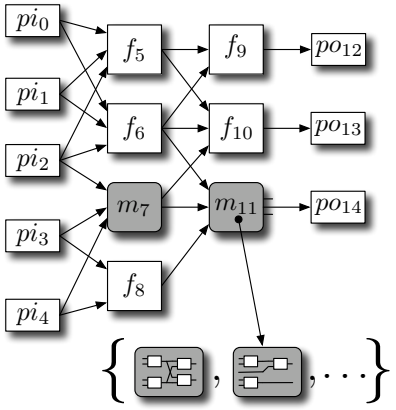

Fig. 5. The Embedded Cartesian Genetic Programming Model automatically creates modules as compositions of primitive nodes.

TABLE I
PARAMETERS FOR THE EVOLUTION OF THE ECGP EHW CLASSIFIER

n_i / n_o / n_r / n_c	640 / 1 / 1 / 50–250
n_n / n_f	4 / $\mathbb{B}^4$
#fitness evaluations per generation	4
mutation prob.	1.0
mutation rate	0.03
one point mutation prob.	0.6
compress / expand prob.	0.1 / 0.2
module point mutation prob.	0.04
add / remove module input prob.	0.01 / 0.02
add / remove module output prob.	0.01 / 0.02
maximum module size	10

circuit. A single classifier differentiates between class C_p and the remaining classes. For each feature vector and class, we calculate the maximum of activated classifier circuits and take the class with the most activations as a result. In case of a tie, no classification decision is taken.

During the training phase, we have to determine a fitness value for each classifier circuit. The set of training feature vectors X splits into P subsets X_i for which we know the correct classification result. The fitness for a classifier circuit c_p that recognizes class C_p is determined as the square error distance to the predictions of a perfect classifier c_p^*:

$$f(c_p) = \left[1 + \frac{1}{|X|}\sum_{i=1}^{8}\left[\sum_{x\in X_i}|c_p^*(x) - c_p(x)|\right]^2\right]^{-1}$$

Similar to [27] we have chosen a standard $1+4$ evolutionary strategy as the optimization algorithm. The parameters of the ECGP model and the evolutionary strategy are shown in Table I. The population is initialized randomly with a length of 50 logic blocks. Depending on the created modules, the chromosome is allowed to grow up to 250 blocks.

C. The Functional Unit Row EHW Architecture

The second EHW-based classifier investigated in this paper is specifically tailored to classification tasks and online evolution. To facilitate online evolution, the classifier architecture

is implemented as a circuit whose behavior and connections can be controlled through configuration registers, similar to the VRC approach [11]. By writing the genome bitstream produced by the GA to these registers, one obtains the phenotype circuit which can then be evaluated. The architecture is presented on a hardware-abstracted level in the following sections. Details about the implementation can be found in [13].

a) Classification Module: The classifier system consists of P category detection modules (CDMs), one for each category C_p to be classified – see Figure 6. 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 detected by a maximum detector, and the identifying number of this category will be output from the system. In the case of a tie, the C_p with the lowest index p is chosen.

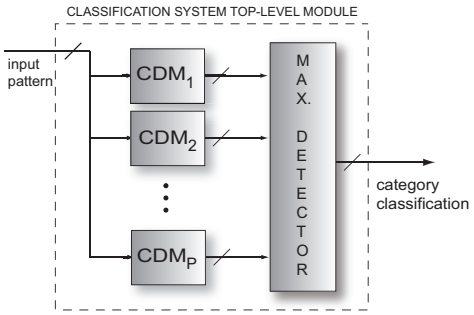

Fig. 6. EHW classification module view. The pattern to be classified is input to all of the category detection modules.

b) Category Detection Module: Each CDM consists of M "rules" or functional unit (FU) rows – see Figure 7. Each FU row 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. This means that all outputs from the FUs must be 1 in order for a rule to be

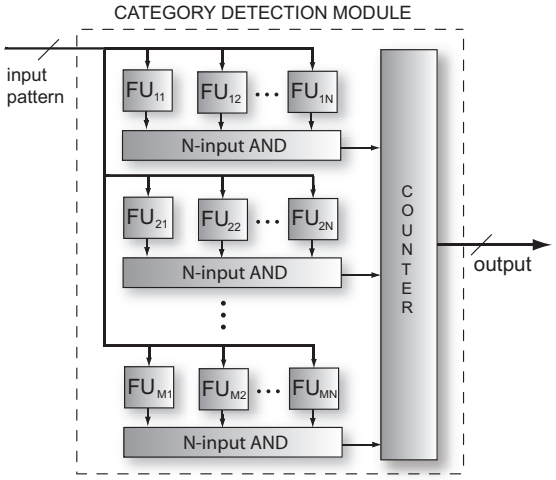

Fig. 7. Category detection module. N functional units are connected to an N-input AND gate.

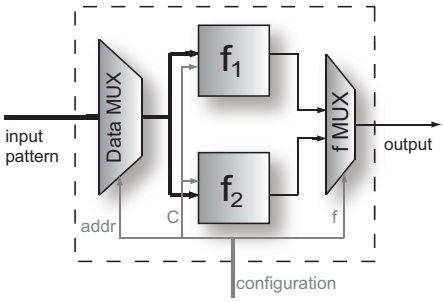

Fig. 8. Functional unit. The configuration lines are shown in gray. The data MUX selects which of the input data to feed to the functions f_1 and f_2. The constant C is given by the configuration lines. Finally, the f MUX selects which of the function results to output.

activated. The 1-bit outputs from the AND gates are connected to an input counter which counts the number of activated FU rows. As the number of FU rows is increased, so is the output resolution from each CDM. Each FU row is evolved from an initial random bitstream, which ensures a variation in the evolved FU rows.

c) Functional Unit: The FUs are the reconfigurable elements of the architecture. As seen in Figure 8, 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 chosen. One data element is thus *selected* from the input bits, depending on the configuration lines. This data is then fed to the available functions. Any number and type of functions could be imagined, but for clarity, in Figure 8 only two functions are illustrated. The choice of functions for the EMG classification application will be detailed below. In addition, the unit is configured with a constant value, C. This value and the input data element are used by the function to compute the output from the unit. The advantage of selecting which inputs to use is that connection to all inputs is not required. A *direct* implementation as done in the LoDETT system [10] would have required $N = 40$ FUs in a row for the PHC application. In contrast, our system uses $N = 4$ units. The rationale is that not all of the inputs are necessary for the classification.

The 40 normalized amplitudes (10×4 channels) of the input EMG signal are converted to 8-bit values before they are input to the FUs. Based on the data elements of the input being 8-bit values, the functions available to the FU elements have been chosen to *greater than* and *less than or equal*. Through experiments these functions have shown to work well, and intuitively this allows for discriminating signals by looking at the different channels' amplitudes. The constant is also 8 bits, and the input is then compared to this value to give *true* or *false* as output. This can be 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

The GA is written to be run on the PowerPC 405 core in the Xilinx Virtex-II Pro (or better) FPGAs [28], or the MicroBlaze soft processor core available for a greater number of FPGA devices. A bit string (genome) is associated with each individual in the population. Each FU is encoded in the genome string with 6, 1, and 8 bits for the *feature address, function type,* and *constant*, respectively. This gives a total of $B_{unit} = 15$ bits for each unit. The total amount of bits in the genome for one FU row is then, with $N = 4$: $B_{tot} = B_{unit} \cdot N = 60$.

Evolving the whole classification system in one run would give a very long genome, therefore an incremental approach is chosen. Each category detector CDM_p is evolved separately, since there is no interdependency between the different categories. This is also true for the FU rows each CDM consists of. Thus, the evolution can be performed on one FU row at a time. This significantly reduces the genome size. One then has the possibility of evolving CDM_p in M steps before proceeding to CDM_{p+1}. However, we evolve only *one* FU row in CDM_p before proceeding to CDM_{p+1}. This makes it possible to have a working system in P evolution runs (that is, $1/M$ of the total evolution time). While the recognition accuracy is lower with only one FU row for each CDM, the system is operational and improves gradually as more FU rows are added for each CDM.

A certain set of the available vectors, V_t, are used for training of the system, while the remaining, V_v, are used for verification after the evolution run. Each row of FUs is fed with the training vectors ($v \in V_t$), 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 other cases do not contribute to the fitness value. The fitness function F for a row can then be expressed in the following way, where o is the output of the FU row:

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

For the experiments, a value of $A = 4$ has been used. This emphasis on the positive matches for C_p has shown to speed up the evolution. Further, the architecture parameters $N = 4$ FUs per row and $M = 24$ rows per CDM have been used. A maximum of 100 generations have been allowed for each evolution run, however, evolution has been terminated earlier in case maximum fitness was reached. The GA follows the Simple GA approach [29], with elitism, a population size of 32 and a crossover rate of 0.9. Mutation follows a customized scheme described in [12].

IV. Experiments and Results

This section first presents the experiments we have conducted and the metrics we have used to evaluate the different classification approaches, and then shows and discusses the obtained results.

A. Classification Experiments

In order to evaluate the classification performance of the different classifiers, we have collected and preprocessed EMG data from a test subject on three consecutive days. As described in Section II, a single feature vector for one movement comprises 40 values. On each day, we have collected feature vectors from 20 iterations of 8 different movements. Collecting data over a period of three days gives us a more varied data set than collecting all the data in a single experiment run and, therefore, provides a more realistic basis for testing the classifiers' generalization abilities.

We use the same k-fold cross-validation technique to determine the classification rates of all classifiers. This technique segments an overall data set into k subsets of approximately equal size. In k runs, one subset is used for testing whereas the others are used for the training of the classifiers. For each separate day (*Day1, Day2, Day3*), we have defined a classification experiment using leave-one-out cross-validation. That is, k is set to equal the number of feature vectors. The next classification experiment combines all three days (*Day1–3*) and determines the classification performance using the same leave-one-out scheme. The final experiment (*2of3*) uses 3-fold cross-validation, where two days are used for training and the third day for testing. This is repeated three times, such that every day once provides test data. The *2of3* experiment has been set up with the intention of measuring the classifiers' session independence, i.e., covering the case of using the prosthesis every day without retraining.

B. Classification Performance Results

As a metric to compare the classification performance of the different approaches we use the classification accuracy expressed by the error rate. Table II presents the test error rates which show the classifiers' generalization abilities, and Table III presents the error rates obtained for the training data sets pointing to the classifiers' approximation abilities. kNN, DT, and SVM are the k-nearest-neighbor, decision tree, and support vector machine approaches, respectively. Table IV presents the error rates for the individual movements for the evolvable hardware approaches and the SVM. EHW1 and EHW2 are the evolvable hardware approaches, where EHW1 refers to the ECGP-based model (see Section III-B) while EHW2 denotes the FU row architecture (see Section III-C).

Since the EHW classifiers are evolved from random genomes, each evolved classifier has a different structure and the classification rates vary slightly. For the experiments *Day1, Day2, Day3* and *Day1–3*, the leave-one-out technique requires us to evolve a rather high number of classifiers which averages out the differences in initial genomes. The *2of3* experiment, however, generates only three classifiers. To achieve sound error rates, we have evolved 10×3 classifiers and averaged the results. Further, the training error for EHW1 is 5% in all

TABLE II
TEST ERRORS (GENERALIZATION)

	Day1	*Day2*	*Day3*	*Day1–3*	*2of3*
kNN	3.5 %	4.6 %	4.6 %	4.5 %	5.6 %
DT	9.7 %	11.3 %	10.5 %	9.0 %	15.9 %
SVM	4.2 %	4.0 %	2.6 %	4.5 %	5.4 %
EHW1	9.8 %	4.0 %	5.3 %	9.0 %	10.6 %
EHW2	9.0 %	4.6 %	4.0 %	4.9 %	8.4 %

TABLE III
TRAINING ERRORS (APPROXIMATION)

	Day1	*Day2*	*Day3*	*Day1–3*	*2of3*
kNN	2.8 %	2.7 %	4.0 %	3.6 %	3.4 %
DT	1.4 %	2.0 %	1.3 %	1.8 %	2.0 %
SVM	1.4 %	0.7 %	2.0 %	3.6 %	2.6 %
EHW1	5.0 %	5.0 %	5.0 %	5.0 %	5.0 %
EHW2	1.9 %	0.8 %	0.0 %	3.6 %	2.6 %

experiments, as in the evolution of EHW1 classifiers this rate has been used as termination criterion.

C. Discussion

From the experimental results, we can make the following observations:

- Among the conventional classifiers, kNN yields surprisingly good results. While this approach is likely to be inapplicable for a real prosthetic hand controller as all the data have to be stored and evaluated for the classification decision, it shows that the classification problems posed by our experiments basically can be solved with very simple classifiers. DT – despite the pruning techniques which have been applied – are prone to overfitting in this application, in particular in experiment *2of3*. Amongst the conventional classifiers, SVM yield – as expected – the best results.
- The EHW approaches – and this is the main result of our experiments – also yield a very good classification performance. The test errors of EHW1 and EHW2 are between those of kNN and DT. EHW2 performs better than EHW1 with a generalization performance which is very similar to that of SVM. When observing the individual movement results (see Table IV), one can see that the lowest error rates vary between the two EHW architectures and the SVM, and there is no approach which is consistently the best.
- From the viewpoint of the different experiments, we can state that the problem in experiment *2of3* is the most difficult. Compared to other experiments, the test errors are higher due to the fact that the classifiers have to cope with data obtained in different sessions. However, the *2of3* experiment is possibly the most important one as it is closest to a real application scenario.

V. CONCLUSION AND FUTURE WORK

In this paper, we have compared two EHW approaches for a multi-motion PHC to state-of-the-art conventional classification techniques. One of the EHW approaches is rather general, whereas the second is tailored for online evolution and classification tasks. We have detailed our method for acquiring EMG data and extracting feature vectors. Based on the obtained data, we have defined several experiments and computed the classification accuracy for the different classifiers. The main result of this paper is that EHW classifiers are on par with conventional techniques. This insight is of utmost importance as the appeal of EHW approaches roots in their suitability for self-adaptation, fast training, and compact system-on-chip implementation. Knowing that EHW approaches also compete in terms of classification performance motivates future work along the following lines:

Our insight gained through experimenting clearly shows that the generalization performance of all classifiers could be improved significantly if more training data from additional sessions were available. Hence, we plan to expand the number of sessions and test subjects in future work.

Interestingly, all of the classification approaches are able to provide a differentiated output of the certainty of the match to a given class (movement). This makes it possible to define thresholds above which no classification result is issued. For example, if there are two movements with high output values and thus high certainty, the prosthetic hand controller will not drive the actuators. Such uncertain classifications can indeed be dropped in a PHC that requires sustained muscle tension in order to perform a motion. A single classification result generates only a limited motion of the prosthetic hand and, overall, the user will perceive fewer errors [16]. We consider this technique essential for developing more responsive PHCs and will investigate it in our future work.

While the two EHW approaches presented both give good classification results, their strategies are different. The ECGP-based model of EHW1 is a very general model which allows for complex structures by applying automatic generation of building blocks. Since the model is general it should be possible to apply it to other tasks with minimal effort. The FU row-based EHW2 architecture, on the other hand, is using more *a priori* knowledge in form of defined building blocks and data buses tailored for classification. It is designed for direct hardware implementation and this has also made online reconfigurability possible. It will be of interest for future experiments to investigate the mapping of the ECGP model to hardware, and the possibility of increasing the flexibility of the FU row architecture.

It is difficult to compare the classification performance of our EHW approaches directly to the earlier EHW approaches due to the different data sets being used. However, in future work it would be desirable to reproduce the earlier proposed architectures in order to perform a comparison using the same data set.

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*, the Research Council of Norway through the project *Biological-Inspired*

TABLE IV
INDIVIDUAL MOVEMENT ERRORS (GENERALIZATION)

		Close	Extension	Flexion	Open	Pronation	Radial dev.	Supination	Ulnar dev.
Day1	SVM	0.0%	0.0%	6.3%	5.3%	0.0%	6.3%	11.1%	5.0%
	EHW1	17.7%	0.0%	6.3%	15.8%	5.3%	6.3%	11.1%	15.8%
	EHW2	0.0%	15.8%	12.5%	5.9%	10.0%	6.3%	16.7%	5.3%
Day2	SVM	5.9%	0.0%	0.0%	15.0%	0.0%	5.3%	5.3%	0.0%
	EHW1	11.8%	0.0%	0.0%	0.0%	5.0%	5.3%	5.3%	5.6%
	EHW2	0.0%	5.0%	0.0%	5.9%	10.5%	5.3%	5.3%	0.0%
Day3	SVM	0.0%	0.0%	0.0%	0.0%	10.0%	5.9%	0.0%	5.0%
	EHW1	0.0%	5.0%	0.0%	10.5%	10.0%	5.9%	0.0%	10.5%
	EHW2	0.0%	5.3%	0.0%	0.0%	10.0%	11.8%	0.0%	5.0%
Day1–3	SVM	3.9%	0.0%	1.9%	5.2%	8.5%	5.8%	5.4%	5.1%
	EHW1	11.8%	3.4%	9.4%	8.6%	6.8%	9.6%	14.3%	8.6%
	EHW2	0.0%	5.2%	3.8%	2.0%	1.7%	13.5%	5.4%	8.5%
2of3	SVM	3.9%	0.0%	0.0%	8.6%	8.5%	9.6%	5.4%	6.8%
	EHW1	5.9%	12.2%	0.0%	3.5%	12.2%	12.8%	22.2%	15.8%
	EHW2	1.9%	12.1%	0.2%	7.8%	4.8%	19.2%	14.3%	7.5%

Design of Systems for Complex Real-World Applications under project number 160308/V30, and the German Academic Exchange Service / Research Council of Norway through the collaborative project *Autonomously Adaptable System-on-Chip* under project number D/06/12740.

REFERENCES

[1] B. Hudgins, P. Parker, and R. Scott, "A new strategy for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 40, no. 1, pp. 82–94, 1993.

[2] K. E. Gordon and D. P. Ferris, "Proportional myoelectric control of a virtual object to investigate human efferent control," *Experimental Brain Research*, vol. 159, no. 4, pp. 478–486, Dec. 2004.

[3] I. Kajitani, T. Hoshino, D. Nishikawa, H. Yokoi, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, M. Iwata, D. Keymeulen, and T. Higuchi, "A Gate-Level EHW Chip: Implementing GA Operations and Reconfigurable Hardware on a Single LSI," in *ICES '98: Proceedings of the Second International Conference on Evolvable Systems*. London, UK: Springer-Verlag, 1998, pp. 1–12.

[4] I. Kajitani, T. Hoshino, N. Kajihara, M. Iwata, and T. Higuchi, "An Evolvable Hardware Chip and its Application as a Multi-Function Prosthetic Hand Controller," in *Proceedings of 16th National Conference on Artificial Intelligence (AAAI)*, 1999, pp. 182–187.

[5] I. Kajitani, I. Sekita, N. Otsu, and T. Higuchi, "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*, 2001.

[6] I. Kajitani *et al.*, "A myoelectric controlled prosthetic hand with an evolvable hardware lsi chip," *Technology and Disability*, vol. 15, no. 2, pp. 129–143, 2003.

[7] J. Torresen, "Two-Step Incremental Evolution of a Digital Logic Gate Based Prosthetic Hand Controller," in *Proceedings 4th International Conference on Evolvable Systems (ICES)*, ser. LNCS. Springer, 2001, vol. 2210, pp. 1–13.

[8] ——, "Evolving Both Hardware Subsystems and the Selection of Variants of Such Into An Assembled System," in *Proceedings 16th European Simulation Multiconference (ESM)*. SCS Europe, June 2002, pp. 451–457.

[9] ——, "A scalable approach to evolvable hardware," *Journal of Genetic Programming and Evolvable Machines*, vol. 3, no. 3, pp. 259–282, 2002.

[10] M. Yasunaga, T. Nakamura, I. Yoshihara, and J. Kim, "Genetic Algorithm-based Design Methodology for Pattern Recognition Hardware," in *Proceedings 3rd International Conference on Evolvable Systems (ICES)*, ser. LNCS, vol. 1801. Springer, 2000, pp. 264–273.

[11] L. Sekanina and R. Ruzicka, "Design of the Special Fast Reconfigurable Chip Using Common FPGA," in *Proceedings of the IEEE Conference on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2000, pp. 161–168.

[12] K. Glette, J. Torresen, and M. Yasunaga, "An Online EHW Pattern Recognition System Applied to Face Image Recognition," in *Proceedings Applications of Evolutionary Computing (EvoWorkshops2007)*, ser. LNCS. Springer, 2007, vol. 4448, pp. 271–280.

[13] ——, "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, pp. 463–470.

[14] ——, "An Online EHW Pattern Recognition System Applied to Sonar Spectrum Classification," in *Proceedings 7th International Conference on Evolvable Systems (ICES)*, ser. LNCS. Springer, 2007, vol. 4684, pp. 1–12.

[15] S. Fuji, "Development of prosthetic hand using adaptable control method for human characteristics," in *Proceedings of 5th International Conference on Intelligent Autonomous Systems*, 1998, pp. 360–367.

[16] P. Shenoy, K. Miller, B. Crawford, and R. Rao, "Online Electromyographic Control of a Robotic Prosthesis," *IEEE Transactions on Biomedical Engineering*, 2008, to appear.

[17] S. Bitzer and P. van der Smagt, "Learning EMG control of a robotic hand: towards active prostheses," in *Proceedings IEEE International Conference on Robotics and Automation*, May 2006, pp. 2819–2823.

[18] Biovision, "EMG Amplifier," `www.biovison.eu`.

[19] National Instruments, "USB-6009," `www.ni.com`.

[20] Sonowin, "USI-01 USB Isolator," `www.sonowin.de`.

[21] H. Gray, "Anatomy of the human body," 1918, retrieved from Wikimedia Commons.

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

[23] J. R. Quinlan, *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[24] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[25] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "YALE: Rapid prototyping for complex data mining tasks," in *Proceedings 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD),* Philadelphia, 2006, pp. 935 – 940.

[26] J. Miller and P. Thomson, "Cartesian Genetic Programming," in *Proceedings 3rd European Conference on Genetic Programming (EuroGP)*. Springer, 2000, pp. 121–132.

[27] J. A. Walker and J. F. Miller, "Evolution and Acquisition of Modules in Cartesian Genetic Programming," in *Proceedings 7th European Conference on Genetic Programming (EuroGP)*, ser. LNCS, vol. 3003. Springer, 5-7 Apr. 2004, pp. 187–197.

[28] K. Glette and J. Torresen, "A flexible on-chip evolution system implemented on a Xilinx Virtex-II Pro device," in *Proceedings 6th International Conference on Evolvable Systems (ICES)*, ser. LNCS. Springer, 2005, vol. 3637, pp. 66–75.

[29] D. Goldberg, *Genetic Algorithms in search, optimization, and machine learning*. Addison–Wesley, 1989.