Classification of Electromyographic Signals: Comparing Evolvable Hardware to Conventional Classifiers

Paul Kaufmann, Kyrre Glette, Thiemo Gruber, Marco Platzner, Jim Torresen, and Bernhard Sick

***Abstract*—Evolvable Hardware (EHW) has shown itself to be a promising approach for prosthetic hand controllers. Besides competitive classification performance, EHW classifiers offer self-adaptation, fast training, and a compact implementation. However, evolvable hardware classifiers have not yet been sufficiently compared to state-of-the-art conventional classifiers. In this article, we compare two evolvable hardware approaches to four conventional classification techniques: k-nearest-neighbor, decision trees, artificial neural networks, and support vector machines. We provide all classifiers with features extracted from electromyographic signals taken from forearm muscle contractions, and let the algorithms recognize eight to eleven different kinds of hand movements. We investigate classification accuracy on a fixed data set and stability of classification error rates when new data is introduced. For this purpose, we have recorded a short-term data set from three individuals over three consecutive days and a long-term data set from a single individual over three weeks. Experimental results demonstrate that evolvable hardware approaches are indeed able to compete with state-of-the-art classifiers in terms of classification performance.**

***Index Terms*—evolvable hardware, classification of electromyographic signals, prosthetic hand control, functional unit row architecture, embedded cartesian genetic programming**

I. Introduction

PROSTHETIC HAND CONTROLLERS (PHCs) are usually operated by signals generated by contracting muscles, i.e., electromyographic (EMG) signals. In our work, we focus on two challenges in the design of sophisticated PHCs: First, traditional PHCs only cover a few motions driven by signals of one or two muscle groups, effectively limiting the usefulness of the PHC. A larger set of muscle groups and contraction types would facilitate selection among a greater number of prosthetic device functions. Second, having access to PHCs which adapt themselves to 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 predefined EMG patterns, partly supported by periodic re-training sessions.

Paul Kaufmann and Marco Platzner are with the Department of Computer Science, University of Paderborn, Warburger Str. 100, 33098 Paderborn, Germany, (e-mail: paul.kaufmann@gmail.com, platzner@upb.de)

Kyrre Glette and Jim Torresen are with the Department of Informatics, University of Oslo, Norway, P.O. Box 1080 Blindern, 0316 Oslo, Norway (e-mail: {kyrrehg,jimtoer}@ifi.uio.no)

Thiemo Gruber and Bernhard Sick are with the Intelligent Embedded Systems Lab, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany, (e-mail: {gruber,bsick}@uni-kassel.de)

Evolvable Hardware (EHW) has originally been described in [1], [2] as a combination of automated design of circuits and reconfigurable hardware. The principle is based on Evolutionary Algorithms (EAs) optimizing a circuit with respect to a fitness metric which defines the input/output behavior. The solution phenotype, encoded within a formal representation model, is mapped to a hardware circuit and then tested on a fitness function. The EHW principle allows a system to adapt to a changing environment, recover from faulty states, and react to new resource requirements at run-time. Several applications of EHW have been presented, some of which have been very successful. Examples include data compression for printers [3], analog filters [4], evolved image filters [5], evolved shapes for antennas [6], and high performance reconfigurable caches [7].

While algorithmic aspects of EHW can be investigated by simulation, a proof-of-concept implementation has to balance the following trade-offs. To reduce the size of the search space, the building block granularity of a potential solution should match the building block granularity of the representation model. Not constraining the search space by using general building blocks allows for radically new and even superior solutions but results in a very large search space and excessive optimization times. Besides that, it is also essential for fitness evaluation in hardware that the mapping from the genotype to the phenotype circuit, including circuit reconfiguration, is fast enough. This usually makes it necessary to avoid complex place and route operations in software.

In the context of PHC, evolvable hardware becomes an interesting approach, providing possibilities for self-adaptation, fast training, and compactness. The combination of evolutionary algorithms (EAs) and reconfigurable hardware allows for automatically constructed hardware systems able to adapt their structure to specification changes. Learning to classify electromyographic signals is basically an *incremental learning* problem when it is applied in practice. In general, a learning task is incremental if the training examples that must be used to solve that task become available over time [8]. In our case, it would be possible that a disabled person using an intelligent prosthesis system conducts some exercises under the guidance of that system. The system records and processes the measured data, extending the available training data with new samples. The advantages of such an approach are obvious: First, the system would adapt to the behavior of the disabled person (and not vice versa). Second, the system would also be able to adapt to long-term changes of the behavior of the disabled person.

An interesting question is whether the classifier can then be trained incrementally instead of training it "from scratch". Updating and retraining the classifier using the most recent data may allow to discard old training data, as the essential knowledge that can be extracted from that data is already contained in the classifier's parameters. For Support Vector Machines (SVMs) and other conventional classifier paradigms, which are used for comparison in this work, there are incremental training techniques available (see, e.g., [9]–[11]). Our EHW approaches, while partially employing incremental evolution, have not been analyzed with regard to dynamically updating the classification function.

This article is a substantially extended version of our work first published at the IEEE "Adaptive Hardware and Systems" conference, Noordwijk, 2008 [12]. Generally, the article investigates the classification performance of EHW approaches for a multi-movement prosthesis control application in order to determine whether they are competitive with conventional pattern matching algorithms. To achieve fair comparison, we spent roughly the same effort for all algorithms on finding well-performing configurations. We use grid search for conventional and our expert knowledge for EHW classifiers. Our results mirror general accuracy tendencies instead of peak performances.

We present two different EHW approaches: a coarse-grained and classification-tailored approach and a fine-grained and more general approach. The performance of these approaches is compared to that of four conventional classifiers, one of which is SVM. SVMs are considered to be one of the most powerful classifier methods existing today.

Another important aspect covered by this article in addition to our previous work is the investigation of the classifiers' behavior on EMG data recorded over a long period of time (see Sec. VI-A and Sec. VI-D). Given the inherent ability of EHW systems for autonomous adaptation, classification of non-stationary data is an intriguing challenge for these architectures. We show that using a *moving average* feature extraction scheme and standard classification algorithms, the accuracy rates degrade after a short period of time. Furthermore, we learn that data recorded at a single day is already sufficient to reach high accuracies for all algorithms in our comparison.

We demonstrate that our EHW architectures are competitive with state-of-the-art classifiers. The rather compact distribution of classification rates among different algorithms implies that implementation requirements are the factor most relevant to the actual selection of a classifier for prosthesis control. For embedded system applications with functional and temporal security aspects, adaptable hardware classifiers offer a range of benefits.

The article is structured as follows. Section II describes related work including both traditional classification methods for PHC and classification by EHW. Sections III and IV describe the setup of our EMG sensor system and the signal processing applied to obtain the feature vectors. The tested conventional classifiers as well as the two EHW approaches are detailed in Section V. Descriptions of the conducted experiments, validation schemes, and the obtained results are given in Section VI. Section VII discusses the results and Section VIII concludes our work.

II. Related Work

The first known prosthesis controlled by electromyography signals is the "Hüfner Hand" [13]. In 1948, Reiter implemented a prosthesis controller using one EMG channel to encode "open" and "close" movements of an artificial hand [14]. A quick contraction and relaxation triggered the "open" movement and a steady force contraction caused the prosthesis to gradually close the hand. Driven by the availability of compact electronic components, the area of prosthesis control gained more popularity in the 1960s and 1970s. Substantial effort went into defining strategies for robust selection of prosthesis actions from muscular activities [15]–[19]. The first commercial system was offered in the early 1960s [20]. In the following, we discuss modern EMG signal classification techniques with both conventional systems and evolvable hardware.

A. Classification of EMG Signals with Conventional Systems

Modern conventional upper limb prosthesis control systems typically use rudimentary algorithms to derive information for steering a prosthesis. There are three popular methods of acting on the signal of a muscle, or more precisely a group of muscles, which consider:

- the intensity of muscular activity. For a single channel, typically two intensity thresholds separate three muscle states – relaxed, slightly contracted, and contracted – allowing the prosthesis to perform, for example, "open" and "close" movements [15].
- the muscular activity growth rate. Similar to the previous method, two thresholds for the speed of the performed contraction partition the channel output into three states [16].
- multiple groups of muscles, discriminating between contracted / non-contracted muscles to encode the prosthesis action. For example, using two channels, up to three prosthesis actions and a neutral state can be selected.

For multi-functional prostheses, the control system can use quick co-contractions to switch between different activity modes (e.g., switching between the "grasping" and the "rotating" modes for an artificial hand). However, such a control mechanism is not intuitive and has to be learned by the user.

Pattern recognition algorithms enable a different way of extracting information of muscular activity. For example, instead of requiring the user to be familiar with the activation of some groups of muscles to trigger an "open" movement, pattern recognition algorithms are able to extract the natural hand "open" impulse from the superimposed EMG signals of the forearm. Pattern recognition methods allow for intuitive control and are also capable of discriminating between a larger number of distinct movements. However, multiple EMG channels are needed for a robust detection of multiple movements.

Early attempts to use pattern recognition algorithms were made by Finely [21], Herberts [17], and Graupe and Cline [19]. In today's literature on EMG signal classification, the signal processing chain is often broken down into three algorithmic components: *feature extraction*, *dimensionality reduction* and *pattern classification*. The feature extraction step isolates application-specific attributes from the EMG signal. Dimensionality reduction decreases the amount of data for a more robust and accurate classification by selecting or projecting features. The final pattern classification step determines the predefined category to which the input data belongs. The complete processing chain has to be carefully balanced; in particular, the choice of a pattern recognition algorithm and the selected features contribute significantly to the recognition accuracy.

Feature extraction schemes for continuous prosthesis control act in a sliding-window manner. That is, a feature set is calculated for each window of the data, where the windows are typically up to 300 milliseconds in length and are selected according to the classification rate of the prosthesis controller.

Historically, the development of computationally efficient algorithms has been of the utmost importance since prosthesis controllers typically run on battery-powered embedded systems. Here, feature extraction methods acting in the time domain (TD) are often regarded as being well-suited because of their simplicity. Examples for time domain methods widely used in EMG classification are mean absolute value (MAV) [22]–[27], zero crossing (ZC) [22], [23], slope sign changes (SSC) [22], [23], [28] and waveform length (WL) [22], [23], [28].

EMG electrodes, being electrically only loosely attached to the skin surface, tend to act as antennas collecting noise from power lines, adjacent electric and electronic prosthesis subsystems, and other electromagnetic sources. Time domain methods in general and methods using amplitude-based features in particular have difficulties dealing with such noise and also with the effects of varying skin conductance. Consequently, a significant part of related work concentrates on frequency domain based feature extraction to suppress noisy influences. Fourier transformation (FT) and short-time Fourier transformation (STFT) [23], [29], [30] are among the most popular methods. Capturing information from the time and frequency domains, wavelet transformation (WT) [23], [29] and wavelet packet transformation (WPT) [23], [30], [31] have also been successfully studied for recognition of EMG signals. Despite being computationally expensive, frequency domain feature extraction schemes are feasible on today's high-performance embedded systems.

Reducing the dimensionality of the feature space while preserving essential information may increase a classifier's generalization ability. Additionally, irrelevant information that is skipped in this step reduces the amount of data to be processed by the classifier. Dimensionality reduction can be implemented as feature selection that aims at maximizing the probability of an correct classification [31], [32]. For the classification of EMG signals, the *projection* of features is quite popular. Projection creates a new and generally smaller feature set by combining original features in a linear or non-linear way. Some of the employed algorithms are principle component analysis (PCA) [31], [32], linear and non-linear discriminant analysis (LDA, NLDA) [33] and self-organizing feature maps (SOFM) [33].

The last step of the signal processing chain covers pattern recognition. A dominant part of related work uses artificial neural network (ANN) classifiers [22], [34]–[36]. More recent work also introduces support vector machines (SVM) for EMG signal classification [26], [37], [38], as well as Bayesian classifiers [31], [39], [40], fuzzy classifiers [41], [42], Gaussian mixtures [43], and hidden Markov models [44].

A compact overview of methods for preparing, processing, and classifying EMG signals is given by Zecca et al. [45] and Parker et al. [46].

B. Classification of EMG Signals with EHW

An early use of EHW for pattern recognition was reported by Higuchi et al. [47]. Their architecture was originally applied to character classification but was later used for classification in a prosthetic hand controller (PHC) [48], [49]. It employed 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 a compact and adaptable system. The controller was trained with feature vectors extracted from EMG data where one input signal consisted of four channels at a resolution of four bits. The classifier distinguished between six different kinds of 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 showed a competitive classification rate for evolved circuits compared to artificial neural networks (ANNs), it was noted that the size of the employed data set might be insufficient; this 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 (800 ms) for the EHW approach was 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 [50]–[53] conducted experiments on incremental evolution using a EHW architecture. The two-layered architecture consisted of `AND-OR` matrices followed by a selector layer. The `AND-OR` matrices were evolved in the first step followed by the evolution of the selectors. In addition, the best subsystems from different runs were combined into one system. The results showed that a two-step incremental approach can lead to a better generalization performance and shorter computation times than traditional one-step evolution and ANN.

EHW classification architectures applied to domains other than PHC include, for example, the function level evolution of [54]. 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 [55], [56]. LoDETT allows for high accuracy classification on problems with a much higher number of inputs and outputs. However, the system does not implement 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.

A major challenge is to map evolved circuits to reconfigurable hardware at runtime, since for today's FPGAs no commercial tools exist that support online reconfiguration at the fine-grained 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 [57]. This method obtains virtual reconfigurability by changing register values and multiplexer control signals in the user circuit. The advantages of this high-level technique are fast reconfiguration and applicability to all SRAM-based FPGAs. However, the method potentially requires more logic resources.

Work has also been carried out on bitstream reverse engineering for recent FPGAs. This was applied to EHW circuits in [58] and adapted to newer FPGA devices in [59]. The approach allows for runtime reconfiguration of FPGA lookup table (LUT) contents. However, changing an FPGA's routing resources is more complex and has not been achieved in the context of EHW. While this approach has the potential to save logic resources, it requires a low-level specification of the circuit and yields potentially longer reconfiguration times. An alternative approach to bitstream based reconfiguration is an intermediate-level shift register based method as described in [60]. This also allows for reconfiguration of LUTs, however the method may not give access to all resources on newer devices.

Using the virtual reconfiguration technique, Glette et al. proposed an online evolvable EHW architecture, the Functional Unit Row (FUR) architecture, for classification tasks [61]–[63]. A device specific, shift register based implementation of the architecture has been proposed [64] as well. 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. The architecture employs function level modules as well as a method of dividing the evolution into several smaller tasks.

Finally, the same architecture was also applied to PHC and compared to another approach based on embedded cartesian genetic programming (ECGP) [65]. The ECGP-based approach uses automatic definition of sub-functions and achieves similar classification accuracies despite the fact that evolution is performed on a more general architecture with lower level primitives. Further investigation of the FUR and ECGP approaches continued in [12].

From simulated reconstructions of earlier proposed EHW architectures of Kajitani et al. [48] and Torresen [50], a comparison of different EHW approaches to classification for PHC has been undertaken. The results have indicated better performance for the FUR and ECGP approaches, with better classification accuracies as well as faster evolution. We can think at least of three reasons that the FUR and ECGP-based architectures are more successful than the early EHW classifiers of Kajitani et al. and Torresen. Firstly, both methods share the same principle of decomposing a recognition function for a single category into an ensemble of multiple and diverse recognition functions. The outputs are combined into a graded metric and compared to the outputs of other category classifiers. The reason for the decomposition is to reduced the complexity a single classification function needs to realize. In return, ensemble classifiers allow to tackle more complex classification tasks.

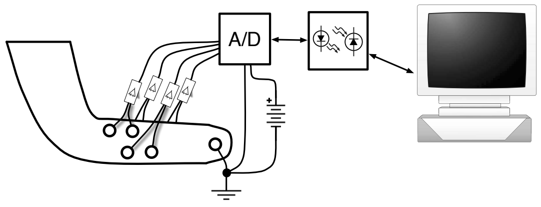

Fig. 1. The EMG measurement system. The analog signal processing part is decoupled from the PC and powered by a battery. EMG signal amplifiers are placed close to the electrodes to reduce noise. Signal processing is implemented completely on the PC.

The second reason lies in the complexity of the basic building blocks. Both early EHW classifiers employ *Programmable Logic Arrays* (PLA). PLAs compute Boolean *sum-of-product* (SOP) functions. Given the restricted functional set of `AND`, `OR`, and `NOT` gates and a fixed circuit depth of two levels, a search algorithm is typically faster at finding solutions when exploring the unrestricted space of digital circuits comprising all Boolean gates and without routing limitations. In contrast to the ECGP representation model, which allows for exploration of the unrestricted space of Boolean circuits, the FUR architecture employes more complex functional elements using comparators acting on binary encoded numbers.

The third reason for the success of the FUR and ECGP-based classifiers is that both architectures process input data at much higher resolution as the early EHW classifiers. Kajitani's et al. and Torresen's PLA based approaches use 4 bits to encode a signal value. As signal values are, for instance, binary encoded, evolution has also to master the implicit task of data type conversion or at least has to deal with differently encoded input data types and function block inputs [49], [65]. Inputs of FUR's basic building blocks, in contrast, match the encoding of the input data types. In the ECGP-based architecture, input values are linearly quantized by a 1-out-of-500 encoder unifying the input data and functional block encodings.

Details on FUR and ECGP-based architectures will be presented in Sec. V-A.

III. EMG Signal Measurement

We use different measurement systems for stationary and portable EMG signal recording. The stationary system comprises four components: EMG sensors (Tyco Arbo*, Ag/AgCl, 35 mm), amplifiers (Biovison [66]), A/D converters (N.I. [67]), and a standard computer. The system shown in Fig. 1 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

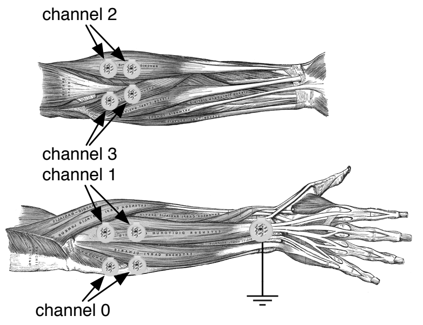

Fig. 2. Sensor placement (muscle anatomy taken from [70]).

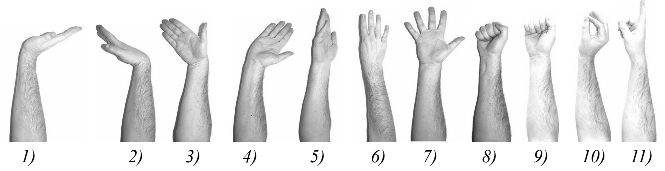

Fig. 3. Motion classes: *1)* extension, *2)* flexion, *3)* ulnar deviation, *4)* radial deviation, *5)* pronation, *6)* supination, *7)* open, *8)*, close *9)* key grip, *10)* pincer grip and *11)* extract index finger.

in the analog signal domain and a reproducible biomechanical experimental setup. To reduce noise, we employ an optical bridge (Sonowin [68]) 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 are placed as near as 10 cm to the skin-attached electrodes in order to minimize parasitic inductance. For portable EMG data acquisition, we use a MindMedia Nexus 10 Biofeedback System [69] to continuously monitor four EMG sensor channels with 24 bit resolution at a sampling rate of 2048 Hz.

We place the four electrode pairs on the top, bottom, medial, and lateral sides of the forearm with the reference at the wrist, as shown in Fig. 2. The exact electrode positions are specifically determined for the test subject to obtain pronounced signals. A reproducible biomechanical experiment setup is an important requirement for such a measurement system. Thus, after the initial calibration we mark the electrode positions to be able to re-establish the experimental setup on different days.

In a single run of the experiment, the test subject has to perform a sequence of different movements. Some of these movements are depicted in Fig. 3. Each movement starts with a relaxation phase followed by a contraction phase, as shown in Fig. 4(a). The EMG signal for the contraction part divides roughly into a one second phase at the onset of the contraction containing the transient components of the EMG signal and a subsequent steady state phase which corresponds to a constant force contraction. We use the steady phase for classification.

For the first experiment, we use the stationary EMG measurement system to record the movements *1)* to *8)*, as presented in Fig. 3, by three individuals on three consecutive days. In each of the nine sessions, an individual repeats the movement sequence 20 times. Each movement consists of

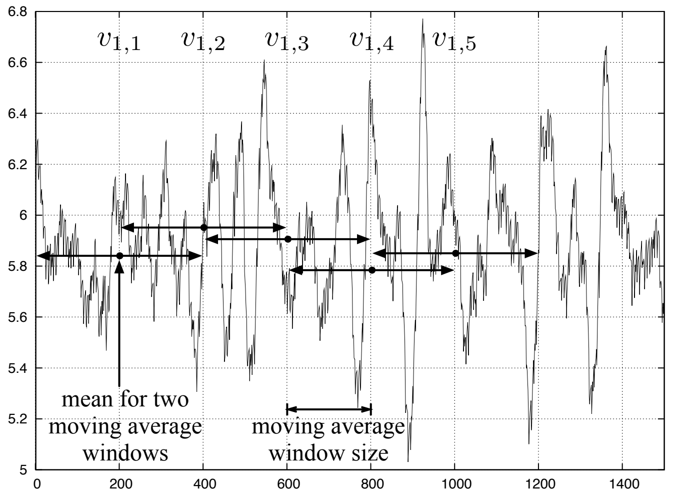

Fig. 5. $(200, 2, 5)$ feature extraction scheme for a single channel signal: five mean average values over 2×200 samples define a feature vector.

a nine second relaxation and an eleven second contraction phase. The data is recorded at a 6 kHz sampling rate. For the second experiment, we use the portable EMG measurement system to have a single individual collect data from all eleven movements presented in Fig. 3 over 21 days. Similar to the previous setup, the subject records a single sequence of movements 5 to 6 times a day. In total, 121 sessions are conducted during different times of a day. Each movement starts with a relaxation phase of about 4 seconds followed by a contraction phase that lasts about 5 seconds. The sampling rate in this experiment is set to 2048 Hz.

The data from the first experiment is analyzed by the *"Day1–3"* and *"2of3"* evaluation schemes and the data from the second experiment is analyzed by the *"121"* evaluation scheme. The definitions of these schemes will be given in Section VI-A.

IV. Signal Preprocessing and Feature Extraction

Signal preprocessing and feature extraction is done completely in the digital domain. Our method is inspired by the *mean average value* (MAV) scheme of Kajitani et al. [49]. We designed a method for an efficient computation procedure by subdividing the input signals into small disjoint intervals. Partial solutions are computed for each interval and reused for feature vector extraction. This minimizes redundant computations.

The feature vector extraction scheme is defined by a 3-tuple (r, s, t) where r is the size of the moving average window in terms of samples, s is the number of moving average windows used to compute a single feature value, and t is the number of values in the feature vector calculated for a particular channel. With k being the number of signal channels, p being an index of a signal sample, d_{ip}, $i = 1, \dots, k$, being the DC compensated raw signal, and $j = 1, \dots, t$, a single feature vector, $v \in \mathbb{R}^{kt}$ is calculated as:

$$v = (v)_{ij} = -\log\left(\frac{1}{rs}\sum_{l=(j-1)r+1}^{(j-1+s)r} |d_{il}|\right).$$

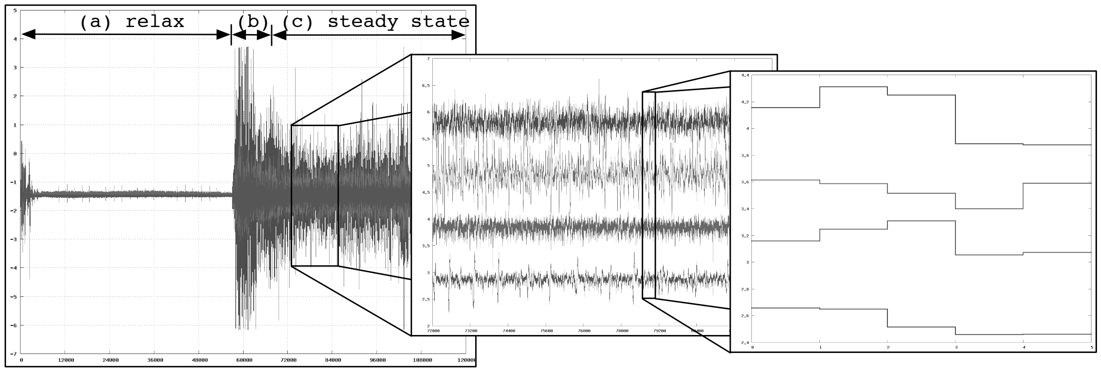

Fig. 4. EMG signal preprocessing. The left figure shows the raw signals for all four channels, consisting of *a)* a relaxation phase, *b)* a transient phase with intensified activity, and *c)* a steady state contraction phase. The center figure presents the DC offset-compensated and rectified signals from the four channels in the steady state phase, and the right figure shows a single extracted feature vector using the $(200, 2, 5)$ scheme.

Thus, a single feature vector in the (r, s, t) scheme consists of $k \times t$ values calculated over $r(s + t - 1)f^{-1}$ seconds where f denotes the sampling rate.

To demonstrate feature vector calculation, we exemplarily compute a $(200, 2, 5)$ scheme for a signal sampled at 6kHz (Fig. 5). The first element of $v = (v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4}, v_{1,5})$ is computed as $v_{1,1} = -log([200 \cdot 2]^{-1} \cdot [|d_{1,1}| + |d_{1,2}| + \cdots + |d_{1400}|])$. Similarly, remaining elements of v rely on means computed using raw signal values with indices $(201, \ldots, 600)$, $(401, \ldots, 800)$, $(601, \ldots, 1000)$, and $(801, \ldots, 1200)$. Altogether, the first feature vector relies on data recorded during $200 \cdot (2 + 5 - 1) \cdot 6000^{-1} = 200$[ms].

Our feature extraction scheme is tailored for the continuous operation mode of a prosthesis controller running on a small embedded system. With the update frequency f_u for the feature extraction and classification chain, the window size r should be set to $f \cdot f_u^{-1}$ to allow the reuse of $(v)_{ij}, i = 1, \ldots, k, j = 2, \ldots, t$ for the calculation of the following feature vector. With $t = f \cdot f_u^{-1}$ only the averages $(v)_{ij}, i = 1, \ldots, k, j = t$ have to be updated. Coming back to the example of Fig. 5, we set the update frequency to 30, which results in a moving average window of $r = 6000 \cdot 30^{-1} = 200$ samples. We can thus reuse $v_{1,2}, v_{1,3}, v_{1,4}, v_{1,5}$ of the first feature vector and thereby relabel the elements to $v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4}$; consequently, for $v_{1,5}$, only the sum of raw signal elements with indices $(1201, \ldots, 1400)$ needs to be computed. $v_{1,5}$ is then the sum of partial results

$$\sum_{l=1001}^{1200} |d_{1l}| \text{ and } \sum_{l=1201}^{1400} |d_{1l}|$$

divided by $rs = 200 \cdot 2$, so that the first partial sum is already computed for the first feature vector.

In our experiments we set the update frequency to $f(r(s + t - 1))^{-1}$. Thus, the feature vectors are computed on disjoint data. If further feature normalization is applied by the classification algorithm, the channels are treated independently. For the *"Day1–3"*, *"2of3"*, and *"121"* experiments, the feature vectors are computed by a $(300, 2, 5)$ and $(100, 2, 5)$ scheme, respectively. Thus, the feature vectors consist of 20 values and the corresponding label. Both feature extraction schemes use the data of roughly a third of a second, which is a realistic assumption for prosthesis control.

V. Classification Paradigms

This chapter introduces and compares two evolvable hardware approaches and four state-of-the-art classifiers. The first EHW approach uses simple Boolean gates to construct pattern recognition circuits. The circuits are encoded within a variant of the *Embedded Cartesian Genetic Programming* (ECGP) representation model. The second EHW approach uses comparators acting on binary encoded numbers. A pattern recognition circuit in this architecture is represented by multiple comparators feeding into a single AND gate. We refer to the first EHW classifier as the ECGP-based and to the second classifier as the *Functional Unit Row* (FUR) architecture.

To evaluate the performances of EHW classifiers, we compare their results to a reference algorithm. As we cannot expect a single algorithm to perform best among all applications and data sets, we select four conventional classifier paradigms, *Artificial Neural Networks* (ANN), *Support Vector Machines* (SVM), *Decision Trees* (DT) and *k-nearest-neighbors* (*k*NN), that realize different forms of decision boundaries between classes.

This section continues with the description of the common structure of our EHW classifiers then proceeds with a detailed presentation of the ECGP-based and FUR architectures, compares their representation model, algorithmic and classification properties, and finishes by sketching the conventional classifiers.

A. Evolvable Hardware Classifiers

Both proposed EHW architectures for classification tasks have a common high-level structure, see Fig. 6. The input pattern is presented to a number of sub-circuits called classifier circuits (CCs). A CC is a function with a binary output indicating a match or a mismatch. Several CCs are grouped together and their outputs are fed into a counter summarizing the number of matches. A set of CCs and the counter are denoted as a Category Detection Module (CDM) since the CCs are detecting the same category. The classifier's global decision is made by selecting the category (CDM output) with

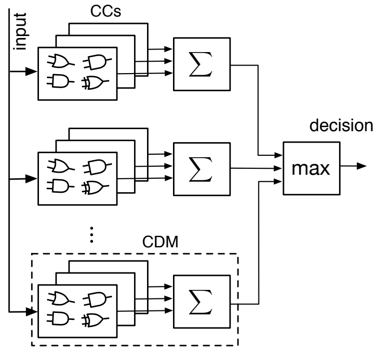

Fig. 6. High-level structure of the EHW classifier architectures.

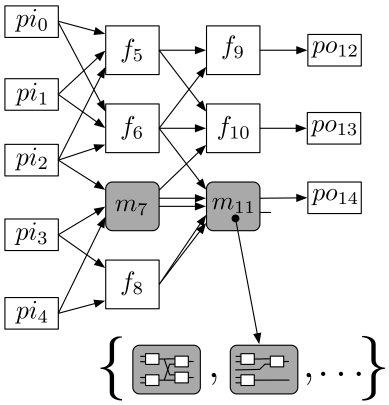

Fig. 8. The Embedded Cartesian Genetic Programming Model automatically creates modules as compositions of primitive nodes.

the highest number of matches. In case of a tie, the category with the lower index wins.

The presented architecture belongs to the family of *ensemble classifiers*. The idea of ensemble classifiers is to train a set of diverse classifiers on the same training set and to combine the predictions. The presented architecture can also be seen as an extension of the early EHW architectures of Kajitani et al. [71] and Torresen [50]. Kajitani used a single Boolean sum-of-products (SOP) function to realize a CDM while Torresen utilized a two-stage scheme of multiple SOPs and a SOP subset selector for CDM implementation. Similar to the architecture presented in Fig. 6, the maximal number of activated SOP's in Torresen's architecture defines the global decision.

1) The Embedded Cartesian Genetic Program Classification Architecture: The first EHW-based classifier relies on a variant of the ECGP model. ECGP is an extension of the popular *Cartesian Genetic Program (CGP)* model [72] which is a structural model that arranges functional blocks in a two-dimensional geometric layout. In contrast to a genetic program [73], which relies on trees to represent evolved functions, a CGP in its original formalization is essentially a restricted *directed acyclic graph* (DAG). The restrictions are imposed by the array structure which limits the number of overall functional blocks and the interconnect depth.

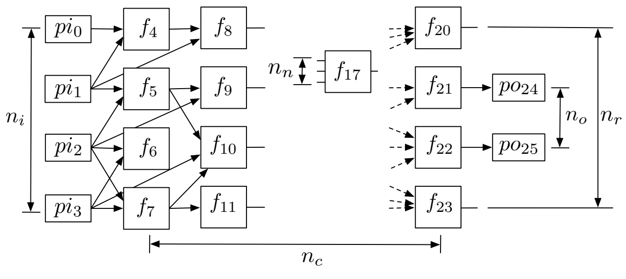

Fig. 7. Cartesian Genetic Programming Model.

Formally, a CGP model as defined in [72] consists of $n_c \times n_r$ functional blocks, n_i primary inputs, and n_o primary outputs. A functional block has n_n inputs and implements one out of n_f different functions on these inputs. While the primary inputs and outputs can be connected to any functional block input and output, respectively, the connectivity of the functional block inputs is restricted. The input of a functional block at column c may only be connected to the outputs of blocks in columns $c - l, \dots, c - 1$ as well as to the primary inputs. The levels-back parameter l restricts wiring to local connections. Since only feed-forward connections are allowed, the creation of combinational feedback loops is avoided. Fig. 7 shows an example of a CGP model together with its parameters. The model in this example has five columns, four rows, four primary inputs, and two primary outputs.

To improve scalability, Walker et al. [74] introduced automatic acquisition and reuse of sub-functions (modules) to CGP. Coined as *Embedded* CGP, it configures the CGP genotype as a single line of functional nodes ($n_r = 1, l = n_c$) and defines modules as compositions of primitive nodes. While in the original work modules are composed from randomly selected primary nodes, in our work we aggregate primitive nodes that have persisted in the genotype for a higher number of generations to assemble a new module. The rationale is that aged nodes are more likely to contribute directly or indirectly to an individual's success and should, therefore, be preferred over randomly selected nodes for module creation. Initially, we described the age-based module creation technique in [75].

In our work we use a standard $(1+1)$ Evolutionary Strategy (ES) algorithm. In every generation, the offspring individual is derived from the parent by a mutation operator. The offspring becomes the new parent except for the case when the parent has a higher fitness. We have selected the $(1+1)$ ES variant as it has demonstrated better convergence behavior than other $(1+n)$ ES variants, with $n > 1$.

The fitness is defined as candidate solution's classification accuracy on the training data set. More precisely, for the set of labeled training feature vectors $X = (x, l)_j$ the fitness f of an evolved classifier circuit c is defined as:

$$f(c) = |X|^{-1} \sum_{(x,l)_j \in X} \begin{cases} 1 & : \text{ if } c(x_j) = l_j, \\ 0 & : \text{ otherwise.} \end{cases}$$

The complete set of ECGP model parameters and the ES configuration are summarized in Tab. I. As our goal is to evolve digital circuits that can be easily mapped to generic FPGAs, we are using 4-input lookup tables as functional

TABLE I
PARAMETERS FOR THE EVOLUTION OF THE ECGP EHW CLASSIFIER.

n_i / n_o / n_r / n_c	10000 / 1 / 1 / 1000–1500
n_n / n_f	4 / $\lvert\mathbb{B}^4\rvert$
fitness evaluations per generation	1
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	3

blocks and single wires for the connections. The mutation operator either re-routes an input connection of a functional block or selects block's function randomly. The population is initialized randomly with genotypes containing 1000 logic blocks. Depending on the created modules, the genotype is allowed to grow up to 1500 blocks. The rather large genotype size helps to avoid stagnation during the final search phase.

Our ECGP-based classifier architecture is configured to evolve 24 and 20 ECGP classifier circuits per category for the *Day1–3* and *2of3* experiments and for the *121* experiment, respectively. Each of the 20 values in a feature vector are linearly quantized to a 9-bit representation and input to a 1-out-of-500 encoder. The resulting 20×500 bits are then fed into a classifier circuit. We noticed no further accuracy improvement in our setup when increasing quantization's precision. However, our feature extraction scheme considers only steady state phases of a muscle contraction. In order to consider the contraction's initial phase, which, as illustrated in Fig. 4, has much higher amplitudes than the steady-state phase, finer quantization would be required.

The remaining ECGP parameters in Tab. I, except the module size, follow the standard configuration presented in [75]. We found that larger module sizes slow down the convergence rate. In the light of the more randomized nature of EMG signals, when compared to arithmetic functions, recurrent and symmetric patterns within EMG signals may be more sparse and compact.

2) The Functional Unit Row EHW Architecture: The second EHW-based classifier investigated in this article is specifically tailored towards classification tasks and online evolution. A regular structure has been chosen to make the mapping to an FPGA-implemented circuit as direct as possible. Furthermore, the choice of functionality in the processing elements, the width of the data elements, as well as the dimensional parameters of the architecture, have been determined based on the input data and through experimentation. 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 [57]. By writing the evolved genome bitstream to these registers, one obtains the phenotype circuit which can then be evaluated. The architecture is presented at a hardware-abstracted level in the following paragraphs. Details about the implementation can be found in [61]. A more hardware-specific implementation, which saves resources by utilizing

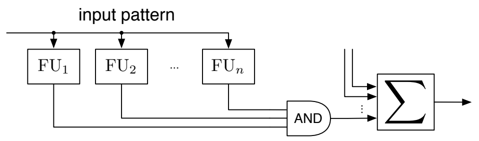

Fig. 9. Category Classifier (CC): n Functional Units (FUs) are connected to an n-input AND gate. Multiple CCs with a subsequent counter for activated CCs define a CDM.

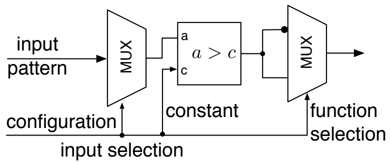

Fig. 10. Functional Unit (FU): The data MUX selects which of the input data is fed to the functions ">" and "≤". The constant c is given by the configuration lines. Finally, a result MUX selects which of the function results is returned.

lower-level FPGA reconfiguration abilities, has been explored in [64]. This implementation would also have a straightforward mapping to a bitstream manipulation approach using partial reconfiguration, such as in [58].

The classifier system consists of P CDMs, one for each category C_p to be classified—see Fig. 6. The input data to be classified is presented to each CDM concurrently on a common input bus. Each CDM consists of m CCs, or functional unit (FU) rows, see Fig. 9. Each FU row consists of n FUs. The inputs to the circuit are passed on to the inputs of each FU. The 1-bit output 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 to activate a rule. 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.

The FUs are the reconfigurable elements of the architecture. As seen in Fig. 10, 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 Fig. 10 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 the suitable inputs is that not all inputs have to be connected. A *direct* implementation as done in the LoDETT system [56] would have required $n = 20$ FUs in a row for the PHC application. In contrast, our system uses $n = 8$ units. The rationale is that not all of the inputs are necessary for the classification.

The 20 normalized amplitudes (5×4 channels) of the input

EMG signal are converted to 8-bit values before they become inputs to the FUs. Based on the data elements of the input being 8-bit values, the functions available in the FU elements have been chosen to be *greater than* and *less than or equal*. Through experiments these functions were seen to work well. Intuitively, this allows for discriminating signals by looking at the different channels' amplitudes. The input is compared to the constant, which is also 8 bits, to give true or false as the output. This can be summarized as follows, with a being the selected input value, and c the constant value, FU's output is defined as:

$$\mathrm{FU}(a,c) = \begin{cases} 1 & : \quad a > c \\ 0 & : \quad \text{otherwise.} \end{cases}$$

A bit string (genome) is associated with each individual in the population. Each FU is encoded in the genome string with 5, 1, and 8 bits for the *feature address, function type,* and *constant*, respectively. This gives a total of $B_{unit} = 14$ bits for each unit. With $n = 8$, the total number of bits in the genotype for one FU row then is $B_{tot} = B_{unit} \cdot n = 112$.

For a more efficient implementation than the standard single bit mutation probability approach, a customized scheme has been adopted as the mutation operator. The number of mutations, N_{mut}, is randomly selected first. Then, N_{mut} random places are mutated (bit-flipped), instead of calculating a random number for every bit in the genome. A standard single-point crossover operator is applied directly to the bit string.

Evolving the whole classification system in one run would give a very long genome and, therefore, an incremental approach is chosen. Each category detector CDM_p is evolved separately. 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.

Fig. 11 illustrates a complete FUR adaptable classifier system. That is, classifier containing all CDMs, while evolution performs further adaptation/tuning in parallel. This could work by having a CPU running the EA and evaluating candidate CCs in a separate evaluation module. The evaluation module would only need to contain enough hardware resources for one CC, and thus requires much less hardware resources than the operational classifier.

A certain subset of the available vectors, V_t, is used for training the system, while the remaining, V_v, is 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

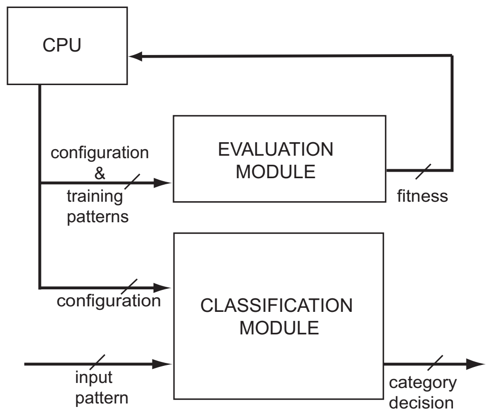

Fig. 11. Run-time adaptation architecture for EHW classifiers.

TABLE II
MUTATIONS RATE DISTRIBUTION FOR THE FUR ARCHITECTURE.

N_{mut}	0	1	2	3
p	0.1	0.6	0.2	0.1

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 $A = 4$ is used. This emphasis on the positive matches for C_p has shown to speed up the evolution. Further, the architecture parameters $n = 8$ FUs per row and 20 rows per CDM are used. A maximum of 150 generations is allowed for each evolution run, however, evolution is terminated earlier in case maximum fitness value is reached. We have implemented a Simple GA [76] with elitism, a population size of 32 and a crossover rate of 0.9. The mutation rate distribution is summarized in Table II. That is, the mutation operator is applied once with a probability 0.6, twice with a probability of 0.2, and so on.

3) Evolvable Hardware Classifier Comparison: While in the ECGP-based approach, the genotype representing the entire classifier is subject to evolutionary optimization, the FUR-based approach employs the concept of incremental evolution. An advantage of this is a simpler search, which in turn should reduce the total evolution time and add the ability to start classifying before all the sub-circuits have evolved, i.e., one could start classifying as soon as one CC for each category has evolved. Both EHW approaches also 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 evolutionary algorithm. While the ECGP approach extracts building blocks automatically, and thus is a very general approach, the FUR architecture uses *a priori* knowledge in the form of predefined building blocks found to be good for classification. The EHW approaches allow for run-time adaptation of hardware and online evolution, as shown in Fig. 11.

Both EHW approaches rely on the principle of having a graded output for each of the categories, which are 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 ensemble approaches such as random decision forests [77]: whereas *single* decision trees (DTs) can be prone to overfitting, having a *collection* of slightly different DTs for one category can significantly reduce this effect.

The decision boundary of ECGP-based classifier is implemented as a Boolean combination of signal values and constants. Therefore, similar to kNN, the decision boundary is a composition of straight lines. The FUR approach compares signal values with constants, thus, analogous to DTs, realizes a decision boundary with sections of straight lines that must be parallel to the axes of the input space spanned by all attributes.

B. Conventional Classifiers

To investigate whether the evolvable hardware approaches are competitive with state-of-the-art classifier paradigms in the field of machine learning, we have chosen four "conventional" classifiers for a comparison. These classifiers differ greatly in the ways they form the decision boundary between any two classes. Building a classifier basically consists of three tasks: (1) choosing the functional form of the classifier, (2) defining an objective function together with an optimization technique to minimize or maximize that objective function, and (3) using both together with a sample data set to find values for the parameters of the classifier (i.e., to train the classifier). In [78], Fisch et al. have shown that not only (1) but also (2) influence the form of the decision boundary and, thus, the performance of the classifier.

1) Nearest Neighbor Classifiers: A *k-nearest-neighbor* (kNN) classifier is a very simple, data-based classification approach [79], i.e., it does not require any training phase as described above. From an analysis of the very general "bias/variance dilemma" for classification tasks [80], [81], which states that variance dominates bias, it can be concluded that classifiers with a low complexity—corresponding roughly to the number of free parameters—perform well in many classification tasks. With this property in mind, we select kNN, having only one parameter (k), as a baseline method. We consider kNN as a reference method that is expected to be outperformed by other kinds of classifiers. However, kNN classifiers require storing of and iteration through all of the sample vectors of the training data set during the operational phase, thus they are barely suitable for many real applications, which require a compact and fast implementation. In a kNN classifier, the form of the decision boundary between two classes is defined locally by the k nearest (using a Euclidean distance measure) samples in the training set. Therefore, the decision boundary is composed of sections of straight lines. In our experiments, the number of neighbors is set to $k = 7$.

2) Decision Trees: Decision trees (DT) can be used for the classification of numerical as well as categorical data [79]. A DT realizes a set of human-interpretable *if-then* rules. In a tree structure, each leaf node represents a classification decision, each non-leaf node evaluates an attribute associated with that node. An input sample vector is classified by successive tests from the root of a DT down to a leaf. Our motivation for including DT in our comparison is the possibility for extracting human-interpretable rules. Moreover, DT can be regarded as a state-of-the-art technique to solve classification problems. DT realize a decision boundary with sections of straight lines that must be parallel to the axes of the input space spanned by all attributes. This restriction basically enables interpretability of training results in the form of "if-then-else" rules. This is one reason why DT are used for many practical applications.

In our experiments, we use the C4.5 algorithm [82] to build a 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 overfitting of the classifier to the training data set. The confidence threshold for pruning is set to 0.25 and the minimum number of instances per leaf are 2.

3) Support Vector Machines: Support vector machines (SVM) use a hyperplane to separate any two classes [83], [84]. For problems that cannot be linearly separated in the input space, SVM 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 having a maximal margin are called optimal, 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. C-SVMs, which are used here, introduce slack variables—being subject to minimization as well—to allow a certain degree of misclassification. With the aid of nonlinear kernel functions, SVM are able to realize arbitrary nonlinear decision boundaries in the input space. The key advantage of SVM is that they are based on the principle of structural risk minimization which typically leads to a very good generalization performance. Thus, one could expect SVM to yield very good results in our comparison.

In our experiments, we use a C-SVM with a Gaussian (radial basis function, i.e., RBF) kernel. For the *"Day1–3"* and *"2of3"* experiments, we set $C = 0.5$ and $\gamma = 0.175$. The parameters for the *"121"* experiment are $C = 3$ and $\gamma = 1$.

4) Neural Networks: Multilayer perceptrons (MLP), also known as backpropagation networks, are neural networks that are biologically inspired [85], [86]. They aim at being discriminative, but they lack a built-in mechanism for structural risk minimization like SVM. Thus, good generalization properties must be assured by means of comprehensive cross-validation or bootstrapping tests. Like SVM, MLP are able to realize nonlinear decision boundaries. A major difference to SVM is that the structure of the classifier (e.g., number of hidden neurons) must be fine-tuned by hand. We have chosen MLP for our comparison because of their use as a reference classifier in related work on EHW.

In our experiments, we use a MLP with 20 input neurons (for the 20 input features), 8 and 11 output neurons (for the 8 and 11 classes), and one hidden layer consisting of 32 neurons. MLP are trained using the backpropagation algorithm with an additional momentum term. The learning rate is set to 0.3, the momentum rate to 0.2, and the number of training epochs is 500.

VI. Experiments and Results

This section presents and compares the performance of the proposed EHW and conventional state-of-the-art classification algorithms. First, we present the classification accuracy calculation schemes. Then, we evaluate the classification algorithms and rank them using three different benchmarks.

A. Validation Schemes

Our first experiment is based on the stationary EMG measurement system and uses cross-validation for evaluation. Cross validation is a partitioning scheme splitting the data into similar-sized chunks, selecting one chunk for performance testing and the remaining chunks for training. This scheme is repeated until all chunks have served for testing. The considered classifiers are trained anew for each test chunk. The data of the first experiment is used to define two benchmarks: In the first benchmark, referred to as *Day1–3* benchmark, we investigate the asymptotic classifier accuracy by merging and shuffling all data from a single individual and evaluating the proposed classifiers with 10-fold cross validation. In the second benchmark, referred to as *2of3* benchmark, we aim at investigating the classifier's generalization capabilities. To this end, we use a 3-fold cross validation defining the data partitioning by the recorded day. Thus, data from two days are used for training and the data from the remaining day are used for testing.

The second experiment, referred to as *121* benchmark, is undertaken with our portable EMG measurement system and investigates longer-term effects on the performance of classifying EMG signals. The main question is whether and how much the classification accuracy degrades over time if the classifiers are not being trained continuously. Despite its importance, this issue has not yet been investigated. The main part of related work on EMG signal classification focuses on accuracy improvement and the number of discriminated movements. Assuming the EMG signal changes over time, one needs to study the nature of the change, the way it can be measured, and the effects has on the classification accuracy. To design practical prosthesis controllers, one finally has to devise appropriate feature extraction schemes compensating for EMG signal variations and also look at the interdependency between a continuously retrained controller and the amputee interacting with the prosthesis controller. Furthermore, one also has to analyze technical issues such as the amount of training data needed for reaching nearly asymptotic accuracy, the selection of most stable feature extraction scheme, dimensionality reduction method, classification algorithm combination, and incremental learning. In this work, we address a subset of these issues, in particular the amount of data required to reliably reach high accuracies and the fundamental question of accuracy degradation for an initially trained classifier. In the 121 benchmark we define three validation schemes. For a test trial i, $i \geq 2$, we configure the training set to consist of:

1) $1, \ldots, i-1$,
2) $1, \ldots, \min(s, i-1)$, and
3) $\max(i-s, 1), \ldots, i-1$

trials. Here, s denotes the number of trials sufficient for all algorithms to reach high accuracy. Over all classifiers, we found five trials to be sufficient. The first validation scheme determines the accuracy using all available data for training. It is *a priori* unclear whether this results in the best possible performance, since in general, aged data might lower the classification accuracy. Moreover, a permanently growing training data set also permanently increases the computational load for retraining. The goal of the second scheme is to check whether the accuracy degrades, when a classifier is trained with data from the first day only. Finally, the third validation scheme investigates the evolution of the accuracy when using only recent data for training and thus tries to answer the question: Can the classification accuracy be improved by stripping aged data?

Experiments using kNN, DT, MLP, and SVM algorithms are conducted with the data mining framework RapidMiner [87]. RapidMiner uses the LIBSVM [88] implementation for support vector machines and the WEKA [89] implementation for decision trees and multi-layer perceptrons. In the case of SVM, multi-class problems are handled by LIBSVM using the one-against-one method [90]. For the ECGP-based architecture, the experiments are carried out using the MOVES-toolbox [91].

B. *The* Day1–3 *Benchmark*

We use the error rate as a metric to compare the classification performances of different approaches. Tab. III summarizes the error rates, arranged by the particular individual. The test error rates show the classifiers' generalization abilities, and the error rates obtained for the training data sets point to the classifiers' approximation abilities. Bold numbers symbolize the best results. kNN, DT, MLP and SVM are the k-nearest-neighbor, decision tree, multi-layer perceptron and support vector machine approaches respectively. EHW1 and EHW2 are the evolvable hardware approaches, where EHW1 refers to the ECGP-based model (see Section V-A1) and EHW2 denotes the FU row architecture (see Section V-A2).

TABLE III
Day1–3 Experiment: Approximation and Generalization errors in %. Bold numbers represent the best error rates.

	user 1		user 2		user 3	
	training	test	training	test	training	test
kNN	3.14	**3.96**	12.94	**17.29**	3.64	**4.75**
DT	1.42	8.68	4.51	25.85	1.72	8.95
MLP	3.09	4.45	23.73	25.44	4.31	5.67
SVM	5.53	5.63	32.22	32.30	6.59	6.68
EHW1	7.50	8.86	38.00	39.57	8.81	8.30
EHW2	9.59	10.02	45.67	46.08	11.03	11.40

The first major observation is that we achieve high training and test error rates for user 2. Since we carefully configured and adjusted the EMG sensor positions and ran tests before starting the data experiments, we can dismiss the experimental setup as a reason for the high differences in the error rates. The day-wise analysis of user 2 data reveals similar bad recognition rates, separately for each day. Thus, we assume that the results are due to either a lax tension when performing the contraction phases or the physiological properties of the subject. The second observation, and this comes as a surprise, is the excellent performance of the kNN classifier. This could be explained by the "bias/variance dilemma", mentioned in Sec. V-B1. It states that even a simple classifier can achieve high accuracy rates, as low model complexity corresponds to low variance. MLPs come second, followed by SVMs. The small gaps between training and test accuracy rates imply correct parametrizations and negligible effects of overfitting. In contrast to this, and as a third observation, DTs show a larger distance between training and test accuracies despite using pruning techniques to prevent overfitting. The EHW-based classifiers close the comparison being near to DTs. The accuracy rates are within a 6% margin for user 1 and user 3 benchmarks and within a 29% margin for the user 2. High and compactly distributed accuracy rates for the user 1 and 2 experiments among all algorithms let us assume that the task of EMG signal classification using mean average features tends not to be too complex.

C. The 2of3 *Experiment*

In this experiment we focus on the real-world situation in which data of a past time period is used to train the classifiers and the performance of a prosthesis is measured on a consecutive time period. To this end, we define, as described in Subsection VI-A, a 3-fold cross validation scheme partitioning the data of the folds by the recording day. Since the EHW classifiers are evolved from random genotypes, each evolved classifier has a different structure and the classification rates vary slightly. The EHW experiments generate only three classifiers when computing the 3-fold cross validation. To achieve reliable accuracy rates, we evolved 10×3 classifiers and averaged the results.

TABLE IV
2of3 EXPERIMENT: APPROXIMATION AND GENERALIZATION ERRORS IN %. BOLD NUMBERS REPRESENT THE BEST ERROR RATES.

	2of3					
	user 1		user 2		user 3	
	training	test	training	test	training	test
kNN	2.70	12.38	12.62	**40.46**	3.02	18.25
DT	2.28	17.95	7.68	48.28	2.82	23.19
MLP	2.43	14.49	20.93	44.94	2.97	19.25
SVM	4.88	**12.10**	32.14	45,53	5.64	**17.04**
EHW1	2.90	19.63	6,88	48.43	1.49	18.05
EHW2	8.75	14.55	42.69	54.13	9.26	20.07

Characteristic of the *2of3* experiments are the higher error rates and larger distances between training and test rates compared to the *Day1–3* experiment (see Tab. IV). This can be explained by an insufficient amount of data for prediction model creation and by a smaller portion of training and larger portion of test data used in the 3-fold cross validation.

TABLE V
121 EXPERIMENT: AVERAGED ERRORS IN % (GENERALIZATION), WHEN TRAINED ON FIRST FIVE TRIALS (ROUGHLY DATA RECORDED ON A SINGLE DAY), FIVE RECENT TRIALS AND ON ALL PRECEDING TRIALS.

	first five	preceding five	all preceding
kNN	26.19	10.45	7.86
SVM	26.23	9.00	8.66

Additionally, data used for testing have been recorded on a different day than the data used for training. With these differences in mind, no algorithm consistently dominates this benchmark. While SVM and kNN still perform well, being among the three best algorithms and always taking the first place, MLPs are taking the second place for user 2 and EHW1 for user 3 experiments. Analogous to the *Day1–3* experiment, DTs again demonstrate some overfitting. Additionally, EHW1 shows larger overfitting effects indicating the necessity for pruning techniques. The ranges for test errors over all algorithms are roughly the same as in the previous benchmark, lying around 7% for the user 1 and user 3 experiments and around 14% for the user 2 experiment.

While observed performance behaviors in the *2of3* experiment can be explained by the same reasons mentioned when analyzing the *Day1–3* experiment, the results suggest two additional conclusions. The closer distances between SVMs, MLPs and the EHW approaches raise the question whether the higher decision boundary flexibility of SVMs and MLPs is really necessary. Additionally, the results support the idea that periodic retraining may be useful to maintain high classification rates.

D. The 121 *Experiment*

Fig. 12 and 13 plot the accuracy results for the best-performing conventional classifiers in our *121* experiment: kNN and SVM. As described in Subsection VI-A, we evaluate three validation schemes in this experiment: employing all preceding trials, preceding five trials, and first five trials for training and the following one trial for testing. For a trial i, while using all or five previous trials for training yields similarly good accuracy rates of roughly about 90%, training kNN and SVMs with the first five trials result in degrading accuracy. This accuracy loss became visible after two to three days of omitted re-training. The averaged error rates are summarized in Tab. V. Interestingly, there is no significant indication of a negative influence of "old" data used for training. For both algorithms, the error rates improve from the second to the third column of Tab. V. While the improvement is small for SVM, kNN improves its error rate by 2.59%. However, fading out old training data for potential accuracy improvement might make sense considering the real-world prosthesis usage periods. Large training data bases also result in high computational effort for SVM training and kNN classifying.

We limit the evaluation for the remaining classification algorithms to the "preceding five trials" scheme, as training became costly with increasing data sizes. We chose this scheme as it is relevant to the prosthesis controller application.

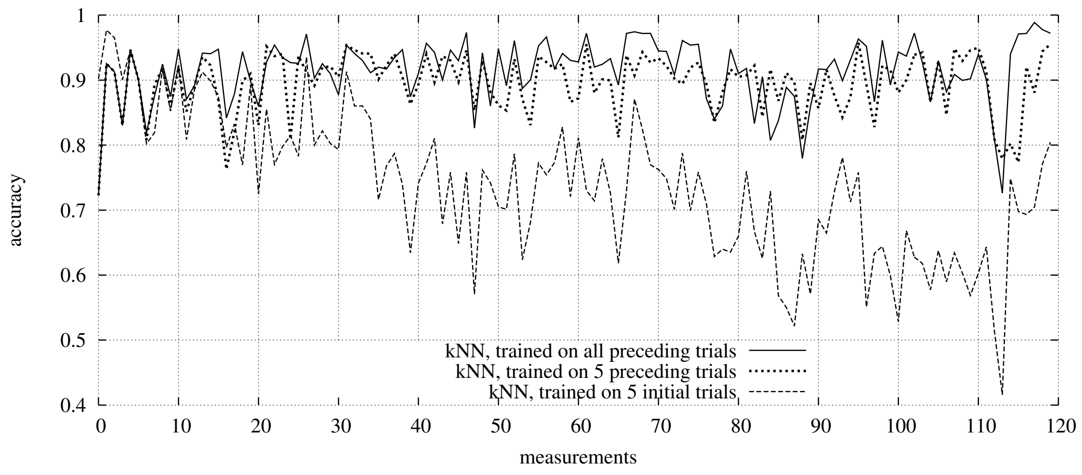

Fig. 12. 121 Experiment: Test accuracy for the *k*NN algorithm trained on the first five, last five, and all preceding trials.

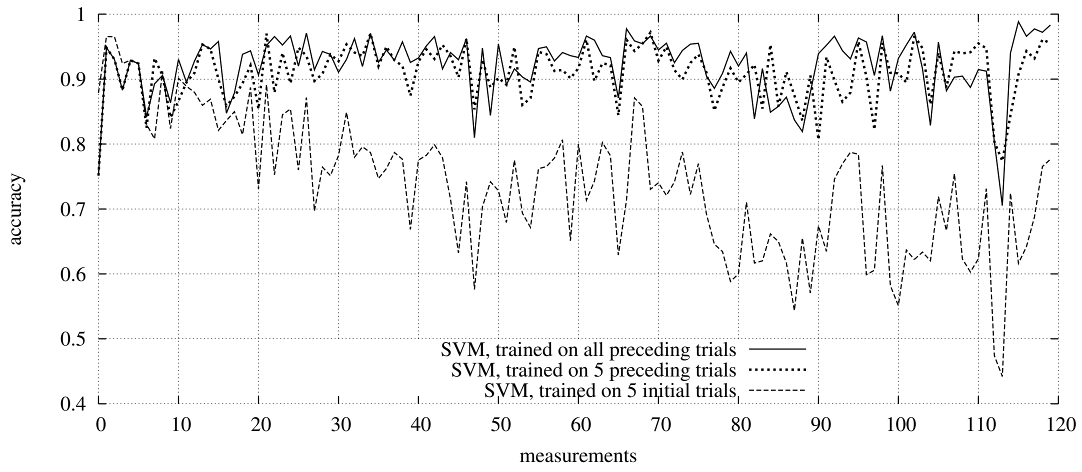

Fig. 13. 121 Experiment: Test accuracy for the SVM algorithm trained on the first five, last five an all preceding trials.

The results are summarized in Tab. VI. Similar to the previous experiments, *k*NN and SVMs perform best followed by MLPs. EHW approaches lie between this group and the DTs. The error rates spread in a 9% interval bounded by SVMs at 9.0% and DTs at 17.91%. The values are similar to the user 1 and 3 results of the *2of3* experiment. The peak classification rates are slightly better while being somewhat broadly distributed. The similarities of the *2of3* and *121* experiments are not surprising. Both evaluation schemes use data from disjoint recording sessions for training and testing. The slightly better results of the *121* experiment can be explained by a larger data portion used for training. This also is probably an explanation for the broader accuracy distribution. Data in the *121* experiment were recorded for user 1. All algorithms are able to improve their classification accuracies using more training data in the *121* experiment (compare Tab. IV, user 1 test column and Tab. VI). While for some algorithms the improvements are rather small (cf. DT with a 0.04%), other algorithms manage to get roughly 3% to 4% improvements, as the EHW1, SVMs and MLPs.

TABLE VI
121 EXPERIMENT: AVERAGED ERRORS IN % (GENERALIZATION), WHEN TRAINED ON FIVE RECENT TRIALS (ROUGHLY DATA RECORDED ON A SINGLE DAY). BOLD NUMBERS REPRESENT THE BEST ERROR RATE.

	error rate
*k*NN	10.45
DT	17.91
MLP	10.44
SVM	**9.00**
EHW1	16.48
EHW2	13.72

VII. DISCUSSIONS

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

- Among the conventional classifiers, *k*NN 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 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, kNN and SVM yield the best results. However, similar to kNN, SVM might be susceptible to a growing training data set.

- The EHW approaches—and this is the main result of our experiments—also yield a good classification performance. While the *Day1–3* experiment turned out to be tough for both approaches where EHW1 and EHW2 clearly rank at the end, the *2of3* experiment shows a more compact distribution of accuracies with the EHW approaches deviating 2%, 8%, and 1% from the best performing classifier. In the *121* experiment, the distances to the best performing algorithms amount to 4,7% and 7.5%.
- We observe that the task of EMG signal classification using mean average features needs online learning. The classification rates fall by 8%, 23% and 12% from the *Day1–3* experiment, where 10-fold validation was applied on the shuffled data set, to the *2of3* experiment, where data of two days was used to classify the data of the third day. This is an indication that data of few days is not sufficient to evolve a highly predictive model. In an additional experiment we observed that the classification rates degrade when not retraining continuously.

The achieved classification rates, except the results for the second user in the *2of3* experiment, are high enough for a prosthetic hand controller. One of the open issues in the area of prosthesis controllers is the definition of a proper quality metric. A PHC relies only partly on the accuracy of the utilized classifier. In recent work, a more holistic approach has been introduced by Englehart et al. [92] and Shenoy et al. [38], accounting for the success and execution time of complex and real-world hand movements. Typical exercises are, for instance, the grasping and turning of a door knob, and picking up and relocation of an object. These kind of metrics implicitly rate the classification accuracy. Summarizing his results, Englehart states that: “Perhaps most importantly, these results [92] support clinical observations that training data which includes transient MES [myoelectric signal] information can lead to more robust usability and performance while yielding a seemingly “worse” classifier. The authors [Englehart et al.] therefore suggest caution in accepting classification error as the sole measure of a systems usability and performance” [92]. Englehart et al. found that accuracy rates below 85% start to impact on the performance of complex movement executions.

The accuracy rates in our experiments can certainly be improved. The goal of our work is not to reach highest classification rates possible, but to have a fair comparison of pattern matching algorithms. To this end, we have spent roughly the same amount of time finding good performing configurations for the different classifiers. We used grid search for conventional and our expert knowledge for EHW algorithms. Additionally, the experiments were executed by subjects not familiar with EMG controlled prostheses. An amputee usually spends months of training before being able to reliably create pronounced muscular tensions. He/She also learns how and which muscles to activate to achieve the desired response from the prosthesis. Higher recognition rates can be approached by a model view on prosthesis’ mechanics. A prosthesis cannot respond to very short misclassifications of some milliseconds. Elimination of misclassification glitches by a low-pass filter, while slowing down prosthesis latency, increases the overall accuracy rate. In our experiments we observed improvements of 3% to 7%. Furthermore, complex hand movements decompose into basic hand actions. Each action has a typical duration and some follow-up actions, specified by their probabilities. Capturing prosthesis’ action space by a Markov-chain based model, helps reduce misclassification during a single action and select a correct follow-up action more quickly and reliably.

Incremental learning can help to reduce computational complexity for situations where online learning is required. For instance, kNNs need only add labeled data to their data base to build a new model. There are several elegant methods to update support vectors of a SVM (see, e.g., [9], [10]). The update process of a neural network is more complicated, often demanding a return of complete learning algorithm (see, e.g., [11]). However, porting software algorithms to hardware with an option for reconfiguration is not a trivial task. Our EHW approaches, while being reconfigurable, have not yet been investigated regarding efficient retraining techniques.

All of the classification approaches are able to provide a differentiated output of the certainty of the match to a given class (movement). This implies implementation details being key to classifier selection for prosthesis control. PHC’s primary requirements are a secure operation mode with guaranteed functional and temporal aspects, as well as energy efficiency. This gives HW approaches an advantage over SW methods for the following reasons: The exclusive usage of a computational resource often allows for giving HW approaches precise execution time estimations and response time guarantees. HW solutions also have a somewhat simpler function verification. While some modern microcontrollers have low energy consumption [93], they lack larger memories and floating point units. Together with limited computational power, often only simplified classification algorithms can be realized there. Evolving and retraining classifiers, storing larger data models and data sets, and acquiring dynamic data is seldom possible on such small systems. HW approaches, on the other hand, while also offering energy efficiency and compactness, are much more capable of solving computationally extensive tasks and can be designed to efficiently interface large memories.

Although it is hard to state precise comparisons between possible hardware implementations of the EHW approaches and the other classifiers in this article, we can make some general considerations. The kNN is data driven, a hardware implementation of this approach is therefore expensive in terms of distributed memory as the number of input vectors grow [94]. Our EHW approaches operate on a fixed number of CCs and thus the implementation and classification time would not grow with the number of training vectors. HW implementations of SVM face the challenge of a constrained quadratic programming problem for the training phase, which grows in complexity with the number of training vectors. A fully parallel approach is restricted by the reconfigurable device size, which also limits the training set [95]. The popular specialized sequential minimal optimization has recently been proposed for a hardware SVM system [96]. However, there are still important resource costs associated with fixed-point arithmetic operations. As our EHW approaches bear some similarities to DTs, a DT hardware implementation would be scalable in terms of the number of training vectors. However, generated DTs are often complex and care would have to be taken into generating hardware-friendly trees and avoiding floating point operations. The straightforward implementation of ANNs requires much resources because of heavy use of floating point arithmetic in the nodes. However, there are alternatives which are better suited for HW implementation.

In summary, we can conclude that the observed results give us rough classification tendencies for EMG signal classification. More representative results require experiments with a greater number of subjects. The classification rate in a real prosthesis controller application might be significantly better because of amputee training, personalization of algorithms' configurations, fewer number of classes, and more elaborate online learning.

VIII. Conclusion and Future Work

In this article, 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 results of this article are that EHW classifiers are at par with conventional techniques and that classification of EMG signals requires an adaptable classification architecture with an incremental learning approach. This finding is of utmost importance, as the appeal of EHW approaches is rooted in their suitability for self-adaptation, fast training, and compact system-on-chip implementation. Knowing that EHW approaches are also competitive in terms of classification performance motivates future work along several lines:

While the two EHW approaches both provide 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, uses more *a priori* knowledge in the form of predefined 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. For example, in [97] the authors demonstrated the FUR architecture to be robust to resource changes, showing fast recovery in case of architecture reconfiguration.

The modular and scalable structure of our approaches is well suited to run-time reconfiguration; incremental training and learning is thus a strong candidate for future research. For non-stationary data sets comprising variable components over time, computational complexity reduction of classifier update is the primary goal for incremental learning. Incremental learning for the ECGP-based architecture requires re-running the evolution with the updated data set on previously evolved classifier. FUR's incremental learning needs the architecture to "forget" some learned pattern matching functions. However, FUR's category classifiers are hierarchical superpositions of such functions. Changing or removing a single function may invalidate the complete hierarchy. Thus, FUR needs to be evaluated regarding new learning strategies.

Classification accuracy influences the acceptance of a EMG-driven prosthesis only to some extent. Englehart et al. [92] found that accuracies over roughly 85% can lead to "robust usability and performance", when considering real-world composite hand movements. Our results can be improved by a model-driven approach. To this end, upper limb action space needs to be captured by a model considering movements' durations and the likeliness of typical movement sequences. Another issue is the prosthesis mechanics latency. A prosthesis cannot react to classification impulses below some threshold. Filtering out these misclassifications also improves the accuracy. An amputee is constantly learning how to work with the prosthesis' classifier to get the right response. With amputee's visual feedback, a sufficiently good classifier may be the right choice for reliable and intuitive prosthesis control. A holistic approach considering more facets of prosthesis control and employing a model-based approach needs to be evaluated, to allow conclusive insights into this area of human interfaces.

The application of prosthesis control relies on temporal and functional security aspects. HW solutions exclusively using computational resources can often guarantee execution times. When compared to HW implementations of conventional SW classifiers, our EHW architectures have further advantageous elements: While compact and energy efficient, they are able to operate at very high frequencies and reuse the classification architecture for training. EHW architectures do not require complex functional units, like floating point ALUs, and have a plain structure. Being inherently reconfigurable and utilizing a simple local-search like $(1 + \lambda)$-ES, a complete run-time adaptable system can be implemented efficiently within a compact footprint.

Generally, we can state that the presented EHW architectures do not have as high peak accuracy rates as today's best pattern matching algorithms on EMG signal classification, but can compete with well know approaches such as decision trees. The observed classification accuracies are close to each other, suggesting that implementation details become dominant when selecting a classifier for a prosthetic control system. Additionally, the requirement of dynamic learning makes the run-time adaptable EHW system one of the most promising candidates for application in prosthesis control.

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 Design of Systems for Complex Real-World Applications* under project number 160308/V30, the German Academic Exchange Service / Research Council of Norway through the collaborative project *Autonomously Adaptable System-on-Chip* under project number D/06/12740 and by the European Union Seventh Framework Program under grant agreement 257906: "EPICS: Engineering Proprioception in Computing Systems".

REFERENCES

[1] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolving Hardware with Genetic Learning: a First Step Towards Building a Darwin Machine," in *From Animals to Animats*. MIT Press, 1993, pp. 417–424.

[2] H. de Garis, "Evolvable Hardware: Genetic Programming of a Darwin Machine," in *Intl. Conf. on Artificial Neural Nets and Genetic Algorithms*. Springer, 1993, pp. 441–449.

[3] M. Tanaka, H. Sakanashi, M. Salami, M. Iwata, T. Kurita, and T. Higuchi, "Data Compression for Digital Color Electrophotographic Printer With Evolvable Hardware," in *Intl. Conf. on Evolvable Systems (ICES)*, ser. LNCS. Springer, 1998, vol. 1478, pp. 106–114.

[4] J. Koza, M. Keane, and M. Streeter, "Routine High-Return Human-Competitive Evolvable Hardware," *NASA/DoD Conference on Evolvable Hardware*, pp. 3–17, June 2004.

[5] L. Sekanina, "Evolutionary Design Space Exploration for Median Circuits," in *Applications of Evolutionary Computing*, ser. LNCS, vol. 3005. Springer, 2004, pp. 240–249.

[6] J. Lohn, G. Hornby, and D. Linden, "Evolutionary Antenna Design for a NASA Spacecraft," in *Genetic Programming Theory and Practice II*. Springer, 13-15 May 2004, ch. 18, pp. 301–315.

[7] P. Kaufmann, C. Plessl, and M. Platzner, "EvoCaches: Application-specific Adaptation of Cache Mappings," in *Adaptive Hardware and Systems (AHS)*. IEEE CS, 2009, pp. 11–18.

[8] C. Giraud-Carrier, "A Note on the Utility of Incremental Learning," *AI Communications*, vol. 13, no. 4, pp. 215 – 223, 2000.

[9] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental Training of Support Vector Machines," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 114 – 131, 2005.

[10] P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, "Incremental Support Vector Learning: Analysis, Implementation and Applications," *Journal of Machine Learning Research*, vol. 7, pp. 1909 – 1936, 2006.

[11] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: an Incremental Learning Algorithm for Supervised Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, no. 4, pp. 497 – 508, 2001.

[12] K. Glette, T. Gruber, P. Kaufmann, J. Torresen, B. Sick, and M. Platzner, "Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control," in *Adaptive Hardware and Systems (AHS)*. IEEE, 2008, pp. 32–39.

[13] D. S. Childress and M. V. Podlusky, "Letter To The Editor," in *Medical and Biological Engineering and Computing*, vol. 7, no. 3. Springer, 1969, p. 345.

[14] R. Reiter, "Eine neue Elektrokunsthand," in *Grenzgebiete der Medizin*, vol. 4, 1948, pp. 133–135.

[15] D. S. Dorcas and R. N. Scott, "A Three-state Myoelectric Control," in *Medical & Biological Engineering*, vol. 4, 1966, pp. 367–370.

[16] D. A. Childress, "A Myoelectric Three State Controller Using Rate Sensitivity," in *8th ICMBE*, vol. S4–5, 1969.

[17] P. Herberts, "Myoelectric Signals in Control of Prostheses," in *Acta Orth. Scand.*, vol. 40, 1969, p. 124.

[18] D. S. Childress, "An Approach of Powered Grasp," in *4th Intl. Symposium on External Control of Human Extremities.*, 1973.

[19] J. Graupe and K. Cline, "Functional Separation of EMG Signals Via ARMA Identification Methods for Prosthesis Control Purposes," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 5. IEEE Press, 1975, pp. 252–259.

[20] A. E. Kobrinski, S. V. Bolkovitin, D. M. Voskoboinikova, L. M. Ioffe, E. P. Polyan, B. P. Popov, Y. L. Slavutski, Y. A. Sysin, and Y. S. Yakobson, "Problems of Bioelectric Control," in *Automatic and Remote Control*, vol. 2, 1960, pp. 619–629.

[21] R. R. Finley and R. W. Wirta, "Myocoder Studies of Multiple Myocoder Response," in *Arch Phys Med Rehabil*, vol. 48, 1967, p. 598.

[22] B. Hudgins, P. Parker, and R. Scott, "A New Strategy for Multifunction Myoelectric Control," in *Biomedical Engineering*, vol. 40(1). IEEE, 1993, pp. 82–94.

[23] K. Englehart, B. Hudgins, and P. A. Parker, "Time-Frequency Based Classification of the Myoelectric Signal: Static vs. Dynamic Contractions," in *Engineering in Medicine and Biology Society (EMBS)*. IEEE, 2000.

[24] M. Zardoshti-Kermani, B. Wheeler, K. Badie, and R. Hashemi, "EMG Feature Evaluation for Movement Control of Upper Extremityprostheses," in *IEEE Transactions on Rehabilitation Engineering*, vol. 3(4). IEEE Press, 1995, pp. 324–333.

[25] S. Park and S. P. Lee, "EMG Pattern Recognition Based on Artificial Intelligence Techniques," in *IEEE Transactions on Rehabilitation Engineering*, vol. 6. IEEE Press, 1998, pp. 400–405.

[26] A. Boschmann, P. Kaufmann, M. Platzner, and M. Winkler, "Towards Multi-movement Hand Prostheses: Combining Adaptive Classification with High Precision Sockets," in *Technically Assisted Rehabilitation (TAR)*, 2009.

[27] I. Kajitani, M. Murakawa, D. Nishikawa, H. Yokoi, N. Kajihara, M. Iwata, D. Keymeulen, H. Sakanashi, and T. Higuchi, "An Evolvable Hardware Chip for Prosthetic Hand Controller," *microneuro*, vol. 0, p. 179, 1999.

[28] K. A. Farry and I. D. Walker, "Myoelectric Teleoperation of a Complex Robotic Hand," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 3. IEEE, 1993, pp. 502–509.

[29] S. Karlsson, J. Yu, and M. Akay, "Time-frequency Analysis of Myoelectric Signals During Dynamic Contractions: a Comparative Study," in *Biomedical Engineering*, vol. 47(2). IEEE, 2000, pp. 228–238.

[30] P. Sparto, M. Parnianpour, E. Barria, and J. Jagadeesh, "Wavelet and Short-time Fourier Transform Analysis of Electromyography for Detection of Back Muscle Fatigue," in *Rehabilitation Engineering*, vol. 8(3). IEEE, Sep 2000, pp. 433–436.

[31] K. Englehart, B. Hudgins, P. A. Parker, and M. Stevenson, "Improving Myoelectric Signal Classification Using Wavelet Packets and Principal Components Analysis," in *21st Intl. Conf. of the IEEE Engineering in Medicine and Biology Society*. IEEE Press, October 1999.

[32] S. Micera, A. M. Sabatini, P. Dario, and B. Rossi, "A Hybrid Approach to EMG Pattern Analysis for Classification of Arm Movements Using Statistical and Fuzzy Techniques," in *Medical engineering & physics*, vol. 21(1350–4533). Butterworth-Heinemann, 1999, pp. 303–311.

[33] J.-U. Chu, I. Moon, Y.-J. Lee, S.-K. Kim, and M.-S. Mun, "A Supervised Feature-Projection-Based Real-Time EMG Pattern Recognition for Multifunction Myoelectric Hand Control," *Transactions on Mechatronics, IEEE/ASME*, vol. 12, no. 3, pp. 282–290, 2007.

[34] A. Hiraiwa, N. Uchida, N. Sonehara, and K. Shimohara, "EMG Pattern Rcognition by Neural Networks for Prosthetic Fingers Control - Cyber Finger," in *Measurement and control in Robotics*, 1992, pp. 535–542.

[35] D. Nishikawa, W. Yu, H. Yokoi, and Y. Kakazu, "EMG Prosthetic Hand Controller Discriminating Ten Motions Using Real-time Learning Method," in *Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 1999, pp. 1592–1597.

[36] H.-P. Huang, Y.-H. Liu, L.-W. Liu, and C.-S. Wong, "EMG Classification for Prehensile Postures Using Cascaded Architecture of Neural Networks With Self-organizing Maps," in *Robotics and Automation (ICRA)*, vol. 1. IEEE, Sept. 2003, pp. 1497–1502.

[37] S. Bitzer and P. van der Smagt, "Learning EMG Control of a Robotic Hand: Towards Active Prostheses," in *Intl. Conf. on Robotics and Automation*, 2006, pp. 2819–2823.

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

[39] K. Englehart and B. Hudgins, "A Robust, Real-time Control Scheme for Multifunction Myoelectric Control," in *IEEE Transactions on Biomedical Engineering*, vol. 50(7). IEEE Press, 2003, pp. 848–854.

[40] K. Englehart, B. Hudgins, and P. A. Parker, "A Wavelet Based Continuous Classification Scheme for Multifunction Myoelectric Control," in *Biomedical Engineering*, vol. 48(3). IEEE Press, 2001, pp. 302–331.

[41] B. Karlik, M. Osman Tokhi, and M. Alci, "A Fuzzy Clustering Neural Network Architecture for Multifunction Upper-limb Prosthesis," in *Biomedical Engineering*, vol. 50(11). IEEE, Nov. 2003, pp. 1255–1261.

[42] F. Chan, Y.-S. Yang, F. Lam, Y.-T. Zhang, and P. Parker, "Fuzzy EMG Classification for Prosthesis Control," in *Rehabilitation Engineering*, vol. 8(3). IEEE, Sep 2000, pp. 305–311.

[43] Y. Huang, K. Englehart, B. Hudgins, and A. Chan, "Optimized Gaussian Mixture Models for Upper limb Motion Classification," in *Engineering in Medicine and Biology Society (IMBS)*, vol. 1. IEEE Press, Sept. 2004, pp. 72–75.

[44] A. Chan and K. Englehart, "Continuous Myoelectric Control for Powered Prostheses Using Hidden Markov Models," in *Biomedical Engineering*, vol. 52(1). IEEE Press, 2005, pp. 121–124.

[45] M. Zecca, S. Micera, M. C. Carrozza, and P. Dario, "Control of Multifunctional Prosthetic Hands by Processing the Electromyographic Signal," in *Critical Reviews in Biomedical Engineering*, 2002, pp. 459–485.

[46] P. A. Parker, K. B. Englehart, and B. S. Hudgins, *Electromyography : Physiology, Engineering, and Non-Invasive Applications*, ser. IEEE Press Series on Biomedical Engineering. IEEE, 2004, ch. Control of Powered Upper Limb Prostheses.

[47] T. Higuchi, M. Iwata, I. Kajitani, H. Iba, Y. Hirao, B. Manderick, and T. Furuya, "Evolvable Hardware and its Applications to Pattern Recognition and Fault-Tolerant Systems," in *Towards Evolvable Hardware: The evolutionary Engineering Approach*, ser. LNCS. Springer, 1996, vol. 1062, pp. 118–135.

[48] 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 *Intl. Conf. on Evolvable Systems (ICES)*, ser. LNCS, vol. 1478. Springer, 1998, pp. 1–12.

[49] I. Kajitani, I. Sekita, N. Otsu, and T. Higuchi, "Improvements to the Action Decision Rate for a Multi-Function Prosthetic Hand," in *Measurement, Analysis and Modeling of Human Functions (ISHF)*, 2001, pp. 84–89.

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

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

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

[53] ——, "Incremental Evolution of a Signal Classification Hardware Architecture for Prosthetic Hand Control," *Int. J. Know.-Based Intell. Eng. Syst.*, vol. 12, pp. 187–199, 2008.

[54] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, "Hardware Evolution at Function Level," in *Parallel Problem Solving from Nature (PPSN)*, ser. LNCS, vol. 1141. Springer, 1996, pp. 62–71.

[55] M. Yasunaga, T. Nakamura, and I. Yoshihara, "Evolvable Sonar Spectrum Discrimination Chip Designed by Genetic Algorithm," in *Systems, Man and Cybernetics*, vol. 5. IEEE, 1999, pp. 585–590.

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

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

[58] A. Upegui and E. Sanchez, "Evolving Hardware by Dynamically Reconfiguring Xilinx FPGAs," *Evolvable Systems: From Biology to Hardware*, pp. 56–65, 2005.

[59] F. Cancare, M. Santambrogio, and D. Sciuto, "A Direct Bitstream Manipulation Approach for Virtex4-Based Evolvable Systems," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 853–856.

[60] G. Tufte and P. C. Haddow, "Biologically-Inspired: A Rule-Based Self-Reconfiguration of a Virtex Chip," in *Proc. of International Conference on Computational Science 2004*, ser. Lecture Notes in Computer Science, vol. 3038, May 2004, pp. 1249–1256.

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

[62] ——, "An Online EHW Pattern Recognition System Applied to Sonar Spectrum Classification," in *Intl. Conf. on Evolvable Systems (ICES)*, ser. LNCS. Springer, 2007, vol. 4684, pp. 1–12.

[63] ——, "Online Evolution for a High-Speed Image Recognition System Implemented On a Virtex-II Pro FPGA," in *Adaptive Hardware and Systems (AHS)*. IEEE, 2007, pp. 463–470.

[64] K. Glette, J. Torresen, and M. Hovin, "Intermediate Level FPGA Reconfiguration for an Online EHW Pattern Recognition System," in *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on*. IEEE, 2009, pp. 19–26.

[65] K. Glette, J. Torresen, P. Kaufmann, and M. Platzner, "A Comparison of Evolvable Hardware Architectures for Classification Tasks," in *Intl. Conf. on Evolvable Systems (ICES)*, ser. LNCS, vol. 5216. Springer, 2008., pp. 22–33.

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

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

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

[69] MindMedia, "Nexus 10," `www.mindmedia.nl`.

[70] H. Gray, "Anatomy of the Human Body," *Wikimedia Commons*, 1918.

[71] I. Kajitani, T. Hoshino, M. Iwata, and T. Higuchi, "Variable Length Chromosome GA for Evolvable Hardware," in *Intl. Conf. on Evolutionary Computation (ICEC)*. IEEE, 1996, pp. 443–447.

[72] J. F. Miller, P. Thomson, and T. Fogarty, "Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study," in *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*. John Wiley and Sons, 1998, pp. 105–131.

[73] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[74] J. A. Walker and J. F. Miller, "Evolution and Acquisition of Modules in Cartesian Genetic Programming," in *European Conf. on Genetic Programming (EuroGP)*, ser. LNCS, vol. 3003. Springer, 2004, pp. 187–197.

[75] P. Kaufmann and M. Platzner, "Advanced Techniques for the Creation and Propagation of Modules in Cartesian Genetic Programming," in *Genetic and Evolutionary Computation (GECCO)*. ACM Press, 2008, pp. 1219 – 1226.

[76] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison–Wesley, 1989.

[77] T. K. Ho, "Random Decision Forests," in *Intl. Conf. on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 1995, p. 278.

[78] D. Fisch, B. Kühbeck, S. J. Ovaska, and B. Sick, "So Near and Yet So Far: New Insight Into Properties of Some Well-Known Classifier Paradigms," *Information Sciences*, vol. 180, no. 18, pp. 3381–3401, 2010.

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

[80] J. H. Friedman, "On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 55–77, 1997.

[81] P. Domingos, "A Unified Bias-Variance Decomposition for Zero-One and Squared Loss," in *Artificial Intelligence and Innovative Applications of Artificial Intelligence*, 2000, pp. 564–569.

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

[83] 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.

[84] L. Hamel, *Knowledge Discovery With Support Vector Machines*. John Wiley & Sons, 2009.

[85] C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[86] R. Rojas, *Neural Networks – A Systematic Introduction*. Springer, 1996.

[87] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "YALE: Rapid Prototyping for Complex Data Mining Tasks," in *Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 935 – 940.

[88] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines*, 2001.

[89] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," in *SIGKDD Explor. Newsl.*, vol. 11(1). ACM, 2009, pp. 10–18.

[90] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multi-class Support Vector Machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[91] P. Kaufmann and M. Platzner, "MOVES: A Modular Framework for Hardware Evolution," in *Adaptive Hardware and Systems (AHS)*. IEEE, 2007, pp. 447–454.

[92] L. Hargrove, Y. Losier, B. Lock, K. Englehart, and B. Hudgins, "A Real-Time Pattern Recognition Based Myoelectric Control Usability Study Implemented in a Virtual Environment," in *Engineering in Medicine and Biology Society (EMBS)*. IEEE, 2007, pp. 4842–4845.

[93] Texas Instruments Inc., *16-Bit Ultra-Low Power MSP430 Microcontrollers*, Texas Instruments, Dallas, Texas, USA, 2011.
[94] E. Manolakos and I. Stamoulias, "IP-cores Design for the kNN Classifier," in *Proc. Intl. Sym. Circuits and Systems (ISCAS)*. IEEE, 2010, pp. 4133 –4136.
[95] D. Anguita, A. Boni, and S. Ridella, "A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation," *Transacrions on Neural Networks*, vol. 14, no. 5, pp. 993–1009, 2003.
[96] K. Cao, H. Shen, and H. Chen, "A Parallel and Scalable Digital Architecture for Ttraining Support Vector Machines," *J Zhejiang Univ-Sci C (Comput & Electron)*, vol. 11, no. 8, pp. 620–628, 2010.
[97] T. Knieper, P. Kaufmann, K. Glette, M. Platzner, and J. Torresen, "Coping with Resource Fluctuations: The Run-time Reconfigurable Functional Unit Row Classifier Architecture," in *Intl. Conf. on Evolvable Systems (ICES)*, ser. LNCS, vol. 6274. Springer, 2010, pp. 250–261.

Paul Kaufmann received diplomas in computer science and mathematics from the University of Paderborn, Germany, in 2005. Currently, he is working towards a Ph.D. degree in computer science at the Faculty for Electrical Engineering, Computer Science and Mathematics of the University of Paderborn. There, he is conducting research and development in the areas of automatic digital circuit design, evolutionary multi-objective optimization and optimization of adaptable systems with applications in processor design, adaptable controller evolution and recognition of electromyographic signals.

Kyrre Glette received his M.Sc. in Computer Engineering (2004) from the Norwegian University of Science and Technology, Norway, and his Ph.D. in Evolvable Hardware (2008) from the University of Oslo, Norway, with stays at the French Space Agency (CNES) in Toulouse, France, and the University of Tsukuba in Japan. He is currently employed at the University of Oslo as a Postdoctoral Research Fellow. His research interests are artificial intelligence and biologically-inspired systems, with a special focus on embedded and runtime evolvable hardware systems. A second research interest is evolutionary robotics with an emphasis on design and prototyping of biologically inspired robots.

Thiemo Gruber received a diploma in computer science from the University of Passau, Germany, in 2007. Currently, he is working towards a Ph.D. degree in computer science at the Intelligent Embedded Systems Lab of the University of Kassel, Germany. There, he is conducting research and development in the areas of machine learning (in particular, real-time time series analysis) with applications in gesture spotting and activity recognition, on-line signature verification and identification as well as medical applications of biometric writing systems.

Marco Platzner is Professor for Computer Engineering at the University of Paderborn. Previously, he held research positions at the Computer Engineering and Networks Lab at ETH Zurich, Switzerland, the Computer Systems Lab at Stanford University, USA, the GMD - Research Center for Information Technology (now Fraunhofer IAIS) in Sankt Augustin, Germany, and the Graz University of Technology, Austria. Marco Platzner holds diploma and PhD degrees in Telematics (Graz University of Technology, 1991 and 1996), and a "Habilitation" degree for the area hardware-software codesign (ETH Zurich, 2002). His research interests include reconfigurable computing, hardware-software codesign, and parallel architectures. He is a senior member of the IEEE, a member of the ACM, serves on the program committees of several international conferences (eg. FPL, FPT, RAW, ERSA, DATE), and is an associate editor of the International Journal of Reconfigurable Computing, the EURASIP Journal on Embedded Systems, and the Journal of Electrical and Computer Engineering. Marco Platzner is member of the board of the Paderborn Center for Parallel Computing and served on the board of the Advanced System Engineering Center of the University of Paderborn. He is faculty member of the International Graduate School Dynamic Intelligent Systems of the University of Paderborn, and of the Advanced Learning and Research Institute (ALaRI) at Universita' della Svizzera Italiana (USI), in Lugano.

Jim Torresen Jim Torresen received his M.Sc. and Dr.ing. (Ph.D) degrees in computer architecture and design from the Norwegian University of Science and Technology, University of Trondheim in 1991 and 1996, respectively. He has been employed as a senior hardware designer at NERA Telecommunications (1996-1998) and at Navia Aviation (1998-1999). Since 1999, he has been a professor at the Department of Informatics at the University of Oslo (associate professor 1999-2005). Jim Torresen has been a visiting researcher at Kyoto University, Japan for one year (1993-1994) and four months at Electrotechnical laboratory,Tsukuba, Japan (1997 and 2000) and one year at Cornell University (2010-2011).

His research interests at the moment include reconfigurable hardware, machine learning, bio-inspired computing, robotics and applying this to complex real-world applications. Several novel methods have been proposed. He has published a number of scientific papers in international journals, books and conference proceedings. 10 tutorials and several invited talks have been given at international conferences. He is in the program committee of more than ten different international conferences as well as a regular reviewer of a number of international journals (mainly published by IEEE and IET). He has also acted as an evaluator for proposals in EU FP7. A list and collection of publications can be found at the following URL: http://www.ifi.uio.no/~jimtoer/papers.html

Bernhard Sick received a diploma (1992), a Ph.D. degree (1999), and a "Habilitation" degree (2004), all in computer science, from the University of Passau, Germany. Currently, he is professor for Intelligent Embedded Systems at the Faculty for Electrical Engineering and Computer Science of the University of Kassel, Germany. There, he is conducting research and development in the areas of theory and application of soft-computing techniques, organic computing, collaborative data mining, intrusion detection, and biometrics. Bernhard Sick authored more than 80 peer-reviewed publications in these areas. He is a member of IEEE (Systems, Man, and Cybernetics Society, Computer Society, and Computational Intelligence Society) and GI (Gesellschaft fuer Informatik). Bernhard Sick is associate editor of the IEEE Transactions on Systems, Man, and Cybernetics – Part B; he holds one patent and received several thesis, best paper, teaching, and inventor awards.