Investigating Evolvable Hardware Classification for the BioSleeve Electromyographic Interface

Kyrre Glette
Department of Informatics
University of Oslo
Oslo, Norway
Email: kyrrehg@ifi.uio.no

Paul Kaufmann
Faculty of EE and CS
University of Kassel
Kassel, Germany
Email: paul.kaufmann@gmail.com

Christopher Assad, Michael T. Wolf
Jet Propulsion Laboratory
California Insitute of Technology
Pasadena, CA, USA
Email: {chris.assad,wolf}@jpl.nasa.gov

***Abstract*—We investigate the applicability of an evolvable hardware classifier architecture for electromyography (EMG) data from the BioSleeve wearable human-machine interface, with the goal of having embedded training and classification. We investigate classification accuracy for datasets with 17 and 11 gestures and compare to results of Support Vector Machines (SVM) and Random Forest classifiers. Classification accuracies are 91.5% for 17 gestures and 94.4% for 11 gestures. Initial results for a field programmable array (FPGA) implementation of the classifier architecture are reported, showing that the classifier architecture fits in a Xilinx XC6SLX45 FPGA. We also investigate a bagging-inspired approach for training the individual components of the classifier with a subset of the full training data. While showing some improvement in classification accuracy, it also proves useful for reducing the number of training instances and thus reducing the training time for the classifier.**

I. Introduction and Related Work

The performances of embedded systems are continuously rising. Despite the manufacturing process limitations for high-performance computing devices, embedded systems steadily become lighter, more energy efficient and capable of solving more complex tasks faster. Applications such as monitoring devices that can be worn permanently by living beings and systems that are able to operate autonomously for decades are just two examples. Especially the higher processing power opens new application areas so far only accessible to full-fledged computers. In our work, we take a look at one of such areas: human-machine-interface devices for processing of recorded skin surface muscular signals. We will show that an embedded hardware system is able to process a multi-channel signal and recognize upper-limb movements accurately.

For our work we are using the BioSleeve system developed at the Jet Propulsion Laboratory [1] for interfacing upper-limb muscular signals. The BioSleeve is equipped with a high number of electromyographic (EMG) channels, with more planned in future versions. This allows not only to reconstruct simple actions of the monitored extremity, but to extract more features for complex tasks such as robotic actuator and instruments control and steering of vehicles [2]. While the current BioSleeve prototype transmits raw sensor data over a tether to an off-board computer for processing, the long-term goal is to design a wireless, fully enclosed BioSleeve system. Here, the gesture recognition will be implemented with in-sleeve processing, to minimize the bandwidth required to communicate gestures to a remote receiver. To ensure long battery autonomy and robustness to changes in physical conditions such as electrode slippage and user fatigue, low power processing and adaptable algorithms should be considered. In this context evolvable hardware (EHW) is an interesting approach, providing possibilities for self-adaptation, fast training, and compactness [3]. While software-based Support Vector Machine (SVM) classification was considered in [1], in this paper we adapt and investigate the performance of a previously proposed EHW classification system [4] on data from the BioSleeve, explore a method for reducing the training time, and consider implementation tradeoffs. We also compare the EHW approach with results from conventional classifier paradigms.

Most previous explorations in decoding gestures from EMG signals have been for the control of prostheses, typically in decoding activity from a few residual muscles in the arm to control a low-DOF gripper or wrist. Recent research (see review in [5]) is focused on higher-DOF pattern recognition that has potential to control more sophisticated prosthetic arms that better approximate the human arm. In short term lab tests, greater than 90% classification accuracy on up to 12 static hand gestures is reported [6], [7]. For the control of dexterous actions, forearm EMG has been shown to provide accurate representations of finger movements and forces [8]. Hand and individual finger tracking from a small forearm EMG array were previously demonstrated at NASA Ames for virtual keypads and joysticks [9]. The BioSleeve system builds upon these prior works, a larger number of gestures has been decoded at a very high accuracy [1].

An early use of EHW for pattern recognition was reported in [10]. Their architecture was originally applied to character classification but was later used for classification of EMG signals in a prosthetic hand controller [11]. It employed a programmable logic array (PLA)-like structure, and the configuration was evolved using an on-chip genetic algorithm (GA) resulting in a compact and adaptable system. The classifier distinguished between six different kinds of movements, with accuracies and training times competitive to artificial neural networks (ANNs). Using similar EMG data, experiments on incremental evolution using a EHW architecture were conducted in [12], using a two-layered architecture. 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 ANNs. EHW classification architectures applied to domains other than EMG include, for example, the function level evolution of [13]. 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 [14]. 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.

Using the virtual reconfigurable circuit (VRC) [15] technique, we proposed an online evolvable EHW architecture, the Functional Unit Row (FUR) architecture, for classification tasks in [16]. 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. The architecture and another approach based on embedded cartesian genetic programming (ECGP) were applied to EMG signal classification for 8 classes in [17] and up to 11 classes in [18], with promising results. The dataset used in the following experiments contains a higher number of gestures as well as a higher number of input channels than previously studied with EHW approaches. A comparison of different EHW approaches to EMG signal classification was performed in [19]. The results indicated better classification accuracies for the FUR and ECGP approaches. Also, the FUR approach showed a convergence in classification accuracy with a significantly lower number of evaluations than the other approaches. This is related to the ensemble-like incremental training approach where evolution is performed on a relatively short genome, resulting in a small number of evaluations required to find a good solution. Inspired by the *bagging* method for improving stability and classification accuracy in conventional ensemble classifier paradigms [20], in this paper we investigate the effects of applying the method to the FUR classifier.

The remainder of the paper is structured as follows: Sec. II describes the BioSleeve hardware and the data collection procedure. Then, Sec. III introduces the FUR EHW classification architecture and the evolutionary training process. Sec. IV documents the experiments performed and presents the obtained results, which are then discussed in Sec. V. Finally, conclusions and directions for future work are given in Sec. VI.

II. The BioSleeve EMG System

The BioSleeve uses surface electromyography (EMG) to detect the activation of muscles in the forearm. Electrodes are placed on the surface of the skin with two poles placed along the length of a muscle fiber. The bioelectrical signals which activate the muscle produce a voltage across the poles, which is then amplified and recorded by the sensing circuitry.

A. BioSleeve Hardware

The BioSleeve V2 prototype, which is considered in this paper, is shown in Fig. 1. A detailed description of the

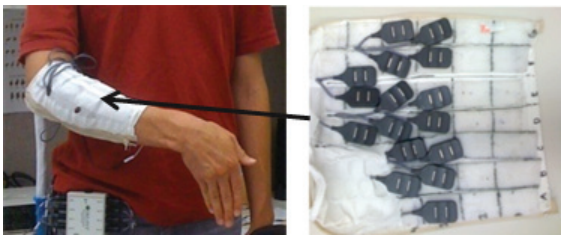


Fig. 1. The JPL BioSleeve prototype. The EMG sensors are shown on the right. Note that there is no need for precision alignment of sensors.

BioSleeve prototypes can be found in [1]. It incorporates 16 commercial dry-contact surface EMG sensors, which have active bipolar channels, band-pass filtered from 20 to 450 Hz. Dry contact electrodes are used since the ultimate goal is to have a quick "slip on" donning of the BioSleeve. The electrode sensors are mounted via Velcro fasteners on the inside of a modified compression sleeve to provide constant mechanical pressure on the users arm and maintain good skin contact. The sensor array is positioned on the proximal forearm near the elbow, to monitor activation of the major forearm muscles. These EMG signals contain a combination of force and position information for the wrist, hand and fingers [8]. As it can be observed in Fig. 1 the electrodes are not fixed in a regularly spaced pattern, the rationale is that by using a large enough number of electrodes, machine learning procedures will find the relevant electrodes and adapt to the current user's fitting of the sleeve.

B. Data Acquisition and Gesture Set

For the classification experiments in this paper, we use BioSleeve gesture data from recording sessions reported in [1]. Here, the raw EMG signal is recorded at 1000 Hz, and the signal is analyzed in 500-ms time windows at a rate of 10 Hz (i.e. there is an overlap of 400 ms between adjacent time windows). Within the kth time window, the feature vector $\mathbf{x}^k = (\sigma_1^k, ..., \sigma_d^k)^T$ is extracted, where σ_j^k is the standard deviation of the signal for the jth electrode, $j = 1, ..., d$:

$$\sigma_j^k = \left[\frac{1}{n-1} \sum_{i=1}^{n} \left(y_{j,i}^k - \bar{y}_j^k \right)^2 \right]^{\frac{1}{2}}, \qquad (1)$$

where $y_{j,i}^k$ is the ith voltage sample on the jth channel in the kth time window, $\bar{y}_j^k$ is the mean value of the $\{y_{j,i}^k\}_i$ samples, and n is the number of samples in the window ($n = 500$ for a 500-ms window at 1000 Hz). Preliminary experiments comparing this feature extraction method compared to more advanced features such as the number of zero crossings, slope sign change, and waveform length, indicate that recognition performance is not significantly affected by the choice of method [1].

For the data collection, the subject donned the BioSleeve without paying particular attention to precise electrode placement. The subject was prompted to make a particular static gesture and hold it for a period time, before relaxing and then commencing the recording of a new gesture, repeated for the number of specified gestures. The data used in this paper correspond to the data collected from subject B in [1]. Here, an equal number of training and test data were recorded in two

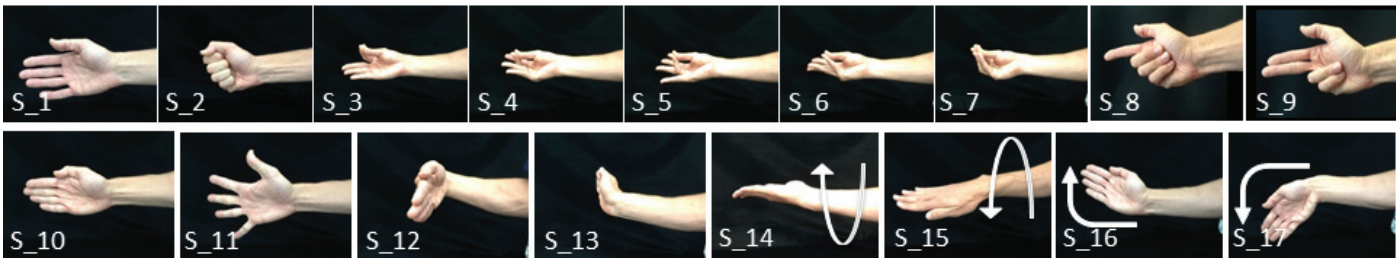


Fig. 2. Set of 17 hand gestures.

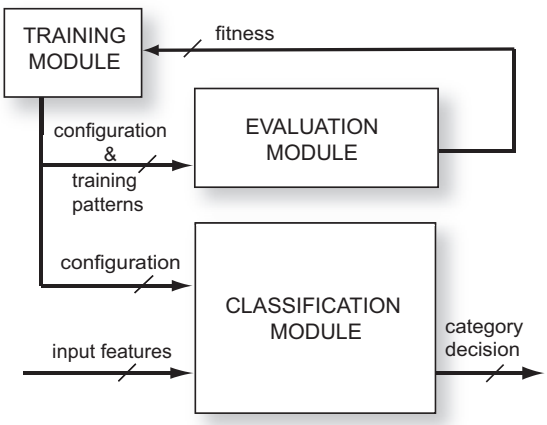


Fig. 3. A high-level overview of the on-chip EHW classification architecture

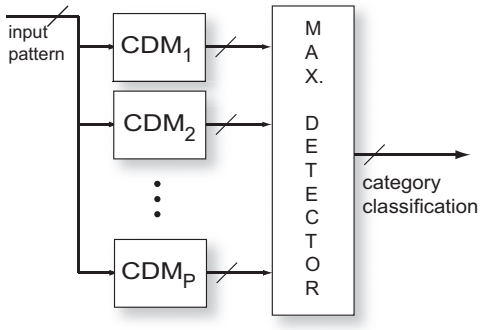


Fig. 4. Classification module.

separate sessions, without removal of the BioSleeve, about 10 minutes apart. A total of 17 gestures were recorded, as shown in Fig. 2. A subset of these gestures was also defined, as a preliminary attempt to have a gesture set which gives higher classification reliability. This subset leaves out gestures S_3 through S_7 and S_13 from the former set, giving a total of 11 gestures. We will hereafter refer to these gesture sets as B_17 and B_11. With the method described above, 1615 and 1045 vectors of 16 features were extracted for the training set for the B_17 and B_11 sets, respectively, and an equal number of vectors for the test sets. For the experiments in this paper, the data was normalized feature-wise to the range $[0..1]$ on the combined training and test set.

III. The FUR online EHW architecture

The FUR EHW architecture, proposed in [4], is designed as a circuit whose behavior can be modified through online reconfiguration. By writing the genome bitstring from the evolutionary algorithm (EA) to the internal configuration lines, one obtains the phenotype circuit which can then be evaluated.

A. System Overview

A high-level view of the proposed classification system can be seen in Figure 3. The training module incorporates an evolutionary algorithm which configures the classification sub-circuit in the evaluation module with a bitstring genome. After configuration, training data is applied to the circuit under evaluation, and a fitness value based upon the correctness of the output can be returned to the training module for use in the evolutionary process. When enough sub-circuits have been trained in the evaluation module, the classification module can be configured with a configuration bitstring. After initial feature extraction, input features will be fed to the classification module for classification, and the result will need to be processed depending on the specific control application. The initial feature extraction and classifier output processing implementations will not be considered in this work.

In the following text a conceptual description of the classification module will be given. For further FPGA implementation details, an implementation of the training and evaluation modules are described in [21]. Here, the training module was implemented on an on-chip CPU, but the simplified evolutionary process considered in this paper would open up for a full hardware implementation, similar to the one reported in [22]. Implementation details of the reconfigurable functional units can be found in [23]. The implementation of the feature extraction module has not been considered in this paper.

B. Architectural Elements

The classifier system consists of K category detection modules (CDMs), one for each category C_i to be classified – see Figure 4. The input data to be classified is presented to each CDM concurrently on a common input bus. The CDM with the highest output value will be detected by a maximum detector, and the identifying number of this category will be output from the system. Alternatively, the system could also state the degree of certainty of a certain category by taking the output of the corresponding CDM and dividing by the maximum possible output.

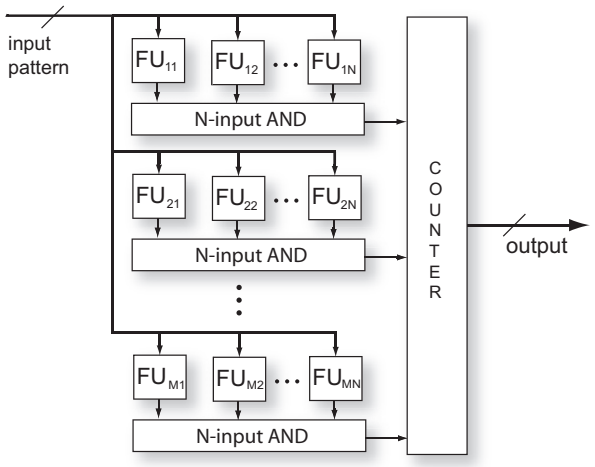


Fig. 5. Category detection module.

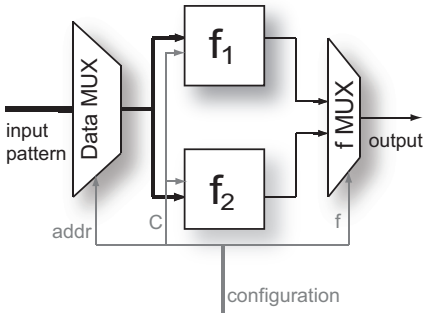


Fig. 6. High-level representation of a functional unit.

Each CDM consists of M "rules" or functional unit (FU) rows – see Figure 5. Each FU row consists of N FUs. The inputs to the circuit are passed on to the inputs of each FU. The 1-bit outputs from the FUs in a row are fed into an N-input AND gate. This means that all outputs from the FUs must be 1 in order for a rule to be activated. The 1-bit outputs from the AND gates are connected to an input counter which counts the number of activated FU rows. As the number of FU rows is increased, so is the output resolution from each CDM. Each FU row is evolved from an initial random bitstring, which ensures a variation in the evolved FU rows.

The FUs are the reconfigurable elements of the architecture. As seen in Figure 6, 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 element, I, is then fed to the available functions. Any number and type of functions could be imagined, but for clarity, in Figure 6 only two functions are illustrated. 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, O, from the unit.

C. Architecture Parameters

Through experimentation, the following functions have shown to work well for classification tasks [4], [17]:

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

These functions allow for the creation of a combination of planar decision boundaries perpendicular to the axes of the selected features for each FU row. Employing multiple FU rows per CDM, the architecture can be compared to ensemble-based classifier methods, such as e.g. Random Forests [24]. For the experiments in the paper, the resolution of the single features and the constant values are 8 bits. The number of features per data vector is equivalent to the number of EMG channels, 16 in this case. The number of FUs per row, N, has been set to 6 through experimentation. Each FU is encoded in the genome string with 4, 1, and 8 bits for the *feature address, function type,* and *constant*, respectively. This gives a total of $B_{unit} = 13$ bits for each unit. With $N = 6$, the total number of configuration bits for one FU row then is $B_{tot} = B_{unit} \cdot N = 78$.

D. Evolutionary Training

While the architecture was earlier trained using a genetic algorithm [4], the evolutionary training process now employs a 1 + 4 evolution strategy (ES) scheme, as recent experiments indicate good results with this more hardware-friendly approach [25]. In this approach a parent spawns 4 mutated offspring, and the best fit offspring will replace the parent if it has equal to or better fitness than the parent. A mutation rate of 3 randomly located bit flips per genome has been used in this work. The evolution follows an incremental fashion, in which each FU row is evolved separately before combined with the others to form a full classifier. This gives a genome size of 78 bits, equal to the configuration string length of an FU row.

Each row is evolved with the training vectors ($v \in V_t$), and fitness is based on the row's ability to give a positive (1) output for vectors v belonging to its own category ($C_v = C_p$), while giving a negative (0) output for the rest of the vectors ($C_v \neq C_p$). In the case of a positive output when $C_v = C_p$, the value A is added to the fitness sum. When $C_v \neq C_p$ and the row gives a negative output (value 0), 1 is added to the fitness sum. The other cases do not contribute to the fitness value. The fitness function F for a row can then be expressed in the following way, where o is the output of the FU row:

$$F = \sum_{v \in V_t} x_v \qquad \text{where } x_v = \begin{cases} A \cdot o & \text{if } C_v = C_p \\ 1 - o & \text{if } C_v \neq C_p. \end{cases} \tag{2}$$

The value A is a means to direct the evolutionary search towards making a row trigger on its designated class, and is set to 16 in the following experiments.

As an alternative to training each row with all training vectors, we propose to train the row using only a subset of the training vectors, similar to the *bagging* technique [20]. Here, the training vector subset of each row V_b is generated by sampling uniformly with replacement from the full training set V_t. The number of samples is defined as a fraction $0 < b \leq 1$

of the total number of training vectors. A new bagged subset is used for each FU row, and fitness is measured in the same way as described above while using training vectors from V_b instead of V_t.

IV. Experiments and Results

This section describes the experiments and presents results obtained.

A. Training on full dataset

We first investigate how the classification accuracy of the architecture varies with the number of FU rows per category, in order to investigate the tradeoff between classification accuracy and implementation cost. Here, we train each row on the full training set and use the test set for measuring the classification accuracy of the full classifier. Accuracy rates are defined as the number of samples correctly classified divided by the total number of samples. We scale the values in the training and test set to integer values in the range $[0..255]$, given by 8 bit precision. FU rows were trained for a maximum of 800 generations for the B_17 set and 400 generations for the B_11 set, but evolution was terminated earlier if a maximal fitness value was attained. The results can be seen in Fig. 7. Since each trained classifier is different because of the evolutionary search, we show the average and standard deviation from 50 training runs. The results were obtained using a software simulation of the architecture.

B. Training on Subsets

In order to investigate the effects of bagged training subsets, we extended the experiments from Sec. IV-A with training new classifiers with different subset sizes. The subset fraction b was chosen in the range of 0.1 to 1.0. The results, averaged over 50 runs, can be seen in Fig. 8. Next we investigate the effect of uniform sampling per row as opposed to training all classifier rows on the same static subset. Some selected subset fractions b were chosen for the bagged procedure and compared with results for equivalent static subsets, as shown in Fig. 9. To have an impression of the impact of bagging on the training time we measured the training time of a whole classifier in the software simulator, using different training subset sizes. We show the results for the two datasets on two different generation limits in Fig. 10.

C. Classification Accuracy Comparison

To evaluate the classification accuracy of the proposed solution we compare the FUR results, using $M = 32$ rows and $b = 0.2$, with some conventional classification approaches: SVM [26] and Random Forests [24].

We chose SVM for comparison as it employs the principle of structural risk minimization which typically yields very good generalization performance compared to other classifier paradigms. The SVM approach uses the same parameters as in previous experiments on BioSleeve data [1]: A radial basis function (RBF) kernel, $C = 128$, $\gamma = 0.5$, and the one-vs-one approach for multiclass decision. The LIBSVM implementation was used for obtaining the results [27].

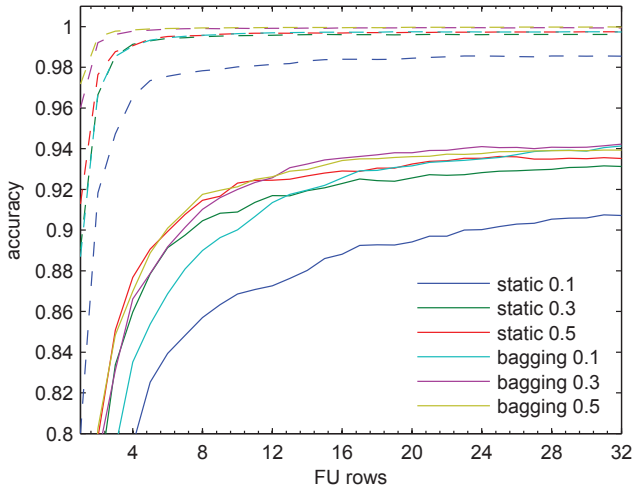


Fig. 9. Effect of training with a bagged subset per row versus a static subset on classification performance, with varying subset fractions b. Solid lines show test accuracy, dashed lines show training accuracy. B_11 dataset, average of 50 runs.

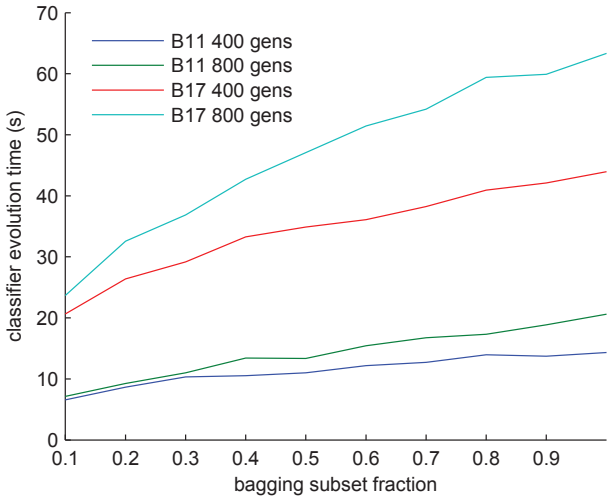
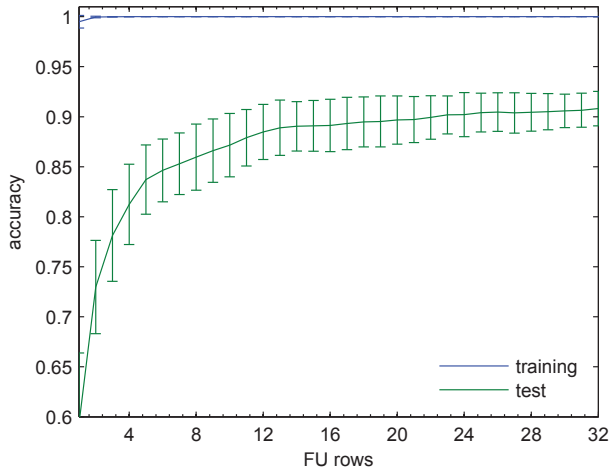


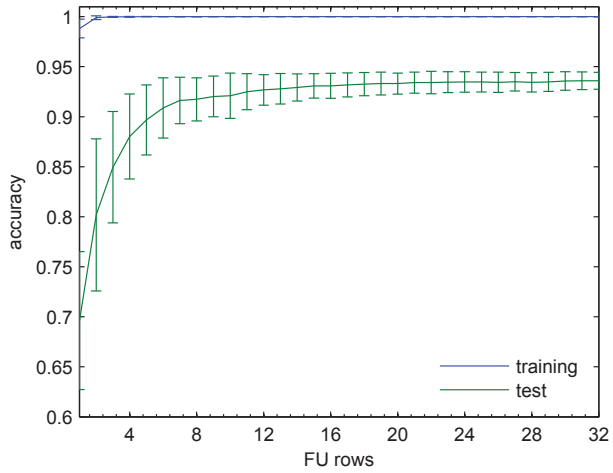
Fig. 10. Training times per classifier in the software simulator as a function of the size of the training subset. Average over 4 runs.

The Random Forests classification paradigm was chosen as it is a popular ensemble-based classifier which has some similarities to the FUR approach – training several simple classifiers on a subset of the features and the training set. The Weka [28] Random Forests implementation was used for obtaining the results. The default values of unlimited tree depth, and int($\log_2 F$ + 1) features to consider (where F is the number of features) per tree, were chosen. 50 trees were chosen to constitute the classifier.

All classifiers were trained with the training set and classification accuracy was evaluated on the test set. For the conventional classifiers we used the RapidMiner software [29] to set up the experiments. The results can be seen in Tables I and II. We also include results for the accuracy of the trained classifier on the training set. In addition, we show the accuracy

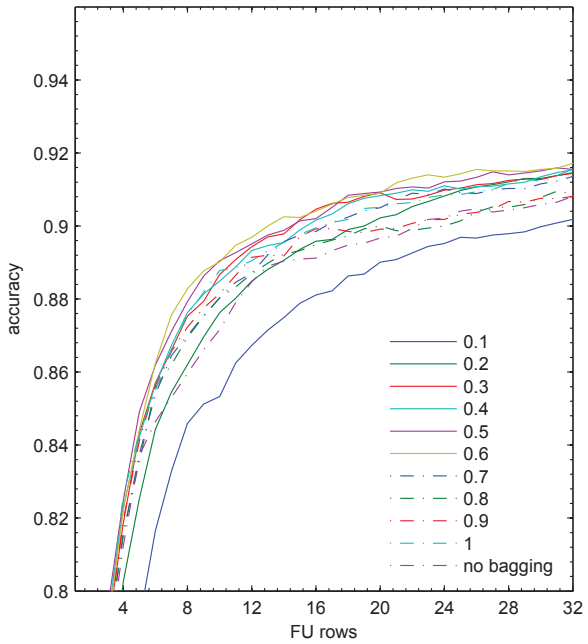


(a) B_17 dataset

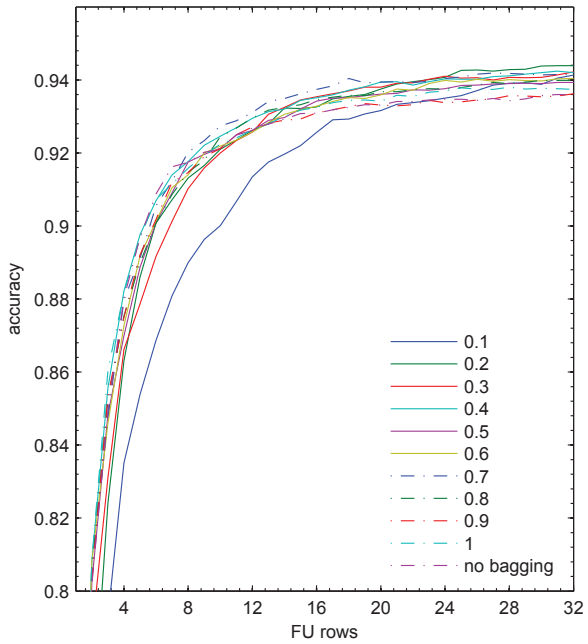


(b) B_11 dataset

Fig. 7. Average classification performance on training and test set over an increasing number of FU rows. Error bars show standard deviation.



(a) B_17 dataset



(b) B_11 dataset

Fig. 8. The effect of various bagging subset fractions b on classification accuracy over an increasing number of FU rows.

of a 10-fold cross-validation process on the combined training and test set. That is, the data is shuffled and divided into 10 folds, 9 of which are used for training, and the 10th for test data. The process is repeated 10 times, each time using a new fold for the test data, and the average of the test accuracies is reported. For the FUR and Random Forest approaches, the cross-validation results were averaged over 5 runs.

D. FPGA Implementation Analysis

In order to have an impression of the FPGA resource requirements for an instance of the EMG classification system,

TABLE I. CLASSIFIER ACCURACY ON 17 GESTURE DATA SET (IN %)

Classifier	training	test	xval
FUR	99.9	91.5	99.6
SVM	100.0	96.8	100.0
Random Forest	100.0	89.9	100.0

TABLE II. CLASSIFIER ACCURACY ON 11 GESTURE DATA SET (IN %)

Classifier	training	test	xval
FUR	99.9	94.4	99.8
SVM	100.0	99.6	100.0
Random Forest	100.0	91.0	100.0

TABLE III. FPGA RESOURCES REQUIRED FOR B_11 ON XC6SLX45.

Resource	Amount	Available	% of total
Slice registers	17952	54576	32
Slice LUTs	11763	12289	45
Occupied Slices	4803	6822	70

a VHDL-based implementation of the architecture was constructed. The implementation follows the VRC-based approach described in [23]. We chose to implement an instance of the B_11 classifier, with 16 rows per classifier as a tradeoff between accuracy and resource requirements. Implementation results for a Xilinx Spartan-6 XC6SLX45 device are shown in Tab. III. Note that the implementation only covers the FU rows and counters of the core classifier architecture and not a fully operational system with training logic.

V. DISCUSSION

From Fig. 7 it can be seen that a high classification accuracy can be obtained with a relatively small number of rows per category, and 16 rows could be a good accuracy-resource tradeoff point for a hardware implementation. However, the classifier seems to converge earlier for the B_11 dataset and has less variation in the evolved classifiers, which confirms that the B_17 dataset is more difficult and could benefit from an even higher number of FU rows. As can be seen in Tab. III, the B_11 instance of the classifier requires a moderate amount of FPGA resources. Moreover, results from [23] suggest that even smaller implementations should be possible using special techniques, without sacrificing classification accuracy. As an instance of the B_17 dataset would benefit from more resources, both in terms of a higher number of categories and a higher number of FU rows for improved accuracy, it would be beneficial to further investigate some methods for reducing the resource footprint. One approach could be to device a resource sharing scheme, where the same FPGA resources would compute the results of different FUs in a time-multiplexed manner. This should be viable given a fast reconfiguration scheme and the high classification speed of the classifier. Another approach could be to devise training methods for achieving higher classification rates with a low number of rows, one possibility could be a boosting-inspired approach [30].

The bagged subset training approach does not have a big effect on classification accuracy, although some improvement over training on the full set can be seen in Fig. 8. The effect is more pronounced on the more difficult B_17 dataset, which indicates that the approach could be promising for other more difficult problems, and that the best bagging parameters will be problem-dependent. We hypothesize that the stochastic nature of the evolutionary training process, in starting from random genomes to producing several different FU rows which all have high fitness, already provides a degree of robustness to the ensemble classifier. This may leave little room for further robustness through the bagging technique. However, a natural side effect of training on a reduced subset is that the fitness evaluation needs to process fewer vectors and the training time goes down. As Fig. 8 shows, each row can be trained with $b = 0.2$ times the full amount of training vectors while maintaining high accuracy. The reduction in training time is confirmed in Fig. 10, but these results are from the software simulated model with high overhead in the genetic operations, and it is believed that the effect could be more pronounced in a hardware implementation. Training time should decrease both because of cycling fewer vectors for each evaluation *and* the lower number of generations required to achieve maximum fitness on fewer vectors. The effect is seen best when using a high maximum number of generations on the B_17 set because evolution in this case tends to spend more generations on finding a good solution. Fig. 9 confirms the need to sample a new training subset for each FU row versus using a static subset, however the need becomes smaller as the subset size becomes larger.

The accuracy drop from training to test as seen in Tabs. I and II, can be explained by the unseen test data having shifted after the 10 minute break between the recordings, as a result of physical conditions. The results on the cross-validation process confirm this, since the classifier is near perfect when trained on samples from both recording sessions. Similar behavior is observed for the Random Forest classifier, which is not surprising given some shared characteristics with FUR in terms of ensembles and decision boundaries. The SVM classifier seems to generate more general decision boundaries which can accommodate better the shift in the test data, but also here the accuracy drop is observed, although to a lesser extent, for the B_17 data set. We speculate that a more advanced training strategy would be required to elevate the classification performance of FUR to the levels of SVM, however, this would likely come at the cost of increased architectural complexity.

A comprehensive comparison of the FUR architecture to other classification algorithms is not possible without hardware implementations of all methods. This would allow for a discussion on the tradeoffs between properties such classification accuracy, training and classification speed, and resource usage. While this is not the scope of this paper, one advantage of the ES-trainable FUR classifier system would be the hardware-friendly integration of the learning structure, while SVM needs a computationally complex learning algorithm.

VI. CONCLUSIONS AND FUTURE WORK

We have investigated the applicability of the FUR EHW classifier architecture to EMG signal classification with the BioSleeve system. Although not giving quite as high classification accuracy as a software-based SVM classifier, the performance is above 90% for a high number of classes and compares well with other approaches. The architecture can fit on a relatively small FPGA device and has potential for further resource optimization.

We have also investigated a method of reducing the training set for the incremental evolution-based classifier approach. The method shows that some accuracy gain can be attained whereas the biggest advantage is a reduced training set size resulting in shorter evolution times without loss in accuracy.

Future work includes a full hardware implementation of the classifier system including the training unit in order to evaluate training speed and power consumption. With the goal of long periods of battery-based operation, active channel selection methods for power saving should be explored in future versions. It would also be interesting to investigate a resource sharing scheme such that a high number of classes

would give less impact on resource usage. Further investigation into training procedures and classification properties of the architecture should also be pursued.

ACKNOWLEDGMENT

Kyrre Glette would like to acknowledge Adrian Stoica for hosting his research stay at the Jet Propulsion Laboratory, and the U.S. - Norway Fulbright Foundation for Educational Exchange for sponsorship. Part of this research was conducted at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Part of this work was sponsored by the Defense Advanced Research Projects Agency (DARPA) MTO under the auspices of Dr. Jack Judy through the Army Research Office, Contract No. MIPR2C080XR010.

REFERENCES

[1] M. Wolf, C. Assad, A. Stoica, K. You, H. Jethani, M. Vernacchia, J. Fromm, and Y. Iwashita, "Decoding Static and Dynamic Arm and Hand Gestures from the JPL BioSleeve," in *Proceedings IEEE Aerospace Conference*. IEEE, 2013, to appear.

[2] M. Wolf, C. Assad, M. Vernacchia, J. Fromm, and H. Jethani, "Gesture-Based Robot Control with Variable Autonomy from the JPL BioSleeve," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, to appear.

[3] I. Kajitani, T. Hoshino, D. Nishikawa, H. Yokoi, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, M. Iwata, D. Keymeulen, and T. Higuchi, "A Gate-Level EHW Chip: Implementing GA Operations and Reconfigurable Hardware on a Single LSI," in *Proceedings 2nd International Conference on Evolvable Systems (ICES)*, ser. LNCS, vol. 1478. Springer, 1998, pp. 1–12.

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

[5] B. Peerdeman, D. Boere, H. J. B. Witteveen, M. H. A. Huis in 't Veld, H. J. Hermens, S. Stramigioli, J. S. Rietman, P. H. Veltink, and S. Misra, "Myoelectric forearm prostheses: State of the art from a user-centered perspective," *Journal of rehabilitation research and development*, vol. 48, no. 6, pp. 719–738, August 2011.

[6] P. Zhou, M. Lowery, K. Englehart, H. Huang, G. Li, L. Hargrove, J. Dewald, and T. Kuiken, "Decoding a new neural–machine interface for control of artificial limbs," *Journal of neurophysiology*, vol. 98, no. 5, pp. 2974–2982, 2007.

[7] G. Li, A. Schultz, and T. Kuiken, "Quantifying pattern recognitionbased myoelectric control of multifunctional transradial prostheses," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 2, pp. 185–192, 2010.

[8] F. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, and N. Thakor, "Decoding of individuated finger movements using surface electromyography," *Biomedical Engineering, IEEE Transactions on*, vol. 56, no. 5, pp. 1427–1434, 2009.

[9] K. Wheeler, M. Chang, and K. Knuth, "Gesture-based control and emg decomposition," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 36, no. 4, pp. 503–514, 2006.

[10] 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.

[11] 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.

[12] 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.

[13] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, "Hardware Evolution at Function Level," in *Proceedings 4th Parallel Problem Solving from Nature (PPSN)*, ser. LNCS, vol. 1141. Springer, September 1996, pp. 62–71.

[14] 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.

[15] 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.

[16] 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.

[17] K. Glette, J. Torresen, T. Gruber, B. Sick, P. Kaufmann, and M. Platzner, "Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control," in *Proceedings 3rd NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. Los Alamitos, CA, USA: IEEE CS Press, 2008, pp. 32–39.

[18] P. Kaufmann, K. Glette, T. Gruber, M. Platzner, J. Torresen, and B. Sick, "Classification of electromyographic signals: Comparing evolvable hardware to conventional classifiers," *IEEE Transactions On Evolutionary Computation*, vol. 17, pp. 46–63, 2013.

[19] 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.

[20] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[21] K. Glette, J. Torresen, and M. Yasunaga, "Online Evolution for a High-Speed Image Recognition System Implemented On a Virtex-II Pro FPGA," in *Proceedings 2nd NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. Los Alamitos, CA, USA: IEEE CS Press, 2007, pp. 463–470.

[22] T. Martinek and L. Sekanina, "An evolvable image filter: Experimental evaluation of a complete hardware implementation in fpga," *Lecture Notes in Computer Science*, vol. 2005, no. 3637, pp. 76–85, 2005.

[23] 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.

[24] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[25] K. Glette, J. Torresen, P. Kaufmann, and M. Platzner, "A Comparison of Evolvable Hardware Architectures for Classification Tasks," in *Proceedings 8th International Conference on Evolvable Systems (ICES)*, ser. Lecture Notes in Computer Science, vol. 5216. Springer, 2008, pp. 22–33.

[26] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[27] C. Chang and C. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[29] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "Yale: Rapid prototyping for complex data mining tasks," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, Eds. New York, NY, USA: ACM, August 2006, pp. 935–940.

[30] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*. Springer, 1995, pp. 23–37.