Event-based Low-Power and Low-Latency Regression Method for Hand Kinematics from Surface EMG

Event-based Low-Power and Low-Latency Regression Method for Hand Kinematics from Surface EMG

Marcello Zanghieri[1◇*], Simone Benatti[1,2], Luca Benini[1,3], Elisa Donati[3,4]
[1] University of Bologna, Bologna, Italy — {marcello.zanghieri2, simone.benatti, luca.benini}@unibo.it
[2] University of Modena and Reggio Emilia, Modena, Italy — simone.benatti@unimore.it
[3] ETH Zurich, Zurich, Switzerland — lbenini@iis.ee.ethz.ch, eldonati@ethz.ch
[4] University of Zurich, Zurich, Switzerland — elisa@ini.uzh.ch

***Abstract*—Human-Machine Interfaces (HMIs) are a rapidly progressing field, and gesture recognition is a promising method in industrial, consumer, and health use cases. Surface electromyography (sEMG) is a State-of-the-Art (SoA) pathway for human-to-machine communication. Currently, the research goal is a more intuitive and fluid control, moving from signal classification of discrete positions to continuous control based on regression. The sEMG-based regression is still scarcely explored in research since most approaches have addressed classification. In this work, we propose the first event-based EMG encoding applied to the regression of hand kinematics suitable for working in streaming on a low-power microcontroller (STM32 F401, mounting ARM Cortex-M4). The motivation for event-based encoding is to exploit upcoming neuromorphic hardware to benefit from reduced latency and power consumption. We achieve a Mean Absolute Error of 8.8 ± 2.3 degrees on 5 degrees of actuation on the public dataset NinaPro DB8, comparable with the SoA Deep Neural Network (DNN). We use $9\times$ less memory and $13\times$ less energy per inference, with $10\times$ shorter latency per inference compared to the SoA deep net, proving suitable for resource-constrained embedded platforms.**

***Index Terms*—electromyography, EMG, surface electromyography, sEMG, hand kinematics, time series, event-based, silicon cochlea, spike trains, regression, machine learning, ML, embedded, microcontrollers, MCU, STM32, ultra-low-power, low-latency, real-time, streaming, event-driven, neuromorphic.**

I. Introduction

Human-Machine Interfaces (HMIs) are an active and rapidly advancing research domain in which gesture recognition is one of the most promising approaches for industrial, commercial, and health scenarios such as robotics, augmented reality, and rehabilitation & prosthetics [1]. Surface electromyography (sEMG)-based HMIs are the State-of-the-Art (SoA) in the communication pathway between a human and a device driven via EMG signals processing [2]. They rely on the analysis of the electrical activity of muscles sensed via electrodes positioned on the surface of the skin in a completely non-invasive way (in sharp contrast with intramuscular EMG), with great potential for wearable control systems.

Finding an effective mapping from sEMG to control commands is not a trivial task, and SoA approaches tackle the problem by resorting to Machine Learning (ML) or Deep Learning (DL) [3] where sEMG signals are mapped to a given set of gestures (classification) [4] or to continuous degrees of freedom (regression) [5]. As opposed to conventional pattern recognition, deep neural networks are able to learn signal features at training time, often outperforming the handcrafted features needed for non-deep ML; the learned information extraction is potentially optimal since data-driven, but not easily explainable.

Current research aims for a more natural and intuitive control, hence the need to move from a limited set of predefined positions to a continuous control that can be performed with a regression-based approach. So far, regression approaches to sEMG represent a minority, while the vast majority of the literature is concerned with classification, focusing mostly on Deep Neural Networks (DNNs) to counteract the inter-session variability of sEMG and make classification robust in the long run through regularization [4] or adaptation [6]. In contrast, regression works are still scarce in the literature compared to robust classification efforts. Most existing proposed solutions for sEMG regression produce DL models that yield a low regression error [7, 8] but do not take fully into account the memory, latency, and energy constraints of resource-constrained embedded computational devices. Some proposed deep networks require processors designed explicitly for linear algebra [5] to be executed with ultra-low power consumption (tens of milliwatts power envelope), a regime suitable for long-term wearable devices.

In the veins of exploring solutions for ultra-low-power execution of computationally demanding tasks, Spiking Neural Networks (SNN) are an emerging class of artificial neural

◇Corresponding author.
*Work done while visiting at the Neuromorphic Cognitive Systems (NCS) research group of the Institute of Neuroinformatics (INI), University of Zurich and ETH Zurich, Zurich, Switzerland.

The research presented in this work was supported in part by the EU's Horizon 2020 project BonsAPPs (EU grant agreement 101015848), and by the Marco Polo Programme of University of Bologna.

networks specifically designed to process data in the format of spike trains (i.e., binary with sparse 1's, coming as a stream in continuous physical time). Neurons in an SNN have a state that emulates the biological membrane potential: it is excited at the reception of events, and decays over time; upon crossing a threshold, a neuron *fires*, i.e., transmits an event (a *spike*) to the connected neurons [9]. Neuromorphic processors, either digital [10] or mixed-signal [11], are an ideal substrate for the deployment of SNNs, since they are accelerators for the sequential computation of the emulated potential varying in time. Moreover, neuromorphic processors perform event-proportional computing [10]: since each neuron influences the others only when it fires, events are sparse spike trains, which cause only sparse updates of the net's neurons' states, greatly reducing latency and energy consumption compared to inference in a conventional neural network that requires to compute the entirety of activation maps.

Integration of sEMG acquisition systems with neuromorphic processors requires converting the digital sEMG raw data into sequences of events, i.e., timestamps associated with each channel to be used as input spike train for the event-based processing. Existing works on processing sEMG on event-driven hardware show that separation patterns can be extracted with SNNs consuming $< 1\,\mathrm{nJ}$ per spike, amounting to as little as $0.05\,\mathrm{mW}$ (hundredths of a milliwatt) of total power [12]. However, these works do not address regression yet, but implement classification [12, 13, 14] or provide insight into the empirical activation patterns and their class-separability [15].

In this work, we propose a hybrid method based on event-based EMG encoding, a bio-inspired feature extraction, combined with regression implemented on a low-power processor ideal for embedded solutions. Our goal is to explore a goodness-complexity tradeoff for sEMG regression against the existing literature that focuses on DL models [5, 7, 8, 16]. Our contribution is three-fold:

- we propose an event-based encoding strategy for the sEMG that works in streaming, i.e., consumes inputs one-by-one, producing a stream of output spike events;
- we tune our encoding scheme on the real sEMG regression dataset NinaPro DB8, achieving a Mean Absolute Error of 8.8 ± 2.3 degrees, comparable with the SoA DNN, proving that our spike conversion preserves enough information for a fine task like regression;
- we profile the resource requirements and execution of our setup on a commercial digital microcontroller, getting $9\times$ smaller memory footprint, $10\times$ shorter latency, and $13\times$ lower energy consumption per inference compared to the SoA deep net, proving our method a perfect fit for resource-constrained embedded platforms.

We release open-source the code of this work at https://github.com/pulp-bio/event-based-semg-regression.

II. Materials & Methods

A. sEMG and NinaPro Database 8

The EMG signal is a very informative marker of muscular activity because it arises from the trains of motor unit action potentials in muscle fibers [17]. Typically, the EMG has amplitude between $10\,\mu\mathrm{V}$ and $1\,\mathrm{mV}$ and bandwidth up to $2\,\mathrm{kHz}$. When targeting health or consumer HMIs, non-invasiveness and acceptance are enabled by sEMG, i.e., the acquisition of EMG via conductive electrodes on the skin surface. Extracting information from the sEMG is made difficult by motion artifacts, signal variability [18, 19], and noise sources such as floating ground, crosstalk, and power line interference.

The Non-Invasive Adaptive hand Prosthetics Database 8 (NinaPro DB8)[1] [20] is one of the public datasets collected to enable the ML and DL era of sEMG processing [3]. In addition to 10 able-bodied participants, it includes 2 right trans-radial amputees, measuring hand movements contralateral to the sensed forearm for all participants. To target modelling of fluid motion and transitions, the protocol consists in slow gestures lasting between $6\,\mathrm{s}$ and $9\,\mathrm{s}$ with rests of approximately $3\,\mathrm{s}$, and it includes single-finger and functional (e.g., grasp) movements. The sEMG was acquired by 16 active double-differential sensors (Delsys Trigno IM Wireless EMG) on the right forearm. Angular position of hand joints was measured by a 18-DoF Cyberglove 2 on the left hand. The shared data are upsampled to $2\,\mathrm{kHz}$ and post-synchronized.

To frame the regression problem, the dataset authors define 5 Degrees of Actuation (DoAs) as linear combinations of the 18 measured DoFs (Eq. S2 in Supplementary Material of [20]) to directly address five relevant hand movements: thumb rotation, thumb flexion, index flexion, middle flexion, ring+little flexion. We target the same DoAs as ground truth, applying the same DoF-to-DoA coefficients matrix. So, the multivariate regression problem consists in optimizing the fit from the 16 sEMG signals to the 5 DoAs.

Recent works tackling NinaPro DB are the ones by Zhou et al. [21] and by Bao et al. [16], whereas the SoA is represented by Zanghieri et al. [5]. Zhou et al. [21] achieved $\geq 90\%$ recognition accuracy applying DL on time-frequency domain features. However, addressing NinaPro DB8 as a classification largely under-exploits the available data, and is even advised against by the dataset authors due to the deliberately long transients . Bao et al. [16] did perform a regression, reaching an average R^2 of 0.68 and an RMSE of 13.5 degrees with a deep network based on a Kalman filter, but they too underexploited the dataset by only addressing 3 DoAs (index, middle, ring+little) out of 5, dismissing the 2 thumb DoAs without discussing this limitation.

The literature SoA on NinaPro DB8 is the work by Zanghieri et al. [5], who achieved a Mean Absolute Error (MAE) of 6.89 degrees by deploying a Temporal Convolutional Network (TCN). That SoA TCN can run in real-time provided that specialized hardware is available, namely a microcontroller mounting a parallel 8-core processor designed for energy-efficient matrix multiplication [22].

In this work, we want to explore an accuracy-computation tradeoff opposite to that pursued in the mentioned SoA work: our research question is to research the regression accuracy

[1] http://ninaweb.hevs.ch/DB8

achievable within a limited budget of memory, latency, and energy.

B. Encoding surface EMG to Events

We perform EMG-to-spike conversion, i.e., encoding of the sEMG data to an event-based format, which is a simplification of the cochlear method. Cochlear spike conversion is a signal processing pipeline inspired by the mechanical separation of acoustic frequencies taking place in the mammalian cochlea prior to transduction and encoding into neural train spikes. The natural cochlea has stimulated bio-inspired hardware designers to implement silicon cochleas, i.e., circuitry mimicking the natural spike conversion, either in digital [23] or analog solutions [24, 25].

The method consists of two stages: (1) bandpass filtering and (2) Leaky Integrate-and-Fire (LIF) neurons.

1. *Bandpass filtering.* Each input channel is passed into a bank of bandpass filters. We used 4 bands covering the whole bandwidth from 0 to f_{Nyquist}, based on 4-th order Butterworth filters. We set cutoff frequencies with exponential spacing (adjusted to reach zero):

$$f_n = \frac{e^{n/N_{\text{bands}}} - 1}{e - 1} \cdot f_{\text{Nyquist}} \qquad n = 0, \cdots, N_{\text{bands}} \tag{1}$$

where $N_{\text{bands}} = 4$ is the number of bands and f_n is the n-th cutoff frequency between adjacent bands. Since NinaPro DB8 has $f_{\text{sample}} = 2\,\text{kHz}$ and thus $f_{\text{Nyquist}} = 1\,\text{kHz}$, the resulting cutoff frequencies are

$$f_{0,1,2,3,4}\,[\text{Hz}] \;=\; 0.0,\; 32.1,\; 119.2,\; 356.1,\; 1000.0 \tag{2}$$

The gains of the 4 4-th order Butterworth filters corresponding to the 4 bands are shown in Fig. 1. NinaPro DB8 is released already filtered with a 4-th order bandpass Butterworth between 10 Hz and 500 Hz, and we did not apply any additional cleaning prior to splitting bands. By passing each input channel in the 4 filters, we expand the number of input channels from 16 to 64. Then, all signals are full-wave-rectified.

2. *LIF neurons.* The spike conversion in strict sense takes place in 64 LIF neurons, each receiving the output of one filter as an injected input current. LIF neurons are a simplified model of the natural neuron, where the state is described by a *membrane potential* $V_{\text{mem}}(t)$ obeying the linear electrical law

$$\frac{\mathrm{d}V_{\text{mem}}}{\mathrm{d}t} = -\frac{(V_{\text{mem}}(t) - \mathcal{E}_{\text{leak}}) - \frac{I_{\text{inj}}(t)}{g_{\text{leak}}}}{\tau} \tag{3}$$

where τ is the membrane relaxation time, $\mathcal{E}_{\text{leak}}$ is the leak reversal potential, I_{inj} is the injected current, and g_{leak} is the leak conductance. Each time V_{mem} crosses a threshold V_{thr}, the neuron emits a spike event corresponding to the time of crossing t_{fire}. After each spike the LIF neuron undergoes a *refractory time* t_{refr}, i.e. a time interval $[t_{\text{fire}}, t_{\text{fire}} + t_{\text{refr}}]$ during which V_{mem} is forced to a reset value V_{reset}, and neither the decay nor the inhomogeneous driving term $I_{\text{inj}}(t)/g_{\text{leak}}$ act:

$$V_{\text{mem}}(t) \equiv V_{\text{reset}} \qquad t \in [t_{\text{fire}}, t_{\text{fire}} + t_{\text{refr}}]\,. \tag{4}$$

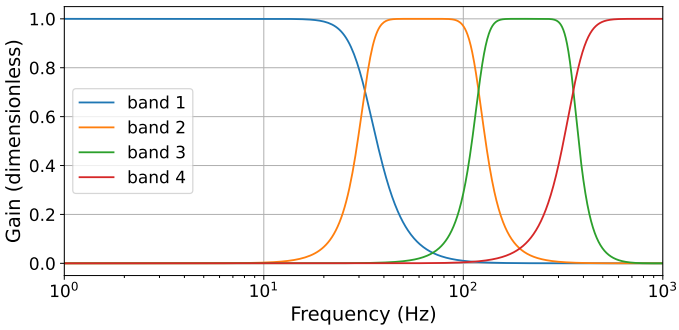

Fig. 1: Gains of the four 4-th order Butterworth filters of the used frequency bands. They stop at 1 kHz since it is the Nyquist frequency of the NinaPro DB8 dataset.

The clearer change of variables

$$x(t) \triangleq \frac{V_{\text{mem}}(t) - \mathcal{E}_{\text{leak}}}{V_{\text{thr}} - \mathcal{E}_{\text{leak}}} \tag{5}$$

$$x_{\text{drive}}(t) \triangleq \frac{I_{\text{inj}}(t)}{g_{\text{leak}}} \tag{6}$$

cleans away all electrical quantities (unneeded in a numerical simulation) and yields a dimensionless state with firing threshold $x_{\text{thr}} = 1$ and law

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -\frac{x(t) - x_{\text{drive}}(t)}{\tau} \tag{7}$$

making it clearer that the injected current plays the role of external driving term. Each of the 64 filtered signals is used to drive an independent LIF neuron, which carries out the spike conversion. We parameterize the LIF neurons based on three values: (*i*) the empirical gain g_{data} to convert the arbitrary-units filtered signals into $x_{\text{drive}}(t)$:

$$x_{\text{drive}}(t) = g_{\text{data}} \cdot |x_{\text{bandpassed}}(t)|\,; \tag{8}$$

we explored 5 values from $1.0 \cdot 10^5$ to $1.0 \cdot 10^6$, approximately exponentially spaced:

$$g_{\text{data}} \in \left\{1.0 \cdot 10^5, 1.7 \cdot 10^5, 3.0 \cdot 10^5, 5.5 \cdot 10^5, 1.0 \cdot 10^6\right\} \tag{9}$$

(*ii*) the membrane relaxation time, which we set to $\tau = 10\,\text{ms}$; (*iii*) the refractory time, for which we explored $t_{\text{refr}} \in \{1\,\text{ms}, 2\,\text{ms}\}$.

To feed spike trains to regression, we adopted a rate encoding. We computed rates based on a causal exponential decay kernel [26] assigning each spike a weight of $\exp(-t/\tau_{\text{post}})$ for $t \geq 0$ in the future. In contrast with discrete windowing, this kernel does not require storing a variable buffer of events timestamps but only one floating-point number. This causal exponential-kernel rate can take values ≥ 0 and $< 1/\left(1 - \exp\left(-t_{\text{refr}}/\tau_{\text{post}}\right)\right)$. We explore 12 values from 1 ms to 5 s, approximately exponentially spaced:

$$\tau_{\text{post}}\,[\text{ms}] \in \{1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000\}. \tag{10}$$

For clarity and compactness, we report all the explored settings of the event-based encoding in Table I.

TABLE I: Settings explored to tune the event-based encoding.

Parameter role	symbol	Explored values
gain from data to dimensionless LIF driving term x_{drive}	g_{data}	$1.0 \cdot 10^5$, $1.7 \cdot 10^5$, $3.0 \cdot 10^5$, $5.5 \cdot 10^5$, $1.0 \cdot 10^6$ (dimensionless)
refractory time of LIF neurons	t_{refr}	1 ms, 2 ms
decay time of post-LIF causal exponential kernel	τ_{post}	1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000 milliseconds

LIF dynamics were implemented in Python 3.8 with the simulator Brian2[2] [27] v. 2.5 and run with simulation timestep 500 µs, found to be a good tradeoff between time resolution and computation time; since this is equal to the sampling time of the data, each filtered sample drives its channel's LIF for exactly 1 simulation step.

C. Regression

We use linear regression to prioritize low memory and computation requirements over refined accuracy. We do not apply any preprocessing to ground-truth DoA signals, nor any postprocessing to the regression outputs. The $\mathbb{R}^{64} \to \mathbb{R}^5$ multivariate regression problem is parameterized by a 5×64 coefficients matrix plus a 5-valued intercept; in `fp32`, this amounts to a memory footprint of 1576 bytes (including input and output), and a number of operations of 645 FLOP.

As suggested by NinaPro DB8's authors, we merged sessions 1 and 2 (10 repetitions of each gesture) and used them for training and validation, and we used session 3 (2 repetitions of each gesture) for testing. We performed all experiments separately for each subject, without any multi-subject training or inter-subject validation.

For training, we downsampled the spike trains with a time step of 100 ms to keep the training computation time < 8 hours for the whole dataset; at testing, we called the inference on the spike trains every 16 ms, which is the same time step as the SoA work [5] we compare with.

D. Profiling

We profiled the proposed processing based on memory footprint, number of operations, power consumption, latency per inference, and energy consumption per inference. To measure the latency per inference and to determine the energy consumption per inference, we implemented the spike conversion (filter banks + LIF neurons) and the linear inference on an STM32 F401RE microcontroller, which mounts an ARM Cortex-M4 processor. STM32 F401RE is not designed for event-based sparse data processing and does not implement event-driven execution. Our motivation for profiling on this platform is to showcase how memory-, time-, and energy-efficient our proposed setup is, even when run on a commercial general-purpose microcontroller.

We programmed it in C and compiled with optimization `-Ofast`. Latency was measured using the debugger of the environment STM32CubeIDE v. 1.11, whose overhead produces a variability of cycle counts in the order of 10 cycles; at a clock frequency of 84 MHz, this amounts to an uncertainty of 0.1 µs, which is accurate enough for our purpose. Power consumption was determined based on the value of 146 µA/MHz reported in the datasheet[3] ; at clock frequency 84 MHz with a power supply of 3.3 V, this amounts to a power consumption of 40.5 mW. Energy per inference was determined by multiplying by experimentally measured latency.

In Section III, we show that our pipeline can run in streaming, i.e., consuming one sEMG sample per channel at a time, fitting 10 update steps for each of the 64 LIF neurons within a latency of 500 µs, which is the sampling rate of the NinaPro DB8 dataset. Porting to actual event-driven neuromorphic devices (either digital [10] or mixed-signal [11]) will constitute our future work.

III. Experimental Results

A. Evaluation Metrics

We evaluate the regression using the Mean Absolute Error (MAE), measured in degrees and defined as

$$\text{MAE} = \frac{1}{N_{\text{infer}} N_{\text{DoA}}} \sum_{i=1}^{N_{\text{infer}}} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_1 \tag{11}$$

where $\mathbf{y}_i, \hat{\mathbf{y}}_i \in \mathbb{R}^5$ are the multivariate ground truth and estimation, measured in degrees, corresponding to the i-th inference, respectively, $\|\cdot\|_1$ is the L_1-norm, $N_{\text{DoA}} = 5$ is the number of DoAs (defined in Subsection II-A), and N_{infer} is the total number of inferences in each subject's Session 3, obtained by calling one inference every 16 ms (as explained in Subsection II-C). The MAE is a reliable metric of the end-to-end control goodness because it has the same scale as the target joint angles; moreover, MAE is a first-order statistic, thus more robust against outliers compared to (R)MSE or the multivariate coefficient of determination R^2, which are quadratic. We average all MAEs over time (hence over movement types and repetitions) and over DoAs, as expressed by Equation 11; and over all the 12 subjects; this allows comparison with [5].

We profile our processing pipeline by determining memory footprint, number of operations, latency per inference (in cycles and milliseconds), and energy per inference, as explained in Subsection II-D.

B. Regression Accuracy

We report regression results in Table II, which displays the optimal decay time $\tau_{\text{post}}^{\text{best}}$ for the causal exponential kernel $\exp(-t/\tau_{\text{post}})$ and the corresponding MAE, obtained for the explored values of the data gain g_{data} and the LIF refractory time t_{refr}, as detailed in Subsection II-B. These results provide insights both into the best settings and general trends.

Focusing on the optimal settings $g_{\text{data}}^{\text{best}} = 3.0 \cdot 10^5$ and t_{refr} equal to 1 ms or 2 ms, we show in Fig. 2 the curve for the

[2] https://github.com/brian-team/brian2

[3] https://www.st.com/en/microcontrollers-microprocessors/stm32f401re

TABLE II: Best decay time $\tau_{\text{post}}^{\text{best}}$ of the causal exponential kernel and regression error obtained for each explored combination of data gain g_{data} and t_{refr} (details in Subsection II-B). Best results in bold.

SETTINGS		RESULTS	
gain (dimensionless)	**refractory time (ms)**	**best decay of kernel $\tau_{\text{post}}^{\text{best}}$ (ms)**	**MAE (degrees)**
$1.0 \cdot 10^5$	1.0	500	9.47 ± 2.57
	2.0	500	9.38 ± 2.52
$1.7 \cdot 10^5$	1.0	500	9.04 ± 2.44
	2.0	500	8.95 ± 2.38
$3.0 \cdot 10^5$	1.0	500	$\mathbf{8.84 \pm 2.28}$
	2.0	500	$\mathbf{8.84 \pm 2.26}$
$5.5 \cdot 10^5$	1.0	500	9.02 ± 2.36
	2.0	500	9.06 ± 2.33
$1.0 \cdot 10^6$	1.0	500	9.39 ± 2.53
	2.0	500	9.58 ± 2.62

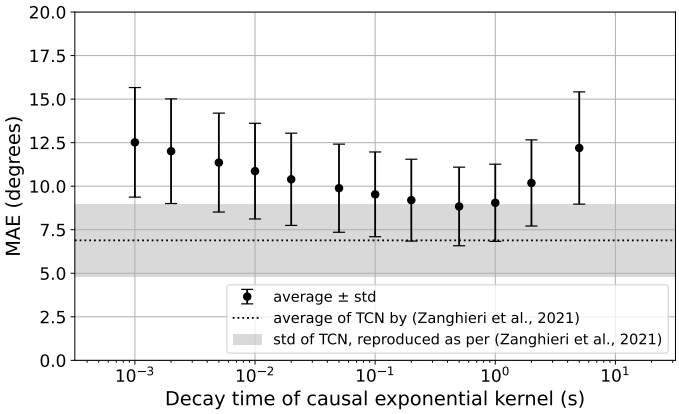

Fig. 2: Tuning curve of the decay time $\tau_{\text{post}}^{\text{best}}$ of the causal exponential kernel, for the optimal settings data gain $g_{\text{data}} = 3.0 \cdot 10^5$ and refractory time $t_{\text{refr}} = 2\,\text{ms}$.

search on τ_{post}, with which we identified the optimal value of the causal exponential kernel length, i.e., $\tau_{\text{post}}^{\text{best}} = 500\,\text{ms}$. The curve is convex because a too-short (too-long) decay time of the exponential kernels weighs too much the recent (old) spikes, i.e. computes the rate at a timescale that does not match the actual timescale of the hand's kinematics. The best MAE of 8.84 ± 2.26 degrees is consistent within 1 standard deviation with the SoA value of the TCN in [5], which is 6.89 ± 2.08 degrees. The TCN's MAE standard deviation is not reported in the SoA paper [5], and we determined it by reproducing the setup of [5]. Compatibility within 1 standard deviation suggests that our setup yields a regression quality as good as the SoA from a practical point of view. The MAE standard deviations in the range from 2.0 degrees to 2.6 degrees are an empirical measure of the general, natural variability in angular regression error across DoAs and subjects; in contrast, the literature reporting standard deviations of regression accuracy only reports the range [20] or the standard deviation [16] of the determination coefficient R^2, which is dimensionless and standardized by DoA, hence not informative about the physical scale of the error variability.

Regarding the general trends outside the best settings, we can see that the choice of the refractory time has little impact compared to the choice of data gain since the former always produces differences in MAE of less than 0.1 standard deviations, and the range of MAEs obtained is mainly due to the data gain setting. Another strong general trend is the consensus about $\tau_{\text{post}}^{\text{best}} = 500\,\text{ms}$ since even the grid-adjacent values of $200\,\text{ms}$ or $1000\,\text{ms}$ never turn out to be optimal; we interpret this consensus as a general estimation of the characteristic time scale of the variation of the kinematics in the NinaPro DB8 data.

It is worth remarking that τ_{post} by no means involves a delay in computation due to a wait. The decay due to the causal exponential kernel is applied recursively at every update of the firing rate: at each simulation step, the rate is updated by multiplying it by $\exp(-\Delta t_{\text{sim}}/\tau_{\text{post}})$, where Δt_{sim} is the simulation timestep, then incrementing by $+1$ if the corresponding LIF neuron has fired at the current simulation step. This sequential update is performed at each step, requiring a lightweight computation, regardless of the numerical value of τ_{post}. In detail, since $\exp(-\Delta t_{\text{sim}}/\tau_{\text{post}})$ is constant and is pre-computed once for all, updating the rate r as $r \leftarrow r \cdot \exp(-\Delta t_{\text{sim}}/\tau_{\text{post}}) + 1\,[\text{if spike}]$ only requires 1 multiplication, 1 condition test, and (if test if positive) 1 addition per neuron per simulation step. A long τ_{post}, such as $\tau_{\text{post}}^{\text{best}} = 500\,\text{ms}$, simply means that the relative decrease per simulation timestep is small.

C. Profiling

We report the profiling results on the STM32 F401RE commercial microcontroller in Table III. Compared with the TCN of [5], the proposed regression setup has $9.8\times$ smaller memory footprint, $10.6\times$ shorter latency, and $13.3\times$ lower energy consumption per inference. Thus, the event-based processing proposed in this work achieves an accuracy comparable to the SoA [5] while fitting in a much more limited resource budget. The memory saving is due to the fact that we do not buffer temporal data, always consuming data in streaming except for the filters' states of the initial filtering stage.

In particular, we notice that the latency of $\lesssim 450\,\mu\text{s}$ per inference is shorter than the sampling time of the NinaPro DB8 dataset (namely, $< 500\,\mu\text{s}$) and of most sEMG applications. Thus, our method meets the real-time constraints of an sEMG-based gesture recognition device.

IV. Conclusion

In this work, we have presented a solution for estimating the hand kinematics from sEMG relying on an event-based encoding of the input data. This is the first proposal of an event-based encoding of sEMG oriented to regression, a more complex task than classification. The regression error comparable with existing solutions proves that the spike event format preserves the information for this finer task while fitting much lower resource requirements than the dominant Deep Learning

TABLE III: Profiling of the proposed event-based encoding and regression processing, compared with the SoA TCN setup.

	Platform	Memory (kB)	Operations	Latency cycles (k)	Latency time (µs)	Power (mW)	Energy per inference (µJ)	MAE (degrees) $\mu \pm \sigma$
TCN of [5]	GAP8[a] [22]	70.90	$6.32{\cdot}10^6$ in int32[b]	476	4760	51.0	243.0	6.89 ± 2.08[c]
This work	STM32 F401RE	7.19	$7.07{\cdot}10^3$ in fp32	37.69 ± 0.02	448.6 ± 0.2	40.5	18.2	8.84 ± 2.26

[a] https://greenwaves-technologies.com/gap8_mcu_ai/
[b] 8-bit models use 8-bit weights and maps, but layers run in int32 and outputs are requantized to uint8 after activation.
[c] The TCN's MAE standard deviation is not reported in [5], so we determined it by reproducing the setup of the original work.

approaches. The long-term research direction after this work is to assess event-based computing paradigms and computing platforms as new candidates for sEMG regression, in addition to the convolutional approach based on matrix multiplication on temporal data buffers. We will target implementation onto event-driven hardware to take full advantage of the energy efficiency of event-proportional computing.

Acknowledgment

We thank David Kubánek for sharing the code[4] of his alternative exploratory PCA + spike-conversion regression on NinaPro DB8.

References

[1] J. Cannan et al. "Human-Machine Interaction (HMI): A Survey". In: *University of Essex* (2011).
[2] L. Guo et al. "Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review". In: *IEEE THMS* 51.4 (2021). DOI: 10.1109/THMS.2021.3086003.
[3] A. Phinyomark et al. "EMG Pattern Recognition in the Era of Big Data and Deep Learning". In: *MDPI BDCC* 2.3 (2018). DOI: 10.3390/bdcc2030021.
[4] M. Zanghieri et al. "Temporal Variability Analysis in sEMG Hand Grasp Recognition using Temporal Convolutional Networks". In: *IEEE AICAS 2020*. 2020. DOI: 10.1109/AICAS48895.2020.9073888.
[5] M. Zanghieri et al. "sEMG-based Regression of Hand Kinematics with Temporal Convolutional Networks on a Low-Power Edge Microcontroller". In: *IEEE COINS 2021*. 2021. DOI: 10.1109/COINS51742.2021.9524188.
[6] U. Côté-Allard et al. "Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning". In: *IEEE TNSRE* 27.4 (2019). DOI: 10.1109/TNSRE.2019.2896269.
[7] R. C. Sîmpetru et al. "Accurate Continuous Prediction of 14 Degrees of Freedom of the Hand from Myoelectrical Signals through Convolutive Deep Learning". In: *IEEE EMBC 2022*. 2022. DOI: 10.1109/EMBC48229.2022.9870937.
[8] R. C. Sîmpetru et al. "Sensing the Full Dynamics of the Human Hand with a Neural Interface and Deep Learning". In: *bioRxiv preprint*. 2022. DOI: 10.1101/2022.07.29.502064.
[9] W. Maass. "Networks of Spiking Neurons: The Third Generation of Neural Network Models". In: *Neural Networks* 10.9 (1997). DOI: 10.1016/S0893-6080(97)00011-7.
[10] A. Di Mauro et al. "SNE: an Energy-Proportional Digital Accelerator for Sparse Event-Based Convolutions". In: *DATE 2022*. 2022. DOI: 10.23919/DATE54114.2022.9774552.
[11] S. Moradi et al. "A Scalable Multicore Architecture with Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)". In: *IEEE TBCAS* 12.1 (2018). DOI: 10.1109/tbcas.2017.2759700.
[12] E. Donati et al. "Discrimination of EMG Signals using a Neuromorphic Implementation of a Spiking Neural Network". In: *IEEE TBioCAS* 13.5 (2019). DOI: 10.1109/TBCAS.2019.2925454.
[13] Y. Ma et al. "Neuromorphic Implementation of a Recurrent Neural Network for EMG Classification". In: *IEEE AICAS 2020*. 2020. DOI: 10.1109/AICAS48895.2020.9073810.
[14] Y. Ma et al. "EMG-based Gestures Classification Using a Mixed-Signal Neuromorphic Processing System". In: *IEEE JETCAS* 10.4 (2020). DOI: 10.1109/JETCAS.2020.3037951.
[15] E. Donati et al. "Processing EMG Signals using Reservoir Computing on an Event-based Neuromorphic System". In: *IEEE BioCAS 2018*. 2018. DOI: 10.1109/BIOCAS.2018.8584674.
[16] T. Bao et al. "A Deep Kalman Filter Network for Hand Kinematics Estimation using sEMG". In: *Pattern Recognition Letters* 143 (2021). DOI: 10.1016/j.patrec.2021.01.001.
[17] R. M. Rangayyan. *Biomedical Signal Analysis*. John Wiley & Sons, 2015. DOI: 10.1002/9781119068129.
[18] A. Burrello et al. "Tackling Time-Variability in sEMG-based Gesture Recognition with On-Device Incremental Learning and Temporal Convolutional Networks". In: *IEEE SAS 2021*. 2021. DOI: 10.1109/SAS51076.2021.9530007.
[19] E. Donati et al. "Long-Term Stable Electromyography Classification using Canonical Correlation Analysis". In: (2023). DOI: 10.48550/arXiv.2301.09729.
[20] A. Krasoulis et al. "Effect of User Practice on Prosthetic Finger Control with an Intuitive Myoelectric Decoder". In: *Frontiers in Neuroscience* 13 (2019). DOI: 10.3389/fnins.2019.00891.
[21] X. Zhou et al. "Time–Frequency Feature Transform Suite for Deep Learning-based Gesture Recognition using sEMG Signals". In: *Robotica* 41.2 (2023). DOI: 10.1017/S026357472200159X.
[22] E. Flamand et al. "GAP-8: A RISC-V SoC for AI at the Edge of the IoT". In: *IEEE ASAP 2018*. 2018. DOI: 10.1109/ASAP.2018.8445101.
[23] A. Jiménez-Fernández et al. "A Binaural Neuromorphic Auditory Sensor for FPGA: A Spike Signal Processing Approach". In: *IEEE TNNLS* 28.4 (2017). DOI: 10.1109/TNNLS.2016.2583223.
[24] S.-C. Liu et al. "Event-based 64-channel Binaural Silicon Cochlea with Q Enhancement Mechanisms". In: *IEEE ISCAS 2010*. 2010. DOI: 10.1109/ISCAS.2010.5537164.
[25] M. Yang et al. "A 0.5V 55µW 64×2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing". In: *IEEE JSSC* 51.11 (2016). DOI: 10.1109/JSSC.2016.2604285.
[26] I. M. Park et al. "Kernel Methods on Spike Train Space for Neuroscience: A Tutorial". In: *IEEE MSP* 30.4 (2013). DOI: 10.1109/MSP.2013.2251072.
[27] M. Stimberg et al. "Brian 2, an Intuitive and Efficient Neural Simulator". In: *eLife* 8 (2019). DOI: 10.7554/eLife.47314.

[4] https://github.com/davidkubanek/SNN-hand-kinematics-estimation-from-sEMG-signals