# sEMG-based Gesture Recognition with Spiking Neural Networks on Low-power FPGA*

Matteo Antonio Scrugli[0000−0001−7441−1425], Gianluca Leone[0000−0001−5265−0759], Paola Busia[0000−0002−1434−9858], and Paolo Meloni[0000−0002−8106−4641]

University of Cagliari, Cagliari, Italy
(matteo.scrugli, gianluca.leone94, paola.busia, paolo.meloni)@unica.it

**Abstract.** Classification of surface electromyographic (sEMG) signals for the precise identification of hand gestures is a crucial area in the advancement of complex prosthetic devices and human-machine interfaces. This study presents a real-time sEMG classification system, exploiting a Spiking Neural Network (SNN) to distinguish among twelve distinct hand gestures. The system is implemented on a Lattice iCE40-UltraPlus FPGA, explicitly designed for low-power applications. Evaluation on the NinaPro DB5 dataset confirms an accuracy of 85.6%, demonstrating the model's effectiveness. The power consumption for this architecture is approximately 1.7 mW, leveraging the inherent energy efficiency of SNNs for low-power classification.

**Keywords:** Spiking Neural Networks · Real-time monitoring · Healthcare.

## 1 Introduction

The continuous development of artificial intelligence and neural networks offers numerous opportunities for long-term monitoring and personalized applications in the healthcare domain. One area of considerable interest is the precise interpretation of surface electromyographic signals (sEMG) for accurate identification of hand gestures, a cornerstone for the evolution of sophisticated prosthetic devices and human-machine interfaces. Real-time monitoring of sEMG poses great challenges due to stringent accuracy and resource limitations in wearable/embedded systems, making it a key research area. Spiking Neural Networks (SNNs) emerge as a promising solution with energy-efficient, event-driven processing. However, exploiting the benefits of event-driven processing often requires specialized computational architectures.

Field Programmable Gate Arrays (FPGAs), due to their highly customizable hardware design, present themselves as suitable candidates for these computational tasks. Their design allows for the exploitation of sparse neuron firing patterns. The core Digital Signal Processor (DSP) slices are engineered to

* This work was supported by Key Digital Technologies Joint Undertaking (KDT JU) in EdgeAI "Edge AI Technologies for Optimised Performance Embedded Processing" project, grant agreement No 101097300.


DOI: https://doi.org/10.1007/978-3-031-62874-0_2

**Table 1.** A comparative summary of relevant works in sEMG classification using Spiking Neural Networks.

| Work | Dataset | Encoding | Classess | Accuracy | Device | Energy | Power | Mops |
|---|---|---|---|---|---|---|---|---|
| [3] | custom 4 subjects | Delta | 4 | 84.8% | SpiNNaker | N.R[1] | 1-4 W | N.R. |
| [4] | custom 8 subjects | Population | 8 | 97.4% | N.R. | N.R. | N.R. | 0.013 |
| [10] | custom 5 subjects | HD-sEMG Decomposition | 10 | 95% | Jetson | 0.97 mJ | 100 mW | 7.97* |
| [12] | custom 10 subjects | Event-Differential | 6 | 98.78% | N.R. | N.R. | N.R. | 6.57 |
| [11] | NinaPro DB5 | Delta | 12 | 74% | Loihi | 246 mJ | 41 mW | 11.56* |
| **Our** | **NinaPro DB5** | **Delta** | **12** | **85.6%** | **FPGA** | **35.68 µJ** | **1.7 mW** | **2.336** |

[1] Not Reported.
* Estimated from the paper.

adeptly handle a suite of arithmetic operations including addition, multiplication, and multiply-and-accumulate. On another front, BRAM (Block Random Access Memory) units, due to their adaptable design and size, are especially suitable for integrating SNN models and facilitating data access and management.

SNN-based systems, although an emerging technology, show promising potential in the classification of sEMG signals for hand gesture recognition, a key step toward the advancement of complex prosthetic devices and human-machine interfaces. This paper presents a real-time sEMG classification system for distinguishing hand gestures and investigates the integration of Spiking neural networks on the Lattice iCE40-UltraPlus FPGA, a device optimized for low-power applications, addressing the challenges and potentials in this area.

The main contributions can be summarized as three most relevant points:

- we present an SNN-based real-time classification system capable of discerning twelve distinct hand gestures using sEMG signals, achieving an accuracy of 85.6% that underscores the effectiveness of the model;
- we investigate the most efficient coding method to transmit the continuous sEMG signal in spike traces, evaluating the benefits of using delta modulation on the original signal, as well as on the first- and second-order derivatives, which enhances classification performance;
- we illustrate an efficient implementation on ultra-low-power Lattice FPGAs, consuming about 1 mW, which exploits the inherent energy efficiency of SNNs for low-power classification.

## 2 Related work

Recent works from the literature demonstrate the efficiency of SNNs and their suitability for the gesture recognition problem based on the sEMG signal processing. In Table 1, we summarize the most relevant contributions, reporting

the reference dataset, the data encoding scheme, the number of different gestures considered, and the classification accuracy. Moreover, we include the on-hardware efficiency of the proposed models, reporting the targeted platform and the measured energy and power consumption for inference execution.

The work of [3] demonstrates the use of the NeuCube spiking model for the recognition of 4 different hand gestures, reaching 84.8% classification accuracy. The reference data was acquired from 4 volunteering subjects, performing two-digit grasp, three-digit grasp, fist, and hand rest. The train of spikes provided as input to the SNN model was obtained through temporal difference encoding. Similarly, the work of [12] presents an SNN model reaching 98.78% accuracy in the recognition of 6 different gestures, based on the custom recordings of 10 subjects, whereas in [10] 95% classification accuracy was obtained in the classification of 10 hand gestures. The authors of [4] propose a different encoding scheme, exploiting a set of Gaussian filters to produce the train of spikes. They report 97.4% accuracy in the recognition of 8 different gestures based on the sEMG recorded from 8 subjects. As can be noticed, the listed works reference custom datasets, comprising different sets of hand gestures, therefore the comparison of the reported results in terms of achieved performance is complicated by the lack of a common data reference [3, 4, 10, 12].

Due to this reason, we considered as our main reference the work of [11], which targeted the open-source NinaPro database, considering a subset of the data collected in the DB5, and particularly 12 gestures comprising flexion and extension of each finger, thumb adduction, and abduction, plus a rest class. The authors perform a design exploration for the selection of the optimal SNN model, comparing topologies including convolutional and fully connected layers. The finally selected model exploits the CUBA leaky-integrate and fire neuron and embeds two fully connected layers, reaching 74% classification accuracy. With this work, we aim at improving the classification performance achieved in [11], proposing a lean topology with a reduced number of required operations and relying on a novel version of the delta encoding scheme, where we include the information provided by the first and second derivatives of the signal.

Moreover, considering the applications of sEMG processing, which include prosthesis control, evaluating the efficiency of the proposed classifier is especially relevant. The SNN model presented in [3] was deployed on the SpiNNaker platform [6], providing real-time classification with power consumption ranging from 1 to 4 W. The model presented in [11] was evaluated on the Loihi [5], demonstrating 5.7 ms inference time, with 246 mJ energy per inference and 41 mW average power consumption for parallel execution on 12 cores. On the other hand, the model presented in [10] was deployed on the NVIDIA Jetson Nano platform, resulting in 9.7 ms inference time, with 0.97 mJ per inference and 100 mW average power consumption. Finally, direct measurements on target hardware are not provided for the models in [12] and [4], we thus report their complexity in terms of the number of required accumulate operations. To the best of our knowledge, the implementation presented for our proposed gesture recognition model results in the lowest power consumption among the referenced

works, thanks to the specialized re-configurable processing architecture designed on the low-power Lattice FPGA and a limited network complexity in terms of the number of required operations.

## 3 Method

In this section, we delineate the specifics of our proposed system, showcasing the reference dataset, the encoding methodology for spike generation, the chosen SNN architecture, and the targeted FPGA-based processor adapted to sEMG signal processing.

### 3.1 Dataset

The data used in this study were derived from the NinaPro DB5 dataset [8], which includes sEMG and kinematic data collected from 10 subjects without amputations or disabilities performing 52 distinct hand movements alongside a resting position. The 52 gestures are divided into subgroups, E1, E2, and E3, in alignment with the work of [11], which provides the main reference for comparison, we selected the E1 subgroup comprising of 12 exercises. The labels are structured to capture different movements for each finger. For the index, middle, ring, and little fingers, there are two labels each, one for flexion and another for extension, making eight labels. The thumb adds four more labels for adduction, abduction, flexion, and extension.

Acquisition was performed using two Thalmic Myo Armbands and the sEMG signals were sampled at a frequency of 200 Hz. Each subject performed six repetitions of the specified movements, as demonstrated by movies on a laptop computer screen, with each repetition lasting approximately five seconds followed by a three-second rest interval.

For our analysis, we focused specifically on the sEMG data encapsulated in the "emg" variable of the dataset, which consists of 16 columns representing signals from 16 channels. The first 8 columns correspond to electrodes evenly distributed around the forearm at the height of the radio-humeral joint, while columns 9-16 represent the second Myo, with a 22.5-degree clockwise slope.

The labels are derived from the "restimulus" variable. While, as will be further explained in Section 3.3, the dataset was divided into training, validation, and testing subsets based on the "rerepetition" variable. The names "restimulus" and "rerepetitin" for the variables derive from the original dataset description.

### 3.2 Encoding

In the context of spiking neural networks, an important part of the process is to transform continuous sEMG signals into spike sequences. In this study, we use Delta modulation, a method commonly used in signal processing and telecommunications to transform analog signals into digital form.

Our approach is different from the usual methods found in the literature because we apply delta modulation not only to the raw signal but also to its first and second derivatives. Using the first derivative, we emphasize information related to the speed of muscle activations. Meanwhile, the second derivative helps us analyze the acceleration or deceleration of these activations, adding more depth to our analysis.

The mechanism of delta modulation generates two distinct traces, indicating either a positive or negative deviation of the signal from a designated threshold, δ, which in our setup, is assigned a value of 15. The signal traces obtained through delta modulation are used directly without any normalization. Through a series of tests, the δ value chosen was validated to effectively represent variations in the sEMG signal. The encoding method is described as pseudo-code in Algorithm 1.

**Algorithm 1:** sEMG Encoding with Delta Modulation

```
   Input: sEMG signal
   Result: Two signals of spikes per channel, POS and NEG.
 1 delta sample = First sEMG sample;
 2 delta value;
 3 while sEMG signal do
 4     for sEMG channel do
 5         sEMG sample;
 6         if sEMG sample > delta sample + delta value then
 7             delta sample = sEMG sample;
 8             POS_channel.append(spike);
 9         else
10             POS_channel.append(no-spike);
11         if sEMG sample < delta sample - delta value then
12             delta sample = sEMG sample;
13             NEG_channel.append(spike);
14         else
15             NEG_channel.append(no-spike);
```

### 3.3 Segmentation

The segmentation process is a crucial step in preparing sEMG data for subsequent analysis and classification activities. This step involves sectioning continuous sEMG signals into segments or windows, which are then used for feature extraction and classification.

Initially, the repetitions within the dataset are allocated to training, validation, and testing subsets. Specifically, repetitions 1, 2, 4, and 6 are earmarked for training, repetition 3 for validation, and repetition 5 for testing.

The windowing process, a central aspect of segmentation, is governed by a set of parameters. Primarily, the window size and shift, which are set at 0.5 seconds and 0.1 seconds respectively, dictate the extent and overlap of the segments. Given the data frequency of 200 Hz, these time-based parameters are translated

into sample-based metrics, yielding a window size of 100 samples and a window shift of 20 samples.

In the NinaPro DB5, each sample is originally labeled as either *exercise* or *rest*. To assign a single label to a window of samples, we established specific criteria based on the proportion of labels within each window. A window is classified as *exercise* if at least 80% of its samples are labeled as such. Conversely, a window is labeled as *rest* if 100% of its samples carry the *rest* label. Windows that do not conform to these criteria are excluded from the training phase.

To eliminate some noise present at the beginning of the signal, we imposed an initial delay of 2 seconds before the start of the windowing process, equivalent to 400 samples based on the data sampling rate.

### 3.4 SNN topology and training

We used the SLAYER [9] libraries to manage the neural network; as a matter of practical implementation, the selected neuron type is the LIF (Leaky Integrate and Fire).

Outlined in Equations 1 and 2, each neuron receives a sequence of spikes $s_{in}$ as input, which, when subject to synaptic weight $w$ multiplication, shape the membrane voltage $v$ within the neuron. The voltage $v$ gradually decreases, influenced by the decay factor $\alpha$.

The SLAYER toolbox is essential for defining custom spiking neuron and synaptic behaviors. It was used in conjunction with the LAVA framework[2, 1], which enhances GPU-based efficiency for SNN training and simulation.

$$\tilde{v}(t) = \alpha \cdot v(t) + \sum w \cdot s_{in}(t) \tag{1}$$

$$v(t+1) = \begin{cases} \tilde{v}(t), & \text{if } \tilde{v}(t) < \theta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

As shown in Equation 3, the neuron emits an output spike $s_{out}(t)$ when its voltage exceeds a certain threshold $\theta$. The output spike can be represented as:

$$s_{out}(t+1) = \begin{cases} 1, & \text{if } \tilde{v}(t) \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

We carried out the training on Google Colab, leveraging the computational power of T4 GPUs. The Adam optimizer was chosen with a learning rate of 0.001 and batch sizes fixed at 32. The loss was assessed through the *slayer.loss* class, with a focus on *SpikeRate*-based calculations. We set a true rate target at 0.2 and a false rate at 0.03. To avoid overfitting, an early stopping technique was used. In our case, training is stopped if there is no improvement within 10 consecutive epochs.

The architecture of our network is assembled through a series of Dense layers, delineated in Table 2. Finally, through the support of the SLAYER [9] framework, we introduced axonal delays to simulate the propagation time of spikes along the axon, setting a maximum delay value of 62 time-steps.

**Table 2.** Topology and parameters of the proposed SNN.

| Layer | Synapse | Neuron | Delay |
|---|---|---|---|
| Dense Layer 1 | 96 | 64 | True |
| Dense Layer 2 | 64 | 128 | True |
| Dense Layer 3 | 128 | 64 | True |
| Dense Layer 4 | 64 | 13 | False |

### 3.5 System overview

Figure 1 illustrates the schematic representation of the system implemented. The core control tasks are orchestrated by a RISC-V processor.

The input signal acquired through the SPI interface is first processed through the *encoding* stage, where it is translated into spike trains using delta modulation, as detailed in Section 3.2. At this point, the dataflow progresses to the *segmentation* block, which defines a partitioning of the data into windows, as described in Section 3.3.

Thereafter, the segmented spike trains are channeled into the SNN processor. The processor analyzes the spike train inputs to extract meaningful information about the gestures represented by the sEMG signals. After each inference, a *voting* phase is initiated to examine the neural network outputs and determine the final classification.

The classification output is then propagated through the UART interface, concluding the structured data processing path from acquisition to classification.

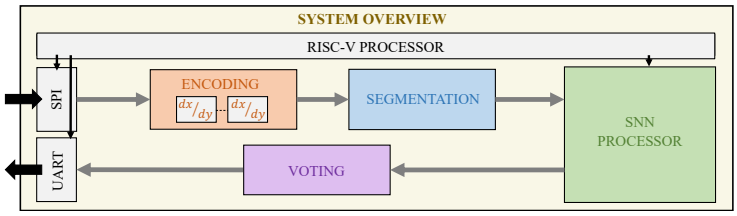


**Fig. 1.** Overview of the ECG monitoring system.

Figure 2 outlines the architecture of the SNN processor utilized in our system, drawing inspiration from the approach delineated in [7]. The input data is represented by the spike trains channeled from the encoding module. The processor hosts two *neuron* modules, each capable of accumulating four 8-bit synaptic weights per cycle, aiding in the computation between synapses and weights.

Both neuron modules are interfaced with a weight memory and a spike memory. The weight memory, populated during initialization with the network weights derived from training, and the spike memory, which stores the results of previous inferences for all neurons in the SNN, provide the data needed for the

calculation. An *address generator* dynamically computes the memory addresses, assisting each of the two available modules in processing a distinct layer of the SNN.

A *spike stack* works with the address generator to point out where the active spikes are in the memory, helping to skip over the inactive synapses. The spikes that come out of this process are sent to the voting module to be counted, determining the final classification.

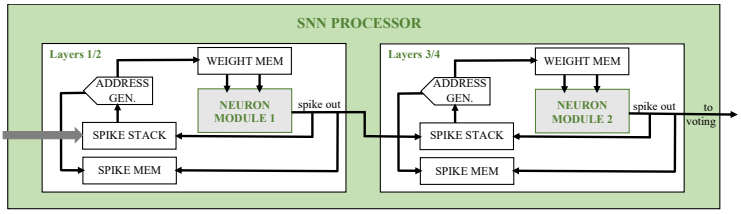


**Fig. 2.** Overview of the ECG monitoring system.

## 4 Experimental results

This section delineates the findings from our experimental analysis, evaluating both the classification accuracy and the on-hardware inference efficiency of the proposed sEMG monitoring system.

### 4.1 Classification evaluation

To reduce the memory footprint and improve the efficiency of the designated hardware system, the chosen model was subjected to quantization, reducing the resolution of the weights to 8 bits with negligible accuracy drop. This approach resulted in a classification accuracy of 85.6% on the test set, obtained with an output post-processing stage, which allows us to filter out single-spot classification, through a voting mechanism that discards classifications resulting in non-consecutive labels, by restoring the last valid output classification.

Table 3 represents the confusion matrix, providing a detailed analysis of classification performance across classes. As can be noticed, a relevant portion of the errors, around 45%, is located in the first column and the first row of the matrix, thus involving the distinction between the rest class and all the remaining gestures.

In this regard, we also evaluated the impact of the classification errors registered during the start of each gesture, knowing that the first windows include a certain portion of *rest* samples and *gesture* samples. In detail, we introduced a tolerance of 200 ms around the gesture onset, where we considered as acceptable classifications both the *rest* and the specific *gesture* class. Considering this additional tolerance, the overall classification accuracy achieved is 87.01%.

A graphical representation of an example of classification output is provided in Figure 3. The plot reports two repetitions of the ring flexion and extension gestures, separated by a resting phase. The classification output is indicated on the plot, and placed according to the predicted class. As outlined in Section 3.3, dataset repetitions are assigned to training (repetitions 1, 2, 4, 6), validation (repetition 3), and testing subsets (repetition 5).

**Table 3.** Confusion matrix reporting classification performance on the test set.

| True Labels \ Predicted Labels | Rest | Idx Flx | Idx Ext | Mid Flx | Mid Ext | Ring Flx | Ring Ext | Lit Flx | Lit Ext | Thm Add | Thm Abd | Thm Flx | Thm Ext |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Rest** | 7078 | 36 | 1 | 28 | 43 | 1 | 4 | 26 | 1 | 39 | 86 | 2 | 32 |
| **Idx Flx** | 62 | 268 | 30 | 15 | 21 | 5 | 17 | 11 | 0 | 0 | 2 | 2 | 2 |
| **Idx Ext** | 31 | 23 | 224 | 8 | 12 | 3 | 1 | 0 | 0 | 0 | 2 | 0 | 0 |
| **Mid Flx** | 30 | 31 | 3 | 285 | 14 | 7 | 8 | 13 | 3 | 3 | 0 | 0 | 0 |
| **Mid Ext** | 17 | 4 | 3 | 0 | 244 | 8 | 4 | 5 | 7 | 0 | 2 | 0 | 0 |
| **Ring Flx** | 43 | 8 | 2 | 12 | 12 | 258 | 2 | 14 | 2 | 3 | 0 | 2 | 2 |
| **Ring Ext** | 39 | 30 | 3 | 17 | 36 | 8 | 218 | 7 | 3 | 2 | 0 | 0 | 10 |
| **Lit Flx** | 48 | 5 | 5 | 6 | 13 | 18 | 26 | 237 | 7 | 10 | 6 | 3 | 11 |
| **Lit Ext** | 52 | 0 | 0 | 5 | 5 | 10 | 13 | 13 | 253 | 0 | 13 | 7 | 0 |
| **Thm Add** | 7 | 10 | 0 | 0 | 0 | 0 | 10 | 4 | 0 | 161 | 6 | 70 | 7 |
| **Thm Abd** | 71 | 2 | 4 | 0 | 0 | 0 | 2 | 3 | 3 | 7 | 190 | 17 | 19 |
| **Thm Flx** | 21 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 64 | 2 | 166 | 10 |
| **Thm Ext** | 32 | 0 | 10 | 2 | 2 | 0 | 0 | 7 | 2 | 9 | 3 | 6 | 252 |

### 4.2 Sparsity

SNNs naturally exhibit sparsity, since in a specific instance only a small subset of neurons are engaged in firing or communication. Sparsity is defined as the ratio of inactive spikes to the maximum possible spike count within the network, reflecting the network's level of inactivity or quietness. In our context, taking the test set into consideration, a sparsity value of 90.99% was calculated, indicative of a mere 9% average presence of potential active spikes. The platform's processing elements allow inference to be calculated for four peaks in parallel. As a consequence, spikes are processed in quartets, therefore it becomes critical to assess the extent to which the inherent sparsity of the model architecture can be exploited on the designated hardware and translated into performance efficiency. As a result of such grouping of spikes, it is necessary to introduce a new element, called *sparsity-hw*, which is defined by the Equation 4.

$$Sparsity\text{-}hw = \left(1 - \frac{\sum_{i=1}^{L} \sum_{j=1}^{G_i} g_{ij}}{\sum_{i=1}^{L} G_i}\right) \times 100 \tag{4}$$

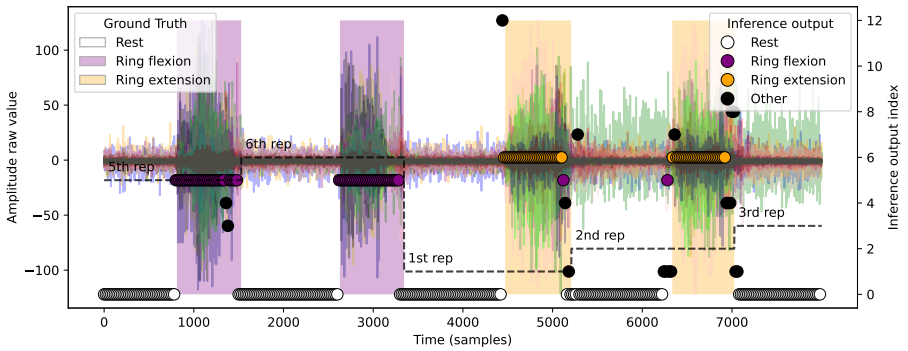


**Fig. 3.** Overview of the EMG signal classification for different repetitions of the rest, ring flexion, and extension gestures. The ground truth is highlighted in the background, whereas the classification output is reported as spots overlapped to the signal plot. Classification errors are reported in black.

Where:

- $L$ is the total number of layers in the network,
- $G_i$ is the number of spike groups in the i-th layer,
- $g_{ij}$ indicates whether there's at least one active spike in the j-th group in the i-th layer (1 if at least one spike is active, 0 otherwise).

In our scenario, a *sparsity-hw* value of 77.11% was achieved.

### 4.3 Power consumption

The system mainly switches between two operating states: the active inference phase and the idle phase. In the active inference phase, the FPGA is actively engaged in analyzing the spike data through the Spiking Neural Network (SNN), classifying the sEMG signals into the corresponding gesture categories. In the idle phase, on the other hand, the system goes into a low-power mode, greatly reducing power consumption. The average power consumption of the system can be assessed over the recurrent 100 ms inference interval. During the active phase, which includes inference, data acquisition, and encoding time and lasts for 3.921 ms, the power consumption is equal to 12.011 mW. In contrast, during the idle state, the power consumption drops to 1.288 mW. The average power over the 100 ms interval can be computed using Equation 5:

$$P_{\text{average}} = \frac{(P_{\text{active}} \cdot T_{\text{active}}) + (P_{\text{idle}} \cdot T_{\text{idle}})}{T_{\text{total}}} = 1.708 \text{ mW} \tag{5}$$

To acquire power metrics, we used a Digilent Analog Discovery 2 oscilloscope in conjunction with three shunt resistors of $1.0 \pm 0.01\ \Omega$ to measure power metrics. These resistors were installed across the three distinct power inputs on the

Lattice iCE40UP5k FPGA, identified as Vcore, VCCIO0&1, and VCCIO2. The internal components of the FPGA receive a 1.2 V supply from Vcore, whereas the I/O pins are powered by 3.3 V from the VCCIO0&1 and VCCIO2 circuits.

### 4.4 Discussion

Our system, implemented on an FPGA, demonstrates a significant advantage in terms of power efficiency, consuming about 1.7 mW. This is substantially less than those mentioned in Section 2, such as [3] with a range of 1-4 W and [10] with 100 mW on Loihi.

Focusing on the NinaPro DB5 dataset, which is common between our work and [11], provides a fair ground for performance comparison. In terms of accuracy, our system achieves 85.6%, which is notably higher than the 74% reported by [11]. Nonetheless, the model proposed in this work requires the execution of 2.336 million of accumulate operations (MOPS) per classification, which is lower than the 11.56 MOPS estimated for the execution of the SNN proposed by [11].

In summary, our work stands out in terms of power efficiency and accuracy, representing a robust solution for real-time gesture recognition using sEMG data. While there is a trade-off in terms of OPS, the significant gains in accuracy and power efficiency underscore the effectiveness of our approach.

## 5 Conclusion

We introduced a real-time sEMG signal classification system aimed at the precise identification of hand gestures. By employing a Spiking Neural Network (SNN) for classification, the work taps into the inherent efficiency of event-based computation, providing a low-power solution apt for implementation on edge devices. The proposed model, when evaluated on the Ninapro DB5 dataset, exhibited a commendable classification accuracy of 85.6%, thereby substantiating the effectiveness of the SNN in recognizing twelve distinct hand gestures. The architecture, tailored for the Lattice iCE40-UltraPlus FPGA, displayed remarkable efficiency in terms of both computation and power consumption. The quantized version of the model, aimed at reducing the memory footprint and enhancing the operational efficiency on the hardware, maintained a high classification accuracy, showcasing the feasibility of deploying such models on resource-constrained platforms. Power examination demonstrated an average power consumption of 1.7 mW, underscoring the energy-efficient characteristic of the proposed system. This work, therefore, lays a solid foundation for the creation of energy-efficient and effective real-time sEMG-based gesture recognition systems, paving the way for advanced human-machine interfaces and prosthetic control.

## References

1. Lava-DL SLAYER (2023), https://lava-nc.org/lava-lib-di/slayer/slayer.html, accessed: 2023-1-8

2. LAVA-NC (2023), https://lava-nc.org/index.html, accessed: 2023-1-8
3. Behrenbeck, J., Tayeb, Z., Bhiri, C., Richter, C., Rhodes, O., Kasabov, N., Espinosa-Ramos, J.I., Furber, S., Cheng, G., Conradt, J.: Classification and regression of spatio-temporal signals using neucube and its realization on spinnaker neuromorphic hardware. Journal of Neural Engineering **16**(2), 026014 (feb 2019). https://doi.org/10.1088/1741-2552/aafabc, https://dx.doi.org/10.1088/1741-2552/aafabc
4. Cheng, L., Liu, Y., Hou, Z.G., Tan, M., Du, D., Fei, M.: A rapid spiking neural network approach with an application on hand gesture recognition. IEEE Transactions on Cognitive and Developmental Systems **13**(1), 151–161 (2021). https://doi.org/10.1109/TCDS.2019.2918228
5. Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G.A.F., Joshi, P., Plank, P., Risbud, S.R.: Advancing neuromorphic computing with loihi: A survey of results and outlook. Proceedings of the IEEE **109**(5), 911–934 (2021). https://doi.org/10.1109/JPROC.2021.3067593
6. Furber, S.B., Lester, D.R., Plana, L.A., Garside, J.D., Painkras, E., Temple, S., Brown, A.D.: Overview of the spinnaker system architecture. IEEE Transactions on Computers **62**(12), 2454–2467 (2013). https://doi.org/10.1109/TC.2012.142
7. Leone, G., Raffo, L., Meloni, P.: On-fpga spiking neural networks for end-to-end neural decoding. IEEE Access **11**, 41387–41399 (2023). https://doi.org/10.1109/ACCESS.2023.3269598
8. Pizzolato, S., Tagliapietra, L., Cognolato, M., Reggiani, M., Müller, H., Atzori, M.: Comparison of six electromyography acquisition setups on hand movement classification tasks. PLoS ONE **12** (2017), https://doi.org/10.1371/journal.pone.0186132
9. Shrestha, S.B., Orchard, G.: Slayer: Spike layer error reassignment in time. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. p. 1419–1428. NIPS'18, Curran Associates Inc., Red Hook, NY, USA (2018)
10. Tanzarella, S., Iacono, M., Donati, E., Farina, D., Bartolozzi, C.: Neuromorphic decoding of spinal motor neuron behaviour during natural hand movements for a new generation of wearable neural interfaces. IEEE Transactions on Neural Systems and Rehabilitation Engineering **31**, 3035–3046 (2023). https://doi.org/10.1109/TNSRE.2023.3295658
11. Vitale, A., Donati, E., Germann, R., Magno, M.: Neuromorphic edge computing for biomedical applications: Gesture classification using emg signals. IEEE Sensors Journal **22**(20), 19490–19499 (2022). https://doi.org/10.1109/JSEN.2022.3194678
12. Xu, M., Chen, X., Sun, A., Zhang, X., Chen, X.: A novel event-driven spiking convolutional neural network for electromyography pattern recognition. IEEE Transactions on Biomedical Engineering **70**(9), 2604–2615 (2023). https://doi.org/10.1109/TBME.2023.3258606