

Article

Hardware-Native Hyperdimensional Computing for Lower-Limb EMG Classification on a Resource-Constrained GateMate FPGA

Krischan Ledwig * , Abdelrahman Noshay Abdelalim Ahmed and Dietmar Fey 

Chair of Computer Science 3 (Computer Architecture), Friedrich-Alexander-Universität Erlangen-Nürnberg, 91058 Erlangen, Germany; abdelrahman.ahmed@fau.de (A.N.A.A.); dietmar.fey@fau.de (D.F.)

* Correspondence: krischan.ledwig@fau.de

Abstract

Electromyography (EMG)-controlled orthoses and wearable assistive systems require classifiers that combine accurate motion recognition, efficient embedded deployment, and support for local model formation. Long-term EMG deployment is affected by signal drift and inter-session variability, motivating architectures that can support post-deployment model updates. Local processing can reduce dependence on external computation and data transfer. To address these challenges, this work presents a hardware-targeted Hyperdimensional Computing (HDC) classifier for trainable EMG classification on the resource-constrained GateMate A1 FPGA from Cologne Chip. The proposed architecture performs on-device HDC model formation and inference directly on FPGA. Linear Discriminant Analysis (LDA) serves as a conventional offline-trained and FPGA-inferred baseline. The evaluation uses eight anonymized single-session EMG recordings from seven healthy participants and one participant with spinal cord injury with 32 channels and five motion classes and includes accuracy, robustness, and routed FPGA implementation analysis. Across all recordings, the per-recording best-case HDC configurations reach 95.20% classification accuracy, while the LDA baseline achieves 98.54% overall accuracy. Under controlled input perturbation with a standard deviation of $\sigma = 0.10$, HDC retains 92.98% mean accuracy compared with 80.98% for LDA. A first board-level power measurement indicates an energy cost of approximately 0.40 mJ per inference sample and 0.87 mJ per training sample. The experiments demonstrate single-session on-device HDC model formation and inference, while longitudinal validation, fatigue robustness, electrode-shift robustness, and inter-session adaptation remain future work. The results indicate that HDC provides an architectural foundation for trainable wearable edge-AI systems with local model updates.

Keywords: hyperdimensional computing; EMG classification; on-device model formation; FPGA; GateMate A1; hardware training; robustness; model-memory faults; wearable edge AI



Academic Editors: Josip Musić,
Marina Prvan and Duje Čoko

Received: 15 June 2026

Revised: 3 July 2026

Accepted: 7 July 2026

Published: 9 July 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).

1. Introduction

EMG-controlled orthoses and wearable assistive systems require embedded classifiers that combine low latency, resource-constrained operation, and robustness against perspiration, fatigue, and day-to-day signal variability [1–3]. Surface EMG signals are inherently non-stationary because muscle fatigue, skin impedance changes, and user-specific activation patterns alter the observed feature distribution [2,4,5]. In wearable use, sweating and changing skin hydration can further modify electrode–skin coupling, contact impedance, and motion artifacts, especially during longer sessions or daily activities [5,6]. These effects

can shift feature distributions even when the intended movement class remains unchanged. As a result, practical systems may benefit from architectures that can support local model updates close to the sensing hardware, although such longitudinal adaptation must be evaluated separately.

Most embedded EMG systems perform training offline and deploy only the inference model to the target device [2,3]. While efficient, this approach limits local model updates after deployment. Orthotic applications are particularly affected because user-specific activation patterns and long-term drift can reduce classification performance over time. Local model formation can reduce repeated transfer of raw or feature-level biosignals to companion devices or remote services, potentially reducing communication overhead and associated data-transfer energy while simplifying security assumptions by limiting communication to configuration, status, or aggregated results.

This work evaluates Hyperdimensional Computing (HDC), also known as Vector-Symbolic Computing, as a hardware-native AI paradigm for trainable EMG classification on a resource-constrained FPGA [7,8]. In this work, hardware-native and trainable HDC refers to an implementation in which the architecture, memory organization, datapath segmentation, and execution flow are co-optimized for the GateMate FPGA while supporting on-device formation of associative-memory class prototypes. The complete workflow comprises feature encoding, model formation, prototype storage, and inference within the same reconfigurable hardware fabric after initialization of fixed encoder memories. HDC operations such as XOR binding, majority bundling, and Hamming-distance search map efficiently to digital hardware and enable training, model formation, storage, and inference to share a common binary representation suitable for edge deployment under strict resource constraints [8–10]. This is in marked contrast to classical statistical classifiers such as Linear Discriminant Analysis, whose training relies on floating-point covariance estimation and matrix inversion—operations that are disproportionately costly to implement on a small, DSP-scarce FPGA such as the GateMate A1. HDC’s reliance on boolean operators and integer accumulation instead of floating-point arithmetic is therefore not merely an implementation choice but the property that makes on-device model formation feasible on such resource-constrained hardware in the first place. Despite this hardware-level limitation, LDA remains a well-established and extensively validated method for EMG pattern recognition [2,11,12], and this work uses it as a compact, thoroughly studied reference baseline for comparison rather than as a direct architectural competitor to the proposed HDC implementation.

The proposed implementation targets the GateMate A1 FPGA from Cologne Chip [13] as a compact resource-constrained deployment platform. Such constraints are particularly relevant for wearable orthotic systems, where local processing capability must be balanced against energy consumption, memory capacity, and implementation complexity [14]. By executing feature encoding, model formation, storage, and inference locally, the architecture reduces dependence on continuous external processing and network availability. The presented architecture therefore evaluates whether this complete HDC workflow can be executed directly on a compact FPGA platform without relying on external training infrastructure. This work is positioned as a computer-architecture and hardware-feasibility study rather than a clinical investigation, prioritizing architectural coverage—parameter sweeps, robustness, and routed FPGA implementation—over recruiting a large, clinically powered participant cohort.

The main contributions of this work are:

- A hardware-targeted HDC architecture enabling on-device model formation and inference for trainable EMG classification on the resource-constrained GateMate A1 FPGA.

- A dedicated robustness and sensitivity analysis quantifying classification accuracy under Gaussian noise injected independently into test-time inputs and into stored associative-memory prototypes, together with a systematic evaluation of training-data fraction on classification performance.
- Resource-aware FPGA implementation analysis covering routed utilization, timing behavior, and architecture–algorithm trade-offs for edge–AI deployment.
- Experimental evaluation across eight EMG recordings from seven healthy participants and one participant with spinal cord injury, including parameter sweeps and baseline comparison.

To position the proposed architecture relative to the existing EMG, HDC, and FPGA literature, the following section reviews related approaches and identifies the addressed research gap.

2. Related Work

2.1. EMG Pattern Recognition and Embedded Wearable Systems

EMG-based pattern recognition has been extensively investigated for prosthetic, orthotic, and wearable assistance applications. Classical approaches typically combine window-based feature extraction with compact classifiers such as Linear Discriminant Analysis (LDA), support vector machines, and related low-complexity methods because these approaches provide predictable execution behavior and efficient deployment on constrained platforms [2,11,12].

Despite strong classification performance under controlled conditions, practical deployment remains challenging because EMG signals exhibit non-stationary behavior caused by electrode displacement, muscle fatigue, skin impedance variation, and user adaptation. Consequently, wearable EMG systems increasingly emphasize robustness, personalization, and reduced dependence on external computation.

Recent embedded EMG studies explored embedded processing and adaptation-oriented strategies to improve robustness, personalization, and long-term usability [3]. However, many practical systems continue to separate model formation from deployment by performing training externally and executing only inference on the target hardware. While this design simplifies deployment, it limits local model updates directly within wearable systems.

2.2. HDC for EMG and Biosignal Learning

Hyperdimensional Computing has emerged as a hardware-oriented learning paradigm based on high-dimensional distributed representations and simple binary operations [7,8]. Because HDC predominantly relies on binary logic, memory access, and population-count operations, it is frequently considered suitable for embedded and potentially energy-efficient implementations.

For biosignal processing, Rahimi et al. demonstrated HDC-based EMG gesture recognition using distributed binary representations and similarity search [15]. Benatti et al. further showed that HDC can support low-power embedded EMG classification with online updates [16]. Moin et al. further demonstrated an EMG gesture-recognition system that combines flexible high-density EMG sensors with a brain-inspired HDC classifier, supporting the relevance of HDC for EMG variability, electrode-dependent signal changes, and user-specific personalization [17]. More recent work extended incremental learning capabilities beyond static deployment. Zhou et al. explored incremental learning across changing limb positions for EMG gesture recognition and reported indications that HDC can maintain classification performance under distribution shifts without complete re-training [18]. Beyond EMG-specific applications, OnlineHD and related HDC approaches

further explored efficient online and single-pass learning procedures for scalable model updates and hardware-oriented deployment [19,20].

These studies demonstrate that HDC supports efficient adaptation-oriented workflows and biosignal learning. However, most contributions primarily evaluate algorithmic behavior or processor-based execution rather than complete trainable FPGA implementations with explicit memory organization and routed implementation constraints.

2.3. HDC Hardware Acceleration and FPGA Deployment

Prior FPGA-based HDC accelerators have primarily targeted larger, performance-oriented FPGA platforms and have focused on inference acceleration rather than on-device model formation [9,10,21,22]. Comparatively little attention has been given to executing a complete, trainable HDC pipeline on a genuinely low-power, cost-optimized FPGA suitable for battery-powered wearable deployment. Moreover, these studies typically report a single or narrow operating point rather than a systematic evaluation across robustness, training-fraction sensitivity, and diverse resource–architecture configurations, which limits their ability to characterize practical accuracy–resource–timing trade-offs. Together, these works demonstrate the relevance of HDC for biosignal learning and its suitability for hardware acceleration. However, they do not fully address the specific combination targeted in this work: lower-limb EMG classification with on-device HDC model formation and inference implemented as a complete FPGA hardware pipeline on a low-power, resource-constrained device under practical resource and routing constraints, together with extensive robustness and training-sensitivity evaluation. In particular, prior HDC-based EMG studies have mainly focused on gesture recognition and upper-limb applications, whereas this work evaluates ankle-motion classification for wearable foot-orthosis-oriented scenarios. Table 1 summarizes this positioning relative to representative prior work.

Table 1. Positioning of this work relative to selected prior work with respect to EMG application, HDC usage, target hardware, trainability, hardware-oriented evaluation, and support for complete hardware execution pipelines. Here, complete hardware pipeline denotes that feature encoding, model formation, model storage, and inference are executed within dedicated hardware fabric and excludes processor-based execution of training or inference workflows.

Work	EMG	HDC	Execution Platform	Trainable	HW Eval.	Complete HW Pipeline
Rahimi et al. [15]	✓	✓	–	–	–	–
Benatti et al. [16]	✓	✓	PULP SoC	✓	✓	–
Zhou et al. [18]	✓	✓	SW	✓	–	–
OnlineHD/LookHD [19,20]	–	✓	SW	✓	–	–
F5-HD/FACH [9,10]	–	✓	FPGA	–	✓	–
HD2FPGA/AeneasHDC [21,22]	–	✓	FPGA	–	✓	–
HDBinaryCore [23]	✓	✓	ASIC	–	✓	–
This work	✓	✓	FPGA	✓	✓	✓

To evaluate the identified research gap, the following section introduces the EMG dataset, HDC formulation, hardware architecture, and experimental methodology.

3. Materials and Methods

3.1. EMG Dataset and Preprocessing

Surface EMG data were collected from eight participants (P1–P8, age range 20–65 years), including seven healthy participants and one participant with spinal cord injury. The participant with spinal cord injury is represented by Recording 04 in the later evaluation and is included to provide an initial indication of how hardware-native model formation behaves under impaired motor activation patterns. Two participants were female and six were male. All participants provided written informed consent. The study was conducted

in accordance with the Declaration of Helsinki and approved by the Ethics Committee of Friedrich-Alexander-Universität Erlangen-Nürnberg (application 24-286-B).

Participants performed five ankle-motion classes: rest, dorsiflexion (up), plantarflexion (down), inversion (right), and eversion (left).

Surface EMG was acquired using a wearable 32-channel dry-electrode bracelet (HD20MM1602B, OT Bioelettronica, Turin, Italy) and a Muovi wireless EMG probe (OT Bioelettronica, Turin, Italy). The bracelet was placed over the muscle bellies on the upper part of the shin below the knee. The system sampled EMG signals at 2000 Hz and streamed data wirelessly via WiFi using a Transmission Control Protocol (TCP) to the host device at an approximately 111 Hz frame rate, delivering chunks of 18 samples per channel every approximately 9 ms. Data acquisition, preprocessing, and baseline model training were performed using the previously published open-source framework MyoGestic [24].

For each participant, the recording protocol consisted of one initial 10 s resting period followed by separate 10 s steady-state maximum voluntary contraction recordings for each movement class. Participants were instructed to reach a comfortable maximum ankle range of motion and maintain the contraction throughout the recording interval.

Each participant contributed one recording session, resulting in eight processed EMG recordings in total. The resulting dataset forms a supervised classification task with five motion classes and 32 RMS-based feature channels and was selected to represent wearable EMG acquisition conditions relevant to trainable orthotic control and future local model updates. Across all recordings, 34,744 samples were used for training and 8489 samples for testing. The window-level train–test split was approximately class-stratified within each recording. The recordings contain 4338–4350 training samples and 870–1090 test samples per recording, corresponding to approximately 870 training windows and 174–219 test windows per represented class. Because each recording is single-session, this split evaluates within-session model formation and inference rather than inter-session adaptation.

Although MyoGestic supports conventional machine learning pipelines, this work uses the recorded and preprocessed EMG features exclusively to evaluate hardware-native HDC training and inference as well as the FPGA-oriented LDA baseline.

The raw EMG signals are sampled at 2000 Hz. The continuous recordings are segmented into overlapping windows of $W = 504$ samples, corresponding to approximately 252 ms. Consecutive windows are shifted by 18 samples, corresponding to approximately 9 ms and an effective feature-update rate of approximately 111 Hz.

The train–test split follows the MyoGestic preprocessing procedure and is applied after window extraction. For each class recording, the first 80% of the windowed samples are used for training, while the final 20% are reserved for testing. With a window length of 504 samples and a hop size of 18 samples, adjacent windows overlap by 486 samples, corresponding to 243 ms or 96.4% of the window duration. However, this overlap is local in time: after 28 window shifts, the start positions are separated by a full window length and the raw-sample overlap vanishes. Therefore, only windows close to the train–test boundary can share raw samples across the split, while each class is evaluated on approximately 174–219 test feature vectors per recording. The reported classification results should consequently be interpreted as within-session window-level performance with locally correlated neighboring windows, rather than as inter-session generalization.

For each window, root-mean-square (RMS) features are extracted independently for all 32 channels. For channel i and a window of raw samples $s_i[t]$, the RMS feature is

$$x_i = \sqrt{\frac{1}{W} \sum_{t=1}^W s_i[t]^2}, \quad (1)$$

where x_i describes the local signal energy of the i -th EMG channel. RMS features are widely used in EMG pattern recognition because they provide a compact estimate of muscle activation and reduce sensitivity to sign changes in the raw signal [4,11,12], among other established EMG feature-extraction approaches such as wavelet-based methods [25]. The resulting feature vector is

$$\mathbf{x} = [x_1, x_2, \dots, x_N]^T, \quad N = 32. \quad (2)$$

After RMS feature extraction, the resulting feature values are standardized using z-score scaling derived from the training data. For HDC encoding, the scaled continuous feature values are then quantized into discrete levels using a range derived from the scaled training data. In the evaluated prototype, RMS extraction, z-score standardization, and feature quantization are performed in software on the host side before FPGA execution. The FPGA accelerator operates on the resulting fixed-point feature representation and does not implement RMS extraction or z-score standardization internally.

This quantization step is part of the evaluated implementation and enables mapping continuous EMG features to the discrete level-memory representation used by the hardware encoder. In alternative deployment scenarios where preprocessing already produces normalized features directly compatible with the encoder input range, explicit re-quantization may become unnecessary. In such cases, the fixed item and level memories remain unchanged and future model updates could be restricted to updating the associative-memory representation.

3.2. Hyperdimensional Computing Classifier

The HDC classifier maps each feature vector $\mathbf{x}$ into a high-dimensional binary hypervector. Let D be the hypervector dimension, M the number of quantization levels, $N = 32$ the number of EMG features, and $C = 5$ the number of classes. The implementation uses binary hypervectors in $\{0,1\}^D$ and record-based encoding, where each feature value is encoded together with its channel identity.

Each feature position i is assigned a fixed random item hypervector $\mathbf{I}_i \in \{0,1\}^D$. The quantization levels use correlated level hypervectors $\mathbf{L}_m \in \{0,1\}^D$ stored in the continuous item memory. The first level hypervector is generated randomly. Subsequent level hypervectors are derived by flipping a fixed number of pseudo-randomly selected bits from the previous level. Neighboring levels therefore remain more similar than distant levels, preserving the order of the feature-amplitude range in the hypervector space.

The quantization range is derived from the training data and represented as a symmetric feature range about zero, with its extent set by a magnitude derived from the training data, before conversion into hardware-compatible fixed-point values. Let this range be $[q_{\min}, q_{\max}]$, with $q_{\min} = -q_{\max}$. Each feature value x_i is first mapped to a clipped relative position

$$r_i = \text{clip}\left(\frac{x_i - q_{\min}}{q_{\max} - q_{\min}}, 0, 1\right). \quad (3)$$

The corresponding HDC level index is then computed as

$$q_i = \left\lfloor r_i(M-1) + \frac{1}{2} \right\rfloor, \quad (4)$$

where $q_i \in \{0, \dots, M-1\}$. This formulation describes the level mapping used by the encoder. In the hardware path, the clipped relative feature position is supplied in fixed-point form and converted to the same discrete level index.

The feature-specific encoded hypervector is generated by binding the item and level hypervectors:

$$\mathbf{E}_i = \mathbf{I}_i \oplus \mathbf{L}_{q_i}, \quad (5)$$

where $\oplus$ denotes bitwise XOR. The sample hypervector $\mathbf{Q}$ is obtained by bundling all encoded feature hypervectors using a component-wise majority operation:

$$Q_j = \begin{cases} 1, & \sum_{i=1}^N E_{i,j} \geq \tau, \\ 0, & \text{otherwise,} \end{cases} \quad j = 1, \dots, D, \quad (6)$$

with threshold $\tau = N/2$. Since $N = 32$ is even, a tie occurs when exactly 16 encoded feature hypervectors contain a one at bit position j . Following the deterministic convention used in prior HDC hardware work by Ge and Parhi [8], ties are resolved to one, because the implemented comparator uses $\sum_i E_{i,j} \geq 16$. For the binary HDC majority operation, resolving ties to zero or to one is an implementation convention as long as the rule is applied consistently. The selected deterministic rule avoids ambiguity in the implementation for the even number of input features and was kept fixed across all experiments.

During training, sample hypervectors belonging to class c are accumulated in class-specific counters:

$$A_{c,j} = \sum_{n:y_n=c} Q_{n,j}. \quad (7)$$

Here, y_n denotes the class label of the n -th training sample, and the sum runs over all training samples assigned to class c , so that $A_{c,j}$ counts how many of these samples have bit j set to one.

$$C_{c,j} = \begin{cases} 1, & A_{c,j} \geq \theta_c, \\ 0, & \text{otherwise,} \end{cases} \quad (8)$$

This formulation describes the algorithmic majority operation, where θ_c denotes the class-specific majority threshold derived from the number of training samples of class c .

For the implemented hardware realization, training accumulation is represented using signed counters that increment for one-valued query bits and decrement for zero-valued query bits. Final class-hypervector generation therefore becomes equivalent to thresholding the accumulated signed representation at zero while preserving the same binary model interpretation.

During inference, the query hypervector $\mathbf{Q}$ is compared with each stored class hypervector using the Hamming distance

$$d_H(\mathbf{Q}, \mathbf{C}_c) = \sum_{j=1}^D Q_j \oplus C_{c,j}. \quad (9)$$

The predicted class is

$$\hat{y}_{\text{HDC}} = \arg \min_c d_H(\mathbf{Q}, \mathbf{C}_c). \quad (10)$$

3.3. Linear Discriminant Analysis Baseline

Linear Discriminant Analysis (LDA) is used as a compact classical baseline for EMG classification. In contrast to the proposed HDC implementation, the LDA model is trained offline using the preprocessing and training workflow provided by MyoGestic [24] and only the inference step is considered for the FPGA-oriented comparison. LDA is therefore used as an established lightweight clean-accuracy and inference-oriented deployment reference, not as an equal-functionality on-device-training baseline. For implementation and algorithmic reference, the LDA training flow follows the formulation commonly used

in scikit-learn [26]. LDA assumes that all classes share a common covariance structure and represents each class by a linear discriminant function.

For an input feature vector $\mathbf{x}$, inference evaluates one score per class:

$$g_c(\mathbf{x}) = \mathbf{w}_c^T \mathbf{x} + b_c, \quad (11)$$

where $\mathbf{w}_c$ denotes the class-specific weight vector and b_c the corresponding bias term. The predicted class is the class with the largest score:

$$\hat{y}_{\text{LDA}} = \arg \max_c g_c(\mathbf{x}). \quad (12)$$

In hardware, LDA inference therefore reduces to a set of multiply–accumulate operations followed by an argmax comparison over the class scores. This makes LDA an efficient inference-only baseline, but model formation and local model updates are not performed on the FPGA in this work. HDC and LDA are therefore not functionally equivalent with respect to post-deployment model updates.

3.4. FPGA-Based HDC Training and Inference

The architecture executes both model formation and inference directly on the FPGA after memory initialization and configuration. Item memory and continuous item memory are fixed encoder memories loaded once before execution, whereas associative-memory contents are created on device during training and form the trainable part of the model. During training, encoded sample hypervectors are accumulated into class-specific counters and thresholded into binary class hypervectors for associative-memory storage; during inference, the same encoder generates a query hypervector that is classified by nearest-neighbor search in Hamming space.

The hardware consists of item memory, level memory, an encoding and bundling datapath, class accumulators, associative memory, and a Hamming distance comparator. The resulting training and inference pipeline is described in more detail below and summarized in Figure 1. Hypervectors are processed in fixed-size datapath segments instead of fully parallel D -bit operations, enabling dimensions up to $D = 10,000$ on GateMate A1 while reducing logic and routing pressure.

For a datapath segment width W , one hypervector is stored in

$$B = \left\lceil \frac{D}{W} \right\rceil \quad (13)$$

segments, corresponding to

$$D_{\text{hw}} = BW \quad (14)$$

stored hardware bits. Other evaluated variants use different segment widths or additional parallelism, allowing the design to trade sequential processing time against routed logic and memory pressure. Table 2 summarizes the main memories used by the accelerator.

Table 2. Main memories used by the HDC accelerator.

Memory	Stored Data	Depth	Access
Item memory	Feature/item hypervectors	$N \cdot B$ segments	Load and encode
Level memory	Quantization-level hypervectors	$M \cdot B$ segments	Load and encode
Associative memory	Binary class hypervectors	$C \cdot B$ segments	Load, finalize, compare

Before an experiment, the accelerator interface loads the item memory and level memory and configures the training or inference run. As illustrated in Figure 1, both training and test samples pass through the same encoder. For every EMG feature, the encoder combines the corresponding item-memory segment with the segment of the quantized level hypervector using bitwise XOR. The encoded feature hypervectors are then bundled by majority voting across the 32 input features. With $N = 32$, the majority threshold is 16; therefore, a 16-to-16 tie is resolved as one in the hardware implementation.

During model formation, the encoded training samples are accumulated class-wise. After all samples of a class have been processed, majority voting converts the accumulated representation into one binary class hypervector, which is stored in associative memory. The associative memory therefore contains one stored prototype hypervector per class.

During operation, the associative-memory contents are stored in on-chip FPGA memory/BRAM. These contents are volatile and must be retrained or reloaded after power loss unless an external non-volatile storage mechanism is added. Power-loss-safe persistence and transactional update procedures are outside the current prototype and remain future work.

During inference, a test sample is encoded in the same way and becomes the query hypervector shown in Figure 1. The query is compared with all stored class hypervectors by segment-wise XOR and population-count accumulation, yielding one Hamming distance per class. The predicted class is the associative-memory entry with the minimum distance. Because all hypervector segments must be processed, inference latency scales approximately linearly with the number of segments B .

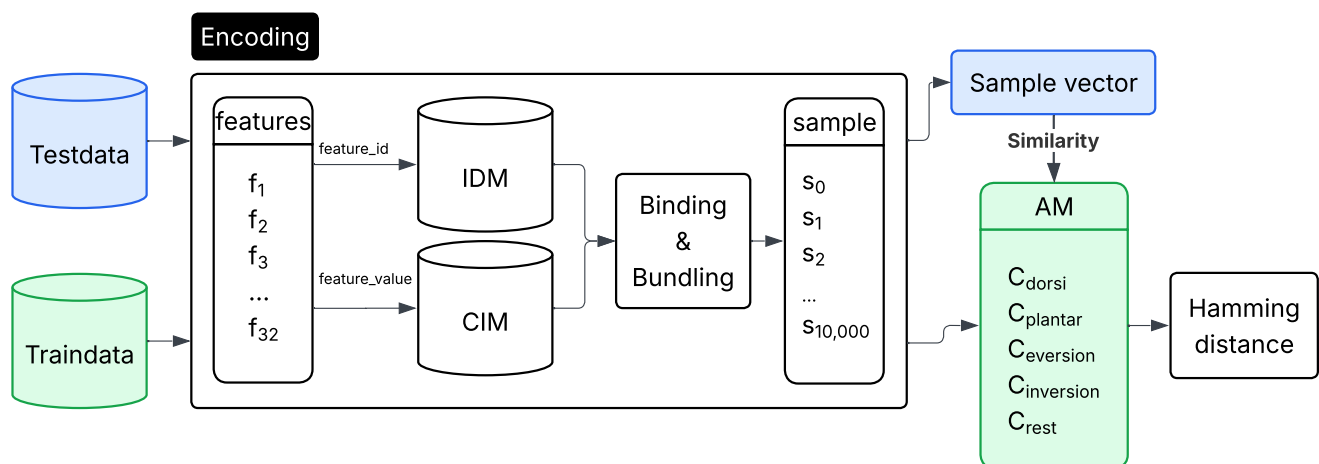


Figure 1. Block diagram of the proposed HDC hardware pipeline. The same encoder maps training and test samples into binary hypervectors. Training samples are accumulated class-wise and stored as associative-memory prototypes after majority voting; during inference, the encoded query is compared against the stored prototypes using Hamming distance accumulation.

3.5. GateMate A1 Implementation and Evaluation Protocol

All FPGA resource results are generated using an open-source GateMate implementation flow based on OSS CAD Suite (release 9 June 2026), combining Yosys with the GHDL frontend and nextpnr-himbaechel for placement, routing, and timing analysis on the CCGM1A1 device. Identical implementation settings are used for all HDC and LDA variants to ensure comparability.

The evaluated variants implement the same HDC training and inference pipeline and differ only in datapath segmentation, encoder replication and pipelining. The comparison therefore reflects architecture–algorithm trade-offs rather than changes to the underlying

classifier. In the variant names used below, a 40-bit or 200-bit segmented datapath denotes the number of hypervector dimensions processed in parallel per datapath segment. Wider segmented datapaths reduce the number of serialized segment iterations for a fixed hypervector dimension D , but increase memory bandwidth, routed resource utilization, and routing pressure. The pipelined replicated variant inserts registers into the replicated encoder datapath to shorten combinational paths at the cost of additional registers and pipeline latency. Specifically, it splits the single encode-and-accumulate step into two registered stages: the item- and level-memory outputs are first XOR-bound and registered, and the resulting encoded bits update the per-bit class counters only in the following cycle. This breaks the otherwise long path from memory output through XOR into the wide per-bit counter-increment logic, improving the achievable clock frequency at the cost of one additional pipeline cycle per processed feature group.

The evaluation covers HDC design-space exploration, training-fraction sensitivity, routed FPGA resource usage, input-noise robustness, and robustness against model-level perturbations of the stored HDC class hypervectors.

The HDC accuracy sweep uses D values from 500 to 10,000 and M values from 10 to 100 across the eight recordings. For each run, the item memory is generated with independent random binary hypervectors, and the level memory is generated by starting from a random level hypervector and repeatedly flipping a fixed number of bits to obtain correlated neighboring levels. Each HDC configuration was evaluated over 10 independent random encoder initializations, and results are reported as mean and standard deviation. The sweep is used to characterize the sensitivity of HDC to hypervector dimension, quantization granularity, and random memory initialization. The per-recording best configurations are reported exclusively to analyze parameter sensitivity and achievable performance ranges and are not used as deployment configurations in subsequent experiments. To avoid using separate best-case parameters in the subsequent robustness and training-fraction experiments, those experiments use one fixed global representative configuration selected across all recordings.

The algorithmic sweep reports the best HDC configuration per recording to account for recording-dependent parameter optima. The hardware resource evaluation reports routed utilization for the implemented FPGA design variants.

Input-level robustness is evaluated by adding zero-mean Gaussian noise directly to the test features. Controlled Gaussian perturbations are commonly used in EMG robustness studies to evaluate the sensitivity of feature extraction and classification pipelines to signal uncertainty and measurement variation. In this work, the perturbation model is intended as a controlled first-order approximation of feature uncertainty rather than a physiological model of EMG degradation, electrode displacement, or long-term session variability [27,28].

$$\tilde{x}_i = x_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2). \quad (15)$$

Here, x_i denotes the original standardized test feature of channel i , $\tilde{x}_i$ the corresponding perturbed feature, ϵ_i an independently sampled Gaussian noise term, and σ the tested noise standard deviation.

The tested noise standard deviations are $\sigma \in \{0, 0.01, 0.05, 0.10, 0.30, 0.50, 0.70, 1.00\}$. Noise is added only to the test features after training and quantization-range selection, so the experiment isolates sensitivity to controlled feature perturbations rather than adaptation to physiological EMG drift.

Model-level robustness is evaluated by injecting random bit flips into the stored HDC model representation while leaving the input test data unchanged. The implementation specifies the perturbation probability in parts per million (ppm); the paper reports the corresponding percentage of flipped model bits, $p_{\%} = p_{\text{ppm}}/10,000$. The tested rates

range from 0% to 20% with denser sampling at low perturbation rates. For HDC, flips are applied after training and before inference to the stored binary class hypervectors in associative memory. This separation between input-level and model-level perturbations allows distinguishing sensor-noise robustness from robustness against internal memory or configuration faults.

Robustness results are reported as absolute accuracy across the eight recordings. The plots report mean accuracy together with the spread across recording means.

The following section evaluates classification performance, training-fraction sensitivity, robustness, routed hardware utilization, and implementation trade-offs across the investigated configurations.

4. Results

4.1. HDC Accuracy and Design-Space Exploration

The HDC classifier was evaluated for different hypervector dimensions D and level counts M using random item memories and bit-flip-generated correlated level memories. The sweep shows measurable recording-dependent sensitivity to these parameters, while accuracy remains comparatively stable over broad configuration ranges for most recordings. Larger dimensions often stabilize the representation, but the optimum is not identical across recordings and does not always occur at the largest tested dimension. Table 3 summarizes the best mean HDC configuration for each recording over 10 random seeds and reports the corresponding full-training LDA baseline for the same recording. These per-recording optima are used to analyze attainable HDC performance and recording-dependent parameter sensitivity. Each reported configuration corresponds to the mean over 10 independent random encoder initializations.

Table 3. Best HDC configurations and LDA baseline accuracy in the eight-recording accuracy sweep.

Recording	D	M	HDC Mean (%)	HDC Std. (pp)	LDA Mean (%)
Recording 01	10,000	60	97.46	0.08	100.00
Recording 02	3000	60	96.11	1.72	100.00
Recording 03	7000	20	94.07	0.30	97.70
Recording 04	1000	70	87.78	1.64	96.90
Recording 05	8000	90	98.74	0.26	100.00
Recording 06	7000	10	94.09	0.92	95.77
Recording 07	7000	30	96.10	0.38	98.44
Recording 08	9000	70	97.27	1.88	99.17

Figure 2 further illustrates the recording-level sensitivity of HDC accuracy to the hypervector dimension and the number of quantization levels. Across most recordings, accuracy remains comparatively stable over broad ranges of D and M , indicating that the evaluated HDC representation is not strongly dependent on a single narrowly tuned

configuration. Larger deviations are observed only for selected recordings, most notably Recording 04 and Recording 06, which suggests that parameter sensitivity is driven primarily by recording-specific signal characteristics.

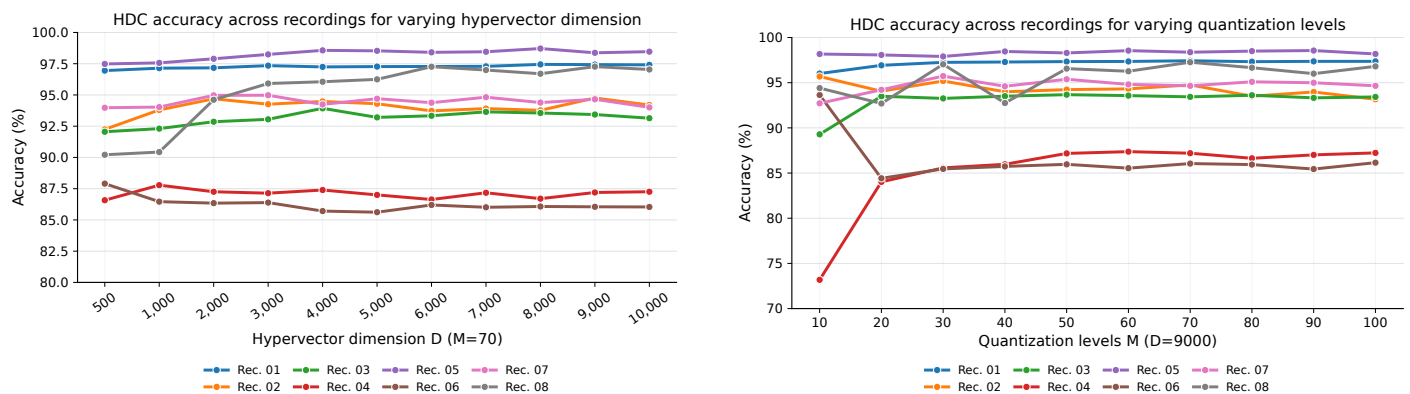


Figure 2. Recording-level HDC accuracy sensitivity for selected one-dimensional sweeps. Left: influence of hypervector dimension D at fixed $M = 70$. Right: influence of quantization levels M at fixed $D = 9000$. Each curve represents one EMG recording.

The best-accuracy comparison confirms the expected behavior: LDA achieves the highest clean accuracy in the evaluated setup, with a sample-weighted overall accuracy of 98.54% across the eight recordings. The HDC per-recording best configurations reach 95.20% mean recording accuracy and remain competitive. Table 3 also shows the recording-wise comparison: on most recordings, the HDC results remain within a few percentage points of the LDA baseline, while Recording 04 shows the largest separation between both methods. As introduced in the dataset description, Recording 04 corresponds to the participant with spinal cord injury included in the evaluation corpus. The recording-wise confusion matrices in Figure 3 show that the main classification difficulties for this recording occur in the inversion and eversion classes. The best single fixed HDC configuration in the clean-accuracy sweep is $D = 9000$, $M = 70$, $F = 1.0$, with a mean accuracy of 93.64% and a standard deviation of 4.65 percentage points across recording means. This configuration is used as the representative configuration for the subsequent robustness and training-fraction studies.

Across the recording-wise confusion matrices, the rest control state as well as dorsiflexion and plantarflexion are generally recognized with high accuracy. The strongest remaining confusion occurs mainly between inversion and eversion, especially in Recording 04. This pattern is consistent with ankle biomechanics: dorsiflexion and plantarflexion are the primary sagittal-plane ankle motions during gait, whereas inversion and eversion occur mainly in the frontal plane and involve the subtalar complex [29,30]. Inversion and eversion also typically have smaller ranges of motion and more coupled joint mechanics than the sagittal-plane motions [30]. Consequently, inversion and eversion may exhibit more similar surface EMG activation patterns, making these classes more difficult to distinguish in some recordings.

The best-accuracy comparison indicates that LDA provides the stronger best-accuracy reference in the evaluated setup, whereas HDC is investigated for the additional capability of hardware-native model formation and unified execution on the FPGA. The comparison should therefore be read as a contrast between a strong offline-trained inference baseline and a trainable HDC hardware architecture rather than as a claim of equal deployment functionality. A fully fair comparison of post-deployment adaptation would require an online-learning baseline with hardware-supported model updates, such as an OnlineHD-

style implementation or another trainable FPGA classifier. Such a baseline is outside the scope of the present work and remains future work.

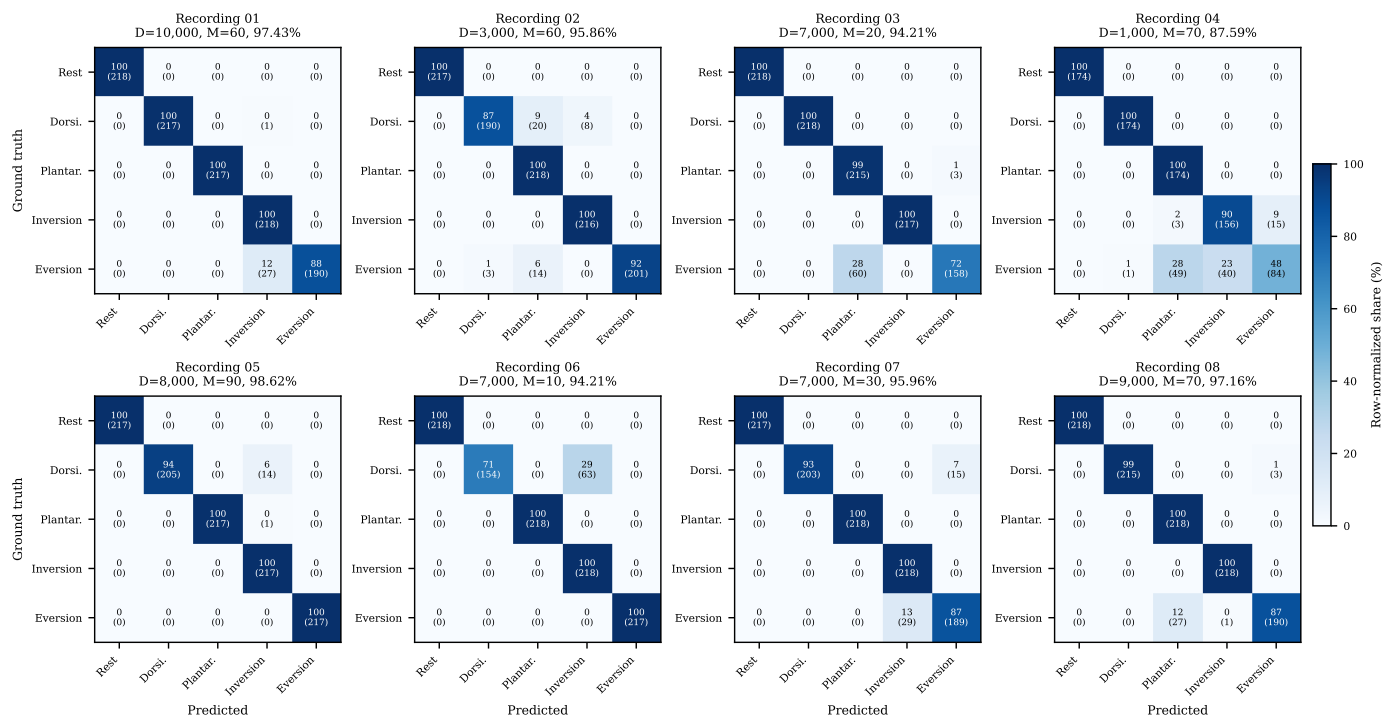


Figure 3. Recording-wise HDC confusion matrices for all eight recordings using the per-recording best configurations from Table 3 and $F = 1.0$. For each recording, the displayed run is the deterministic re-evaluation closest to the corresponding Table 3 mean accuracy. Rows denote ground-truth classes and columns denote predicted classes. Cell values show row-normalized percentages with sample counts in parentheses.

A paired Wilcoxon signed-rank test was performed on the eight recording-level clean-session accuracies in Table 3, using the per-recording HDC best mean accuracy and the corresponding LDA baseline accuracy as paired observations. A non-parametric paired test was selected because the comparison contains only eight paired recordings and does not require assuming normally distributed paired differences. The test indicated higher clean-session accuracy for LDA across the evaluated recordings ($n = 8$, $W = 0.0$, exact two-sided $p = 0.0078$). This statistical result confirms the descriptive clean-accuracy comparison reported in Table 3. As discussed throughout this work, the proposed contribution focuses on unified hardware-native model formation and inference rather than maximizing clean-session classification accuracy.

Using one global configuration prevents the later experiments from relying on per-recording best-case hyperparameters and provides a more realistic hardware-relevant operating point.

4.2. Training-Fraction Sensitivity

The representative HDC configuration was evaluated with training-data fractions $F \in \{0.01, 0.05, 0.10, 0.20, \dots, 1.00\}$. These fractions correspond to approximately 347 to 34,744 training samples across the eight recordings. For HDC, reducing F directly changes the amount of training data used for hardware-native model formation on the FPGA. For LDA, the same fractions are applied only during offline model generation to maintain comparable training-data availability, while the resulting FPGA implementation still performs inference only. Figure 4 shows that mean accuracy remains close to the full-training

result for moderate fractions. The largest HDC reduction occurs at $F = 0.01$, where mean accuracy decreases by approximately 2.76 percentage points relative to full training; the corresponding LDA reduction is approximately 2.20 percentage points.

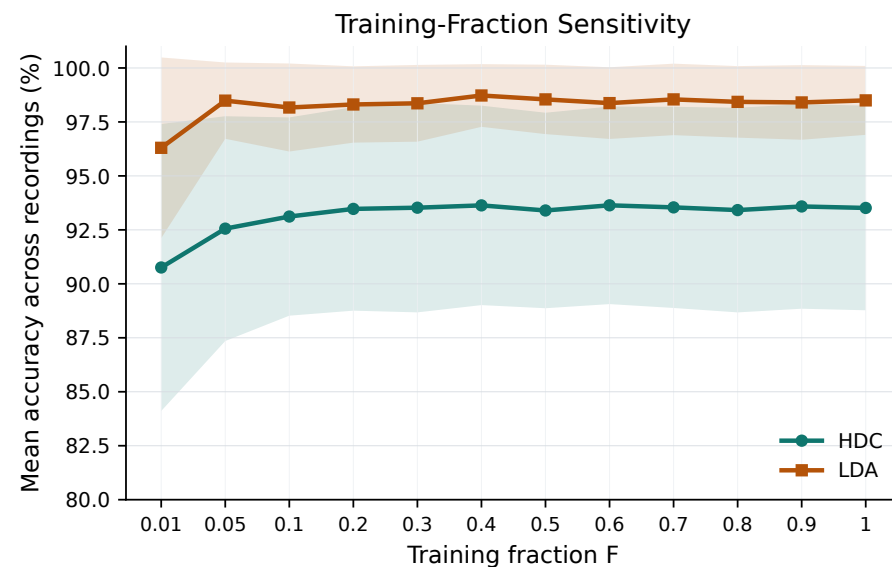


Figure 4. Training-fraction sensitivity of HDC and LDA for the eight-recording evaluation. The tested fractions F use the corresponding portion of the 34,744-sample training split, ranging from approximately 347 samples at $F = 0.01$ to all 34,744 samples at $F = 1.0$. The lines show the mean across recording-level means, and the shaded regions indicate the spread across recordings.

The results indicate that HDC maintains stable classification performance over a broad range of available training data and does not require the full training split to reach competitive accuracy. This behavior is consistent with the distributed representation and majority-based model formation used in HDC. While LDA remains the stronger clean-accuracy baseline, the comparison suggests that HDC can retain practical EMG classification performance while simultaneously supporting on-device model formation and unified hardware execution.

4.3. FPGA Resource Utilization of HDC Variants

Table 4 summarizes the routed resource ranges for the evaluated HDC hardware variants and the LDA reference design. The values are reported as post-place-and-route device-utilization percentages for the accelerator and predictor cores without the UART bootloader interface. For HDC variants with multiple available implementation logs, the table reports the observed range over D at $M = 40$. The 40-bit and non-replicated 200-bit segmented datapaths are reported for $D = 1000$ – $10,000$, the replicated 200-bit encoder for $D = 1000$ – 6000 , and the pipelined replicated 200-bit encoder for $D = 1000$ – 5000 . The reported utilization ranges are primarily caused by varying the hypervector dimension D , which determines the number of stored hypervector segments and therefore the size of the item, level, associative, and intermediate training memories, while all other architectural parameters remain fixed unless explicitly stated. LDA is independent of the HDC hypervector dimension and is therefore reported as a single routed reference design.

The reported logical memory footprint corresponds to the stored HDC model (item, level, associative, and training memories) and is independent of FPGA-specific BRAM packing. For the replicated and pipelined replicated encoder variants, the item and level memories are duplicated to provide parallel encoder lanes, which explains their larger logical memory footprint compared with the non-replicated 200-bit segmented datapath.

Table 4. Routed resource ranges for the evaluated HDC and LDA hardware cores on GateMate A1. Resource values are post-place-and-route utilization percentages without the UART bootloader interface. Percentages are relative to the total routable CCGM1A1 device resources reported by the used implementation flow: 40,960 look-up table sites (CPE_LT), 40,960 flip-flop sites (CPE_FF), and 64 half RAM blocks (RAM_HALF).

Variant	Functionality	D Range	Logical Memory Footprint	LUT	FF	BRAM	$f_{\max}$ (MHz)
40-bit seg. datapath	FPGA training + inference	1000–10,000	178–890 kbit 21.7–108.6 KiB	13–17%	4%	18–71%	32.22–34.76
200-bit seg. datapath	FPGA training + inference	1000–10,000	178–890 kbit 21.7–108.6 KiB	34–39%	10%	34–87%	16.94–20.91
Replicated 200-bit enc.	FPGA training + inference	1000–6000	322–966 kbit 39.3–117.9 KiB	50–61%	10%	50–100%	17.09–21.30
Pipelined repl. 200-bit enc.	FPGA training + inference	1000–5000	322–805 kbit 39.3–98.3 KiB	60–61%	11–12%	51–82%	17.33–19.69
LDA	Offline training + FPGA inference	–	–	54%	16%	0%	12.20

The routed implementation results show that resource scaling is dominated by on-chip memory rather than by routed logic or registers. Across increasing D , LUT and FF utilization remain comparatively stable within each architecture, while BRAM utilization increases strongly. For example, BRAM utilization grows from 18% to 71% for the 40-bit segmented datapath and from 34% to 87% for the 200-bit segmented datapath. The replicated 200-bit encoder reaches full BRAM utilization already at $D = 6000$, indicating that memory capacity becomes the practical implementation limit before logic utilization is exhausted. Because these values are obtained after synthesis, placement, and routing, the reported utilization does not necessarily scale linearly with D . Tool-dependent mapping decisions, packing efficiency, memory inference, and the specific BRAM and routing architecture of the GateMate A1 can introduce step-like or non-monotonic changes in the final resource percentages.

To assess whether the reported implementation characteristics are specific to GateMate A1 or reflect the proposed architecture more generally, the HDC hardware variants were additionally implemented on a Xilinx Zynq UltraScale+ xczu3eg-sbva484-1-i device used by the Ultra96 platform using Vivado 2024.1. This experiment is intended as a cross-platform portability check rather than a board-level benchmark. The Ultra96 implementations close timing at substantially higher clock frequencies, whereas absolute utilization percentages differ because of vendor-specific logic-cell definitions, memory primitives, routing fabrics, and implementation tools. Therefore, the values in Table 5 should not be interpreted as a direct vendor efficiency ranking. For the Ultra96 results, $f_{\max}$ denotes the highest timing constraint that closed with non-negative WNS in the performed Vivado implementation runs. Nevertheless, successful timing closure for the same synthesizable HDC variants on both GateMate A1 and Xilinx Zynq UltraScale+ indicates that the proposed architecture is portable and not tied to GateMate-specific primitives.

Although the absolute utilization percentages differ across FPGA families because of different LUT structures, BRAM granularities, routing architectures, and timing models, the comparison indicates that the proposed HDC pipeline is not inherently tied to GateMate A1. The relative scaling trends remain architecture-driven: increasing D primarily increases memory footprint, while wider datapaths trade lower cycle count for higher routing and memory pressure.

Table 5. Cross-platform implementation comparison for HDC variants at $M = 40$. GateMate A1 values reuse the routed utilization ranges from Table 4; Ultra96 values are post-implementation Vivado 2024.1 routed counts with device-utilization percentages in parentheses. Only HDC variants available on both platforms are included, evaluated over the same D range on each platform.

Variant	Platform	D Range	LUT	FF	BRAM	DSP	$f_{\max}$ (MHz)
40-bit seg. datapath	GateMate A1	1000–10,000	13–17%	4%	18–71%	–	32.22–34.76
40-bit seg. datapath	Ultra96 ZU3EG	1000–10,000	1294–1393 (1.83–1.97%)	1701–1743 (1.21–1.24%)	5.5–28.5 (2.55–13.19%)	0	139.6–140.0
200-bit seg. datapath	GateMate A1	1000–10,000	34–39%	10%	34–87%	–	16.94–20.91
200-bit seg. datapath	Ultra96 ZU3EG	1000–10,000	2978–3006 (4.22–4.26%)	3301–3342 (2.34–2.37%)	12.5–33 (5.79–15.28%)	0	146.4–146.7
Replicated 200-bit enc.	GateMate A1	1000–6000	50–61%	10%	50–100%	–	17.09–21.30
Replicated 200-bit enc.	Ultra96 ZU3EG	1000–6000	3513–3533 (5.01%)	3320–3428 (2.35–2.43%)	18.5–56 (8.56–25.93%)	0	138.7–147.6
Pipelined repl. 200-bit enc.	GateMate A1	1000–5000	60–61%	11–12%	51–82%	–	17.33–19.69
Pipelined repl. 200-bit enc.	Ultra96 ZU3EG	1000–5000	4358–4374 (6.20%)	3808–3946 (2.70–2.80%)	20–55.5 (9.26–25.69%)	0	138.2–143.4

The comparison with LDA reflects different deployment paradigms rather than equal-functionality classifier cost. LDA implements offline-trained FPGA inference only, whereas HDC integrates feature encoding, on-device model formation, binary model storage, and inference in one architecture. Consequently, HDC requires additional on-chip memories for item memories, continuous item memories, associative memories, and intermediate training state, so its resource scaling is primarily memory constrained. The benefit of HDC is therefore not a simple clean-accuracy advantage over LDA, but the integration of model formation and update-capable model storage into the FPGA execution flow. Increasing datapath parallelism or reducing serialization in HDC must therefore be balanced against the available RAM resources on GateMate A1.

The timing results complement this resource trend. They indicate that moderate datapath widths provide the most practical implementation balance, while aggressive inference or training parallelism is limited mainly by BRAM utilization and routing pressure.

4.4. Hardware Latency and Throughput

The latency measurements complement the routed resource results by quantifying the serialization cost of the HDC datapath. Figure 5 shows the routed inference latency for the evaluated $M = 40$ designs, computed from the measured cycle count and the post-route maximum clock frequency of each implementation. The measurements use representative hardware testbench stimuli with one training sample and one test sample to characterize datapath latency rather than full-corpus execution time. The corresponding cycle counts are constant across the measured samples for a fixed configuration.

Table 6 summarizes compact representative timing points for $M = 40$ in the same style as the routed resource overview. Latency is computed from the measured cycle count and the post-route maximum frequency reported by nextpnr-himbaechel.

Cycles, latency, and throughput are reported from the smallest routed configuration to the largest routed configuration available for each variant at $M = 40$, while $f_{\max}$ gives the observed range over the listed routed points. Figure 5 uses only points for which routed maximum-frequency data are available. The replicated 200-bit encoder is reported up to $D = 6000$, while the pipelined replicated 200-bit encoder is summarized from $D = 2000$ to $D = 5000$.

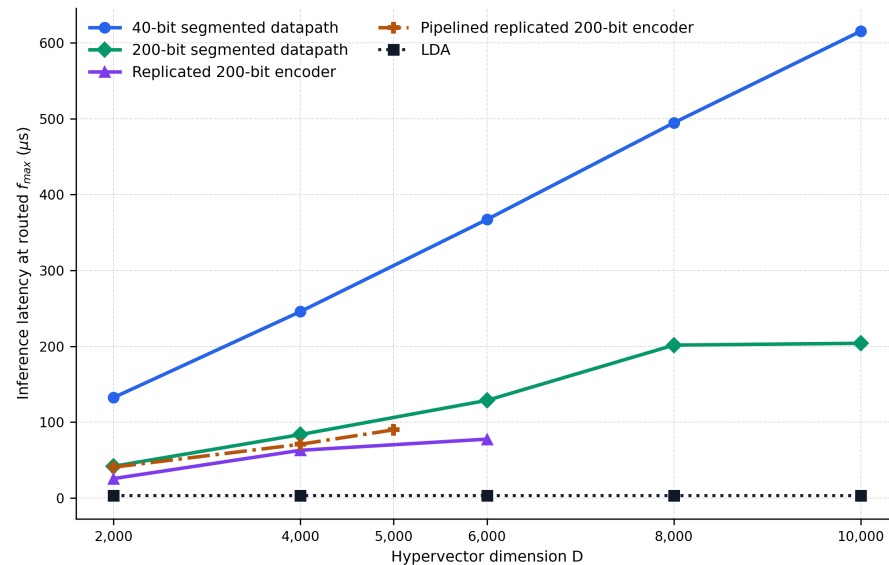


Figure 5. Routed inference latency for the evaluated HDC variants and the LDA inference-only baseline. Latency is computed from measured inference cycles and the post-route maximum clock frequency of each implementation.

Table 6. Representative routed inference latency and throughput for $M = 40$ on GateMate A1.

Variant	D Points	Cycles	f_{max} (MHz)	Latency (μs)	Throughput (Samples/s)
40-bit seg. datapath	2000, 10,000	4263–21,263	32.22–34.56	132.31–615.25	7558–1625
200-bit seg. datapath	2000, 10,000	863–4263	20.67–20.91	41.75–203.87	23,951–4905
Replicated 200-bit enc.	2000, 6000	543–1603	21.30–20.67	25.49–77.55	39,227–12,895
Pipelined repl. 200-bit enc.	2000, 5000	703–1738	17.33–19.35	40.57–89.82	24,651–11,133
LDA	independent	38	12.20	3.11	321,053

The HDC results show the expected dependence on hypervector dimension and datapath segmentation. For a fixed datapath organization, the number of processed hypervector segments increases with D , so inference cycles scale approximately linearly. Wider and more parallel variants reduce this serialization overhead: at $D = 2000$, the 200-bit segmented datapath requires substantially fewer cycles than the 40-bit segmented datapath, and the replicated 200-bit encoder further reduces inference latency. The pipelined variant also uses a replicated 200-bit encoder, but adds registered encoder stages. These latency reductions must be interpreted together with the routed implementation costs shown in Table 4, especially higher BRAM utilization and increased routing pressure. Consequently, the fastest inference configuration does not necessarily represent the most practical overall hardware operating point.

As expected, the LDA baseline achieves very low inference latency because it evaluates a compact fixed-size linear model and performs only inference after offline model generation. The latency comparison therefore reflects different deployment objectives rather than identical hardware functionality. In contrast, the proposed HDC architecture additionally integrates feature encoding, hardware-native model formation, associative-memory generation, binary model storage, and inference within a unified execution flow directly on the FPGA.

Despite these additional responsibilities and deployment on the resource-constrained GateMate A1 FPGA, the evaluated HDC variants reach low-microsecond inference latency through datapath parallelization and encoder replication. The EMG feature stream operates at approximately 111 Hz, corresponding to a frame period of about 9 ms. All evaluated HDC hardware variants remain well below this timing budget for inference, even for the largest tested hypervector dimensions. The reported latency differences therefore mainly affect energy, duty cycle, and available time for preprocessing or communication rather than real-time feasibility. Depending on the selected architecture, the measured throughput already reaches tens of thousands of samples per second. These results indicate that trainable HDC architectures can approach the execution range of compact inference-oriented classifiers while simultaneously providing hardware-native model formation and unified model execution directly in hardware.

As a first board-level power measurement, the 40-bit segmented HDC implementation was evaluated at $D = 4000$ and $M = 40$ using the measured voltage drop across a $1\ \Omega$ shunt resistor in the 5 V supply path. The voltage traces were recorded with PicoScope 7 T&M software, a PicoScope 3405D oscilloscope, and a TA132 probe. The board-side supply voltage was approximately 4.93 V. The measured shunt voltage drop was approximately 92.5 mV in idle and 94.9 mV during training; current was therefore computed as $I = \Delta V / R_{\text{shunt}}$, and board power as $P = V_{\text{board}} I$. This gives an idle board power of approximately 0.456 W and a measured board power during training of approximately 0.467 ± 0.001 W, corresponding to an activity-related increase of about 11.2 ± 0.6 mW. For the same configuration, a time of 0.8513 ms for testing and 1.8502 ms for training was measured. Using the measured training power as a board-level estimate gives approximately 0.398 mJ per inference sample and 0.865 mJ per training sample; the incremental energy over idle is approximately 9.5 μJ and 20.7 μJ , respectively. Class finalization required 1.6003 ms in this configuration and is performed once per finalized class rather than once per input sample. These values should be interpreted as evaluation-board-level estimates that include static board consumption and measurement setup effects, not as isolated FPGA-core power or as a complete power characterization over all HDC variants.

The latency comparison therefore highlights that efficient trainable AI systems can be realized directly on constrained FPGA hardware rather than indicating an inherent performance disadvantage of HDC for embedded EMG classification.

4.5. Robustness Against Input Noise

Input-level robustness was evaluated by perturbing the EMG test data while keeping the trained models unchanged. This robustness experiment uses the representative HDC configuration $D = 9000$, $M = 70$, and $F = 1.0$ and the offline-trained, FPGA-inferred LDA baseline. Figure 6 shows that both classifiers remain close to their clean baselines at low input-noise levels. For stronger perturbations, both classifiers show decreasing accuracy, with the evaluated HDC configuration retaining a larger fraction of its clean-baseline performance over the investigated noise range.

Across the evaluated controlled feature-perturbation setting, the robustness advantage becomes increasingly visible at moderate and high noise levels.

At the representative perturbation level $\sigma = 0.10$, the evaluated HDC configuration retained higher mean recording accuracy than the LDA baseline, consistent with the descriptive robustness trend observed across the investigated noise range.

A possible explanation for this behavior is the distributed high-dimensional binary representation used by HDC, where information is represented redundantly across many dimensions and aggregated through majority-based model formation [7,8,15,16]. Local perturbations in the input space therefore affect only a fraction of the resulting representa-

tion, whereas perturbations in compact linear models can influence decision boundaries more directly.

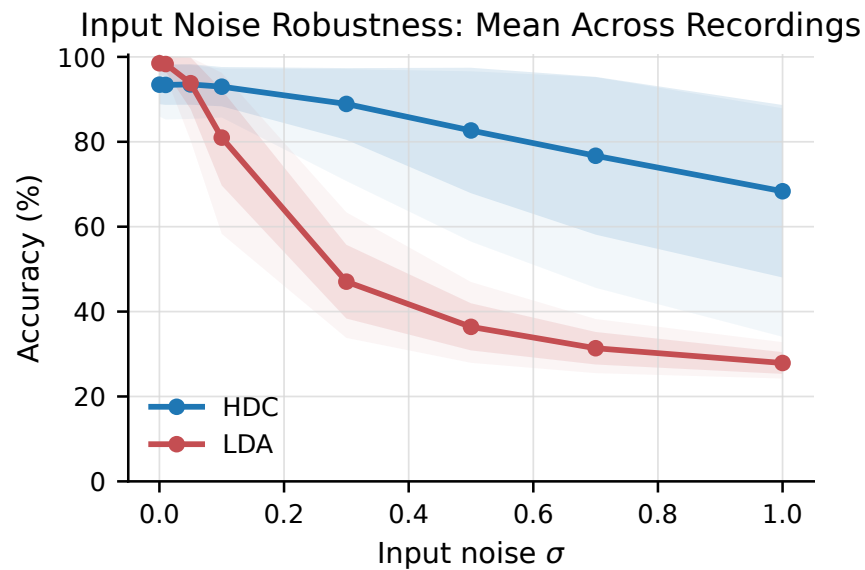


Figure 6. HDC and LDA accuracy under input/test-data noise for the eight-recording evaluation. The lines show the mean accuracy across recording means, and the shaded regions indicate the spread across recordings.

4.6. Robustness Against Model and Configuration Noise

The second robustness experiment evaluates model-level perturbations. For HDC, random bit flips were injected into the stored binary class hypervectors. For LDA, the stored classifier parameters were perturbed using the same relative noise levels. Figure 7 summarizes the resulting accuracy across all eight recordings.

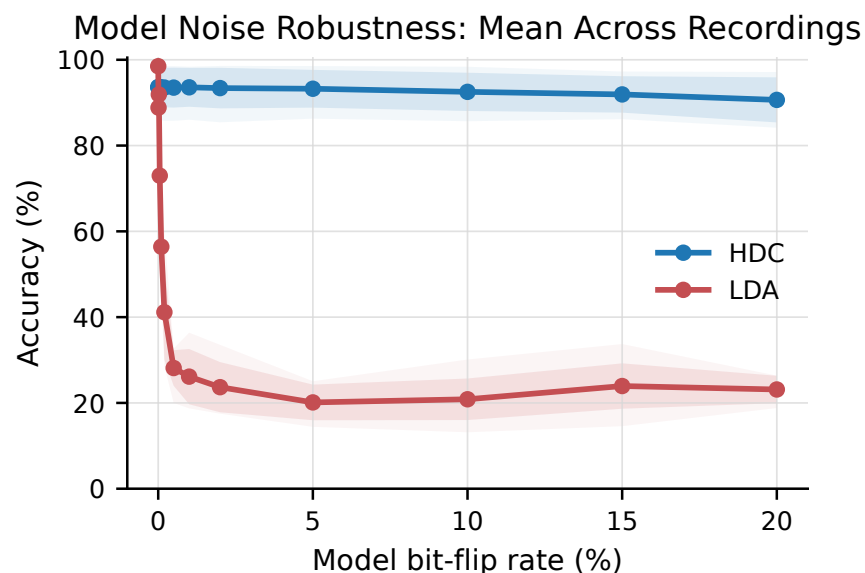


Figure 7. HDC and LDA accuracy under model/configuration noise for the eight-recording evaluation. The lines show the mean accuracy across recording means, and the shaded regions indicate the spread across recordings.

The HDC model representation remains comparatively stable under low and moderate perturbation rates in the evaluated range. Mean accuracy stays close to the clean

baseline up to several thousand ppm and decreases more noticeably only for the largest perturbation rates.

This behavior is consistent with the observations from the input-noise experiment in Section 4.5, where HDC also retained a larger fraction of its clean-baseline accuracy under increasing perturbation levels. A possible explanation is that robustness in HDC originates from the same distributed high-dimensional representation in both cases.

For model perturbations specifically, the associative-memory representation itself consists of high-dimensional binary class hypervectors. Random bit flips therefore modify only a small subset of dimensions while leaving most of the stored class representation unchanged. Since classification is based on global Hamming similarity rather than a small number of critical coefficients, perturbations tend to degrade the decision gradually instead of causing abrupt failure.

From a hardware perspective, this result is relevant because model perturbations can approximate memory faults, configuration disturbances, or storage degradation in embedded systems. The experiment therefore does not only evaluate classifier robustness but also provides an indication of how binary model representations behave under implementation-level disturbances on constrained hardware.

5. Discussion

The presented results demonstrate that trainable HDC can execute directly on a compact FPGA platform while maintaining competitive EMG classification performance, low-microsecond execution latency, and favorable behavior under controlled input and model perturbations. Rather than separating training and deployment into independent stages, the proposed architecture enables model formation, class-prototype storage, and inference within a unified hardware-native execution flow.

From a hardware perspective, the evaluation showed that practical scalability is determined primarily by memory organization and data movement rather than computational complexity alone. Increasing datapath width, encoder replication, and training parallelism reduced serialization overhead and improved latency, but also increased routed memory utilization and implementation pressure. Consequently, selecting HDC parameters and hardware architecture jointly remains essential for achieving balanced deployment points on resource-constrained FPGA devices.

The accuracy sweeps further show that practical configurations do not have to be selected solely from the largest evaluated hypervector dimensions. Because several recordings remain stable over broad ranges of D and M , lower-dimensional configurations may provide attractive deployment points when a small accuracy trade-off is acceptable. Reducing D lowers associative-memory and encoder-memory requirements and also reduces the number of serialized hypervector segments, which can improve per-sample latency for segmented datapaths. The final configuration should therefore be selected as a joint accuracy–resource–timing trade-off rather than by accuracy alone.

From the EMG classification perspective, the results indicate that HDC can maintain competitive accuracy while additionally enabling local model formation and hardware-native execution. At the same time, the experiments show that no single HDC configuration consistently dominates across all recordings, which is why the per-recording best results should be interpreted as a sensitivity analysis rather than as a deployment protocol. Hypervector dimension, quantization granularity, and available training data therefore remain application-dependent design choices that should be selected together with the intended hardware target and deployment scenario.

Repeated model updates do not require regeneration of the fixed item and continuous item memories. The architecture is therefore designed so that future update procedures

could modify only the associative-memory representation on device, reducing deployment complexity while preserving hardware-native model formation. Accordingly, local model updating is treated here as an architectural capability, while explicit repeated-session adaptation remains future work.

The recording-level analysis additionally highlights the influence of subject characteristics on classification performance. While most recordings reached high classification accuracy, Recording 04 produced the lowest overall performance across the evaluated configurations. As introduced in the dataset description, this recording corresponds to the participant with spinal cord injury included in the evaluation corpus. In this case, the main classification difficulty was observed between inversion and eversion movements, indicating increased overlap in the corresponding EMG activation patterns compared to the other recordings. Recording 04 is therefore treated as an exploratory impaired-motor-pattern case rather than as a basis for population-level conclusions about spinal-cord-injury users. Because only one such recording was included, no stratified statistical comparison between impaired and unimpaired participants was performed.

The experimental validation is limited by the dataset size and structure. The dataset contains eight recordings from eight participants and only one impaired participant. Because each participant contributed only one recording session, the reported results should be interpreted as subject-dependent within-session window-level performance. With only eight participants and a single recording per participant, the reported accuracy estimates carry substantial subject-dependent bias, since individual-specific factors such as electrode placement, anatomy, and effort level cannot be separated from algorithmic effects; the small sample size further limits the statistical power of any cross-recording comparison, including the paired test reported above, and the results should not be read as evidence of population-level generalization. The study did not evaluate subject-independent generalization, inter-session variability, long-term signal drift, fatigue, electrode displacement, or longitudinal adaptation. The inclusion of one participant with spinal cord injury provides only an initial indication for impaired motor activation patterns and is not sufficient to draw population-level conclusions for rehabilitation users. These limitations concern the experimental evidence base rather than the hardware architecture itself, which is evaluated here as a platform for single-session model formation and inference.

The robustness experiments further suggest that distributed high-dimensional binary representations provide favorable behavior under controlled feature perturbations and stored-model disturbances, although the perturbation model should not be interpreted as a direct physiological EMG-noise model.

These findings also motivate several directions for future work. From the application perspective, future studies can extend the evaluation corpus with additional participants, repeated-session recordings, and longer observation periods to investigate adaptation behavior under realistic electrode shifts, muscle fatigue, user adaptation, and day-to-day variability. From the hardware perspective, future work can further explore architecture–algorithm co-design, online model-update mechanisms, power-loss-safe model persistence, power and energy measurements across all variants and operating points, and comparisons against online-learning hardware baselines under identical deployment constraints.

Overall, the presented implementation demonstrates that complete HDC model formation and inference are feasible on resource-constrained FPGA hardware and provides a basis for future trainable edge–AI systems combining single-session EMG model formation with hardware-native execution.

6. Conclusions

This work presented a hardware-native Hyperdimensional Computing implementation for trainable EMG classification on the resource-constrained GateMate A1 FPGA. Training, model formation, storage, and inference were executed directly on the FPGA using a unified binary representation.

Across eight single-session EMG recordings, the evaluation demonstrated that trainable HDC can maintain competitive classification accuracy, reaching 95.20% for the per-recording best configurations and 93.64% for the fixed representative configuration, compared with 98.54% for the offline-trained LDA baseline. Under controlled input perturbation at $\sigma = 0.10$, the representative HDC configuration retained 92.98% mean accuracy across recordings. The results further showed that practical deployment is primarily shaped by memory organization and architecture–algorithm trade-offs rather than computational complexity alone.

The design-space and routed FPGA results indicate that HDC configurations should be selected as joint accuracy–resource–timing operating points. Because accuracy remained comparatively stable over broad ranges of D and M for most recordings, lower-dimensional configurations may be attractive when reduced memory use and lower per-sample latency are prioritized. Overall, the presented architecture demonstrates that complete HDC model formation and inference are feasible on constrained reconfigurable hardware without requiring external training infrastructure during the evaluated single-session workflow. At the same time, the results are limited to subject-dependent within-session evaluation and one initial board-level power measurement point. Future work must validate longitudinal sessions, fatigue and electrode-shift conditions, online-learning hardware baselines, full power and energy behavior across operating points, and power-loss-safe persistence strategies before drawing conclusions about post-deployment adaptation in wearable rehabilitation use.

Author Contributions: Conceptualization, K.L. and D.F.; methodology, K.L.; software, K.L.; validation, K.L. and A.N.A.A.; formal analysis, K.L.; investigation, K.L.; resources, D.F.; data curation, A.N.A.A.; preprocessing, A.N.A.A.; EMG data acquisition, A.N.A.A.; LDA implementation and evaluation, A.N.A.A.; writing—original draft preparation, K.L.; writing—review and editing, K.L., A.N.A.A. and D.F.; visualization, K.L.; supervision, D.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Ethics Committee of Friedrich-Alexander-Universität Erlangen-Nürnberg (application 24-286-B, approval date: 17 March 2026).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data presented in this study are available from the corresponding author upon reasonable request. Public release is restricted due to privacy and ethical considerations associated with human EMG recordings.

Acknowledgments: The authors acknowledge the support of the Chair of Computer Science 3 (Computer Architecture), Friedrich-Alexander-Universität Erlangen-Nürnberg.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DSE	Design-space exploration
EMG	Electromyography
FES	Functional electrical stimulation
FF	Flip-flop
FPGA	Field-programmable gate array
HDC	Hyperdimensional computing
RAM	Random-access memory
RMS	Root mean square
BRAM	Block random-access memory
LDA	Linear discriminant analysis
LUT	Look-up table
TCP	Transmission Control Protocol

References

1. van Duijnhoven, E.; Koopman, F.S.; Ploeger, H.E.; Nollet, F.; Brehm, M.-A. Effects of specialist care lower limb orthoses on personal goal attainment and walking ability in adults with neuromuscular disorders. *PLoS ONE* **2023**, *18*, e0279292. [CrossRef] [PubMed]
2. Scheme, E.; Englehart, K. Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use. *J. Rehabil. Res. Dev.* **2011**, *48*, 643–659. [CrossRef] [PubMed]
3. Tam, S.; Boukadoum, M.; Campeau-Lecours, A.; Gosselin, B. A fully embedded adaptive real-time hand gesture classifier leveraging HD-sEMG and deep learning. *IEEE Trans. Biomed. Circuits Syst.* **2020**, *14*, 232–243. [CrossRef] [PubMed]
4. Farina, D.; Merletti, R.; Enoka, R.M. The extraction of neural strategies from the surface EMG. *J. Appl. Physiol.* **2014**, *117*, 1215–1230. [CrossRef] [PubMed]
5. Hermens, H.J.; Freriks, B.; Disselhorst-Klug, C.; Rau, G. Development of recommendations for SEMG sensors and sensor placement procedures. *J. Electromyogr. Kinesiol.* **2000**, *10*, 361–374. [CrossRef] [PubMed]
6. De Luca, C.J.; Gilmore, L.D.; Kuznetsov, M.; Roy, S.H. Filtering the surface EMG signal: Movement artifact and baseline noise contamination. *J. Biomech.* **2010**, *43*, 1573–1579. [CrossRef] [PubMed]
7. Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cogn. Comput.* **2009**, *1*, 139–159. [CrossRef]
8. Ge, Y.; Parhi, K.K. Classification Using Hyperdimensional Computing: A Review. *IEEE Circuits Syst. Mag.* **2020**, *20*, 30–47. [CrossRef]
9. Salamat, S.; Imani, M.; Khaleghi, B.; Rosing, T.S. F5-HD: Fast Flexible FPGA-Based Framework for Refreshing Hyperdimensional Computing. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Seaside, CA, USA, 24–26 February 2019; pp. 53–62.
10. Imani, M.; Salamat, S.; Gupta, S.; Huang, J.; Rosing, T.S. FACH: FPGA-Based Acceleration of Hyperdimensional Computing by Reducing Computational Complexity. In Proceedings of the 24th Asia and South Pacific Design Automation Conference (ASP-DAC), Tokyo, Japan, 21–24 January 2019; pp. 493–498.
11. Hudgins, B.; Parker, P.; Scott, R.N. A new strategy for multifunction myoelectric control. *IEEE Trans. Biomed. Eng.* **1993**, *40*, 82–94. [CrossRef] [PubMed]
12. Englehart, K.; Hudgins, B. A robust, real-time control scheme for multifunction myoelectric control. *IEEE Trans. Biomed. Eng.* **2003**, *50*, 848–854. [CrossRef] [PubMed]
13. Cologne Chip AG. GateMate FPGA Datasheet: CCGM1A1/CCGM1A2. DS1001. August 2025. Available online: <https://colognechip.com/docs/ds1001-gatemate1-datasheet-latest.pdf> (accessed on 11 June 2026).
14. Raurale, S.; McAllister, J.; Martinez del Rincon, J. EMG wrist-hand motion recognition system for real-time embedded platform. *arXiv* **2019**, arXiv:1903.06764.
15. Rahimi, A.; Benatti, S.; Kanerva, P.; Benini, L.; Rabaey, J.M. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In Proceedings of the IEEE International Conference on Rebooting Computing (ICRC), San Diego, CA, USA, 17–19 October 2016; pp. 1–8.
16. Benatti, S.; Farella, E.; Benini, L. Online learning and classification of EMG-based gestures on a parallel ultra-low power platform using hyperdimensional computing. *IEEE Trans. Biomed. Circuits Syst.* **2019**, *13*, 516–528. [CrossRef] [PubMed]

17. Moin, A.; Zhou, A.; Rahimi, A.; Benatti, S.; Menon, A.; Tamakloe, S.; Ting, J.; Yamamoto, N.; Khan, Y.; Burghardt, F.; et al. An EMG gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier. *arXiv* **2018**. [[CrossRef](#)]
18. Zhou, A.; Muller, R.; Rabaey, J.M. Incremental Learning in Multiple Limb Positions for Electromyography-Based Gesture Recognition Using Hyperdimensional Computing. *TechRxiv* **2021**. [[CrossRef](#)]
19. Hernandez-Cano, A.; Matsumoto, N.; Ping, E.; Imani, M. OnlineHD: Robust, Efficient, and Single-Pass Online Learning Using Hyperdimensional System. In Proceedings of the Design, Automation and Test in Europe Conference (DATE), Online, 1–5 February 2021; pp. 56–61.
20. Imani, M.; Zou, Z.; Bosch, S.; Rao, S.A.; Salamat, S.; Kumar, V.; Kim, Y.; Rosing, T.S. Revisiting Hyperdimensional Learning for FPGA and Low-Power Architectures. In Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), Online, 27 February–3 March 2021; pp. 221–234.
21. Zhang, T.; Salamat, S.; Khaleghi, B.; Morris, J.; Aksanli, B.; Rosing, T.S. HD2FPGA: Automated Framework for Accelerating Hyperdimensional Computing on FPGAs. In Proceedings of the International Symposium on Quality Electronic Design (ISQED), San Francisco, CA, USA, 5–7 April 2023; pp. 1–9.
22. Angioli, M.; Jamili, S.; Barbirotta, M.; Cheikh, A.; Mastrandrea, A.; Menichelli, F.; Rosato, M.; Olivieri, M. AeneasHDC: An Automatic Framework for Deploying Hyperdimensional Computing Models on FPGAs. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Yokohama, Japan, 30 June–5 July 2024; pp. 1–8.
23. Datta, S.; Richards, B.C.; Liew, H.; Kim, Y.; Sun, D.; Rabaey, J.M. HDBinaryCore: A 28 nm 2048-Bit Hyperdimensional Biosignal Classifier Achieving 25 nJ/Prediction for EMG Hand-Gesture Recognition. In Proceedings of the IEEE European Solid-State Circuits Conference (ESSCIRC), Lisbon, Portugal, 11–14 September 2023; pp. 229–232.
24. Simpetru, R.C.; Braun, D.I.; Simon, A.U.; März, M.; Cnejevici, V.; de Oliveira, D.S.; Weber, N.; Walter, J.; Franke, J.; Höglinger, D.; et al. MyoGestic: EMG interfacing framework for decoding multiple spared motor dimensions in individuals with neural lesions. *Sci. Adv.* **2025**, *11*, eads9150. [[CrossRef](#)] [[PubMed](#)]
25. Phinyomark, A.; Limsakul, C.; Phukpattaranont, P. Feature Extraction and Reduction of Wavelet Transform Coefficients for EMG Pattern Classification. *Elektron. Ir. Elektrotechnika* **2012**, *122*, 27–32. [[CrossRef](#)]
26. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
27. Phinyomark, A.; Limsakul, C.; Phukpattaranont, P. EMG Feature Extraction for Tolerance of White Gaussian Noise. In Proceedings of ECTI-CON, Krabi, Thailand, 14–17 May 2008.
28. Phinyomark, A.; Limsakul, C.; Phukpattaranont, P. A Novel Feature Extraction for Robust EMG Pattern Recognition. *J. Comput.* **2009**, *1*, 71–80.
29. Perry, J.; Burnfield, J.M. *Gait Analysis: Normal and Pathological Function*, 2nd ed.; SLACK Incorporated: Thorofare, NJ, USA, 2010.
30. Brockett, C.L.; Chapman, G.J. Biomechanics of the ankle. *Orthop. Trauma* **2016**, *30*, 232–238. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.