Neural Prosthetic with In-sensor Shared Control

Alisha Menon

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-244
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-244.html

December 1, 2023

Neural Prosthetic with In-sensor Shared Control

by

Alisha Menon

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jan M. Rabaey, Chair
Associate Professor Rikky Muller
Professor Bruno Olshausen

Fall 2022

Neural Prosthetic with In-sensor Shared Control

Abstract

Neural Prosthetic with In-sensor Shared Control

by

Alisha Menon

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Jan M. Rabaey, Chair

With the explosive growth of wearable devices across a wide range of medical applications, the ability to monitor a broad range of biosignals becomes increasingly viable. However, these sensors face limitations in hardware resources and battery life. Low-power in-sensor intelligence can replace costly transmission of raw data streams to improve battery life. For a neural prosthetic, for example, various biosensors can be used to intelligently map the user's intended movements into prosthetic actuation. Achieving this locally can extend the lifetime of the device and reduce latency, significantly improving the user experience. This thesis leverages the emerging brain-inspired hyperdimensional computing (HDC) paradigm, which uses an inherently simple binary representation, to address this need.

The first section of this thesis explores the energy efficiency of HDC for machine learning, including a comparison against traditional ML algorithms, through design and post-layout simulation of biosignal classification ASICs. With on-the-fly generation instead of memory storage, and vector folding, the proposed architecture achieves 39.1 nJ/prediction; a 4.9x improvement over the state-of-the-art HDC processor and 9.5x over an optimized SVM processor, paving the way for it to become the paradigm of choice for in-sensor classification.

The second section of this thesis explores the use of the paradigm for robotics including the development of a novel reactive robotics algorithm with a weighted heterogeneous sensor encoding scheme that intelligently prioritizes successful behaviors, boosting the success rate in a 2-D navigation task by over 30%, even when integrated into a neural network.

The final section of this thesis pulls together the prior elements for the realization of a user-adaptive neural prosthetic with shared control. The controller recognizes the user's behaviors, predicts their next action based on habitual sequences, and determines prosthetic actuation through intelligent deliberation between the user's goal and sensor feedback-driven autonomy. With each layer designed for hardware-efficiency to enable in-sensor implementation, the system achieves an overall accuracy of 93%.

To my family, *Satish*, *Anita* and *Avish*

For their support at every step of the way.

Contents

Contents **ii**

List of Figures **iii**

List of Tables **vi**

1 Introduction **1**

2 Brain-inspired Biosignal Processor for Ultra-energy-efficient Classification **5**
2.1 Introduction 5
2.2 Implementation 9
2.3 Results & analysis 15
2.4 Alternate Implementations 21
2.5 Conclusion 32

3 Design of Reactive Robotics with High Performance & Efficiency **35**
3.1 Introduction 35
3.2 Problem Formulation 37
3.3 HDC RORB 38
3.4 Hybrid HDC and NN 40
3.5 Experimental Results 41
3.6 Conclusion 45

4 Shared Control for Neural Prosthetic Devices **47**
4.1 Introduction 47
4.2 Data Collection 48
4.3 Proposed System 51
4.4 Experimental Results 58
4.5 Summary & Future Work 67

5 Final Thoughts **68**

Bibliography **69**

List of Figures

1.1 Key operations, elements, and blocks of HDC classification, shown for a 64-electrode EMG gesture recognition task [55]. 2
1.2 Illustration of the scope of this thesis. 3

2.1 Cellular automaton *rule 90* iterative HV generation where ρ is a circular shift and $\oplus$ is a bitwise XOR. HVs are generated through an XOR of the circular right and left shift of the prior iteration. 6
2.2 Proposed HDC processor with CA *rule 90* hypervector generation. 8
2.3 HDC (a) late fusion and (b) early fusion architectures for a three-modality emotion recognition system. The late fusion architecture fuses after the temporal encoder, resulting in 3 parallel temporal encoders - one per modality. In comparison, the early fusion architecture fuses before the temporal encoder, resulting in only 1 temporal encoder. 12
2.4 Average valence and arousal accuracy for the various memory optimizations proposed as the hypervector length decreases. The data labels are shown for the most effective optimization: CA *rule 90*. 14
2.5 Impact of folding the $D = 2000$-bit HDC processor with CA *rule 90* HV generation on classification accuracy. 16
2.6 Generated layout of the proposed HDC processor using 28 nm CMOS process (without rings and pads) 17
2.7 Impact of vector folding a $D = 2000$-bit processor with 1ms classification latency on dynamic & leakage power and energy/prediction 18
2.8 Impact of vector folding a $D = 2000$-bit processor with 1ms classification latency on power consumption of each HDC block. 18
2.9 Dynamic and leakage power and energy/prediction to observe the optimal folding point when targeting 5ms classification latency. 19
2.10 Dynamic and leakage power and energy/prediction to observe the impact of folding the temporal encoder targeting 1ms classification latency. 19
2.11 Energy/prediction when running at the maximum throughput for each fold factor. 21
2.12 Key CPU operations for HDC encoding in the (a) spatial encoder (b) temporal encoder (c) associative memory. 22
2.13 Bit-serial word-parallel technique to accelerate the key HDC bundling operation. 23

2.14 Execution time and energy efficiency with transpose-popcount bundling, bit-serial word-parallel bundling, and after implementation on the vector accelerator. . . . 25
2.15 Impact of the optimizations on execution time in clock cycles for the spatial encoder, temporal encoder and associative memory. 26
2.16 Change in execution time per prediction, average power and overall energy efficiency as the number of lanes in the vector accelerator increase. 27
2.17 HV generation block with (a) ROMs vs. (b) CA *rule 90*. 28
2.18 iM and total area as channels increase with CA *rule 90* HV generator vs. ROMs evaluated with $F = 4$ and 1ms classification latency. 29
2.19 SVM architecture with systolic array 30
2.20 Energy/prediction comparison as the number of channels scales between the SVM, HDC with CA *rule 90* and HDC with ROM. 32

3.1 2-D simulated navigation environment with obstacle (Ob1-Ob4) and target goal x and y direction sensors. The robot can move in the four directions (up, down, left, right). 37
3.2 (a) Visualization of the sensor-actuator pairs generated during training of the program hypervector, and the query and clean-up done during recall in the HDC RORB algorithm. (b) A sensor hypervector is computed from the various sensors information through (c) modality-based encoding (d) directional encoding (e) constraints vs. goals. 38
3.3 The implemented models are (a) NN: input sensor values, output actuation (b) HDC-NN1: input constraint vs. goals sensor hypervector where each bit is a feature for the NN, output actuation (c) HDC-NN2: input constraint vs. goals sensor hypervector, output actuator hypervector with the HDC clean-up process. 40
3.4 The impact of sensor encoding strategy on success rate for HDC RORB. The error bars indicate standard deviation. 42
3.5 Success rates for the NN on its own, the HDC-NN1 with constraints vs. goals sensor encoding, and the HDC-NN2 with constraints vs. goals sensor encoding and actuator clean-up. Error bars indicate standard deviation. 43
3.6 Impact of hypervector dimension on HDC RORB and HDC-NN2 success rate. At 1000, HDC RORB degrades by more than 20% while the HDC-NN hybrid remains within 3% of the peak. Shaded area indicates standard deviation. . . . 44

4.1 The data collection setup integrating flexible force sensors into the EMG and accelerometer acquisition system. 49
4.2 The data collection GUI and objects for common activities of daily living. . . . 50
4.3 Multi-layer shared control scheme with (a) sensor inputs (b) user behavior recognition (c) user intent prediction (d) shared control with HDC RORB (e) actuation & feedback. 52

4.4 Task prediction of continuous motion, shown for "Screwing lightbulb". Raw data is spatially and temporally encoded, bundled across a sub-task, and classified in the associative memory. 54
4.5 Probabilistic model depicting a single slice of the DBNs, integrating error correction and a cost variable for recurring behaviors. The query favors the more reliable source of information. 56
4.6 The adopted HDC RORB algorithm with (a) the originally proposed algorithm combining all training data into a single PV, representing both the naive method and the version in which force-independent cases are pruned to prevent redundancy and (b) the added second PV for the force-independent subset of the data which results in higher recall performance. 57
4.7 EMG-based task recognition accuracy with HDC per ADL and overall for the 4 subjects. The error bars indicate standard deviation. 59
4.8 Similarity matrix of the hypervectors for each sub-tasks within the ADLs measured with cosine similarity. 60
4.9 Sub-task recognition accuracy with a time-series classifier, given an accurate ADL from the HDC classifier. The error bars indicate standard deviation. 61
4.10 Ideal accuracy of sub-task prediction using the probabilistic layer assuming correct prior sub-task classifications. 62
4.11 Sub-task prediction performance when integrating error from prior sub-task classification. 62
4.12 Sub-task prediction performance when using the error correction scheme. 63
4.13 Shared control prosthetic actuation recall accuracy for HDC RORB implemented naively, after eliminating redundancy, and with two shared control models. Shaded area shows standard deviation. 64
4.14 Simulation environment with the Shadow Dexterous hand and all the various ADL objects including a cloth for folding, a lightbulb, a jar, a pen, a comb and a cup. 65

List of Tables

2.1 AMIGOS classification accuracy comparison table 14
2.2 Comparison between the HDC architecture proposed in this work using CA *rule 90* with vector folding and the SVM developed in this work both targeting physiological signal classification. 33
2.3 Comparison of this work at 909 kHz, targeting real-time physiological signal classification, and at 455 MHz, the maximum throughput, against state-of-the-art HDC processors and biomedical classifiers. 34

Acknowledgments

I would like to start with thanking my advisor Professor Jan Rabaey. I have been very lucky to have had the opportunity to be advised by a visionary in the field. I have learned so much from him about always looking towards the future, being creative, working hard, pursuing passions and enjoying the ride while doing so. He is also an incredible mentor. I was first involved in his group after emailing him out of interest as an undergraduate student at Arizona State University. He funded me to work with his group over the summer because, as he said at the time, there is incredible value in encouraging interest in young people. I was blessed to receive a mentorship award from UC Berkeley in 2022, but if I am a mentor today it is only because of the example set by him. Jan gave me the freedom and support to pursue my interests during my Ph.D., even if it was as unconventional as involving 11 undergraduate students in my research. I am deeply thankful for his support and guidance.

I would like to thank all the professors who have given me invaluable advice and mentorship over the years, specifically Professor Rikky Muller whose confidence in my potential was sometimes greater than my own and whose door was always open to me. Also, thank you to professor Sophia Shao for inspiring my interest in hardware for machine learning and giving me the opportunity to be a part of course development for Introduction to Digital Design and Integrated Circuits. I would like to thank Professor Bruno Olshausen who gave me valuable me feedback during my qualifying exam and dissertation.

I also owe a huge thank you to my primary collaborator for the prosthetic project, Laura I. Galindez Olascoaga, who has spent the last few years on a very critical portion of the system. Beyond that, our discussions always led to new ideas and her perspective very much helped shape the direction of the project. Additionally, I want to thank prior students in Jan's group whose work my research was built on. Andy, Ali and George who all worked on the WAND device and built the original static gesture EMG data collection and classification system, creating the foundation for my work. Thank you also to Alba Rozas Cid who provided advice on the PCB modified from Ali and Andy's work to include force sensors. I would also like to thank other my group mates, especially Matthew Anderson for his mentorship. Having such a supportive group made the experience so enjoyable.

I owe a huge debt of gratitude to all the Berkeley Wireless Research Center (BWRC) and EECS department staff members including Candy Corpus, Shirley Salanio, Fred Burghardt, James Dunn, and Yessica Bravo, all of whom made it possible for me to conduct my research seamlessly. Thank you to Emily Naviasky and Keertana Settaluri for their mentorship during my first few years in the program. Their efforts to create a supportive culture at BWRC hugely impacted my experience. I would like to thank the BWRC sponsors and Semiconductor Research Corporation (SRC) for supporting my research through SONIC, TerraSwarm and CONIX centers. I also would like to thank the VIP and HYDDENN programs sponsored by DARPA and the National Science Foundation Graduate Research Fellowship Program under Grant No. 1752814 for enabling this work.

Regarding the work conducted in this dissertation, I owe a huge debt of gratitude to so many people without whom this would not have been possible. Thank you to Harrison Liew

for all of his help with the ASIC flow and RISC-V with Hwacha simulation. Thank you to Kyoungtae Lee for his help with the SVM ASIC implementation. Daniel Sun was an equal contributor on the HDC processor developed in this work. He was an incredible collaborator and I am very thankful to have had the opportunity to work with him on that project. Thank you to Melvin Aristio and Sarina Sabouri for their work on the HDC processor as well. Thank you to Adriel Tan and Meek Simbule for their significant contributions to the vectorized HDC work. Thank you to Denis Kleyko for all the innovative ideas on using HDC for TinyML applications.

Thank you to Youbin Kim and Braeden Benedict for their contributions and ideas for the 2-D navigation project. Thank you to Anirudh Natajaran who contributed to every single project discussed in this work due to his consistent involvement in my research projects starting from his freshman year. Thank you to Reva Agashe for her work on the emotion recognition algorithm development. Thank you to Abbas Rahimi for his initial perspective on the emotion recognition work. Thank you to Jennifer Ruffing for her insightful perspectives on prosthetic users that helped define the shared control direction and for her countless hours spent at BWRC on the data collection setup. Thank you to Niki Shakouri for her work designing the GUI for the prosthetic project and persistence in various research projects. Thank you to Vamshi Balanaga whose contributions to the EMG data processing, multi-layer integration, and ideas for the overall system were invaluable. Thank you to Aayush Shah for his involvement in the physical robotic hand. Thank you to Andreea Bobu who provided a template Pybullet simulation environment which became the basis of the prosthetic simulator developed in this work. Finally, thank you to Ryan Ardalan who worked so hard on implementing the Shadowhand with the JACO arm in this simulation environment and on the high-level actuations using this setup.

I was blessed to have had the opportunity to work with so many incredible undergraduate, masters and even a high school students during my Ph.D., and all from a variety of different backgrounds and experiences. These students all changed the course of my Ph.D. with their ideas and contributions. It has been truly rewarding and inspiring to work with all of them.

During the last four and a half years I have met so many amazing people and built lifelong relationships during this program. I would like to thank them all for making the past years a period that I will look back on fondly, especially Rebekah Zhao, Shreya Ramachandran, Julian Maravilla, Yuhan Wen, Ke Wang, Ekin Karasan, Rozhan Rabbani, Alfredo De Goyenche, Yue Dai, Noelle Davis, Suma Anand, Lisa Qing, Megumi Tanaka, and Yi-Hsuan Shih. Many things change, but my friendship with my best friend since 4th grade, Sehel Tahir, is one that is always a constant. Thank you to her and her whole family for their unconditional support. Also thank you to my roommate's cat, Jun, who provided endless joy over the years. Jun encouraged me to adopt a cat after my dissertation talk, who I named MOSFET - inspired by the completion of my EECS Ph.D. with a focus in integrated circuits.

Finally, thank you to my family. Somehow, during my Ph.D., we managed to squeeze in travel to Greece, Italy, Cancun, Puerto Rico, and Costa Rica. My mom is currently finishing chemotherapy, but I hope that in 2023 we will be able to return to traveling the

world together again. From the day I was born, there was not a single thing my parents wouldn't have done for me to have the opportunities that I have had. I am where I am because of them. They are the most selfless people I know and are my lifelong role models. Being back home to be there for my mom during my last semester of Ph.D. is barely a drop in the bucket to repay all of the things she has sacrificed for me to be where I am. My brother graduated with his masters in computer science from UC Davis 6 months ago, but before that he was close by and was the only one of us with a car. He made so many trips to visit me in Berkeley and was there for all the important moments, including my dissertation talk. There are no words for my gratitude to my family.

Chapter 1

Introduction

With the explosive growth of wearable devices across a wide range of medical applications, the ability to monitor a broad range of biosignals becomes increasingly viable. Wireless sensor networks can be formed with these distributed, heterogeneous biosensor nodes that monitor and communicate data wirelessly. In traditional sensor networks, all data is transmitted to a hub for processing, thus the radio is the largest consumer of power [3]. In-sensor intelligence can be used to aggregate data locally to reduce the amount of communication. By only transmitting classes or events, costly transmission of raw data streams are eliminated. The lack of consistent data streaming also provides increased data security/privacy. However, aggregation algorithms are very power-hungry which limits the benefits of this strategy. The battery life of the sensors is improved if the additional power consumption of the on-board intelligence is less than that saved from reduced data transmission. For an advanced neural prosthetic, for example, various biosensors are used to map the user's intended movements into prosthetic actuation. The ability to achieve this locally can extend the lifetime of the device and also reduce latency; both significantly improve the user experience.

Hyperdimensional computing (HDC) is an emerging brain-inspired paradigm that has the potential to address the needs of wireless biosensor nodes. It is based on manipulating large patterns of information in the form of very long binary vectors [28]. If we randomly generate one very long vector, with each bit randomly selected to be a 1 or a 0, and we randomly generate a second very long vector, the chances that these two vectors are going to be the same is very unlikely. These two vectors are considered orthogonal, or uncorrelated, occupying completely different spaces within the "hyperdimensional space". The randomness results in the orthogonality, and as the dimension of the vectors increase, more and more random orthogonal vectors can fit within the available hyperdimensional space without overlap. To maintain this property to a sufficient extent, dimensions greater than 2000 are usually selected for the vectors, referred to as hypervectors (HVs), as anything smaller compromises on performance/accuracy.

This property of randomness and orthogonality, and three simple binary operations - bundling (bitwise majority count), binding (bitwise XOR), and permutation (cyclical shift) - can be used to achieve high-accuracy classification. Figure 1.1 shows the key HDC operations,

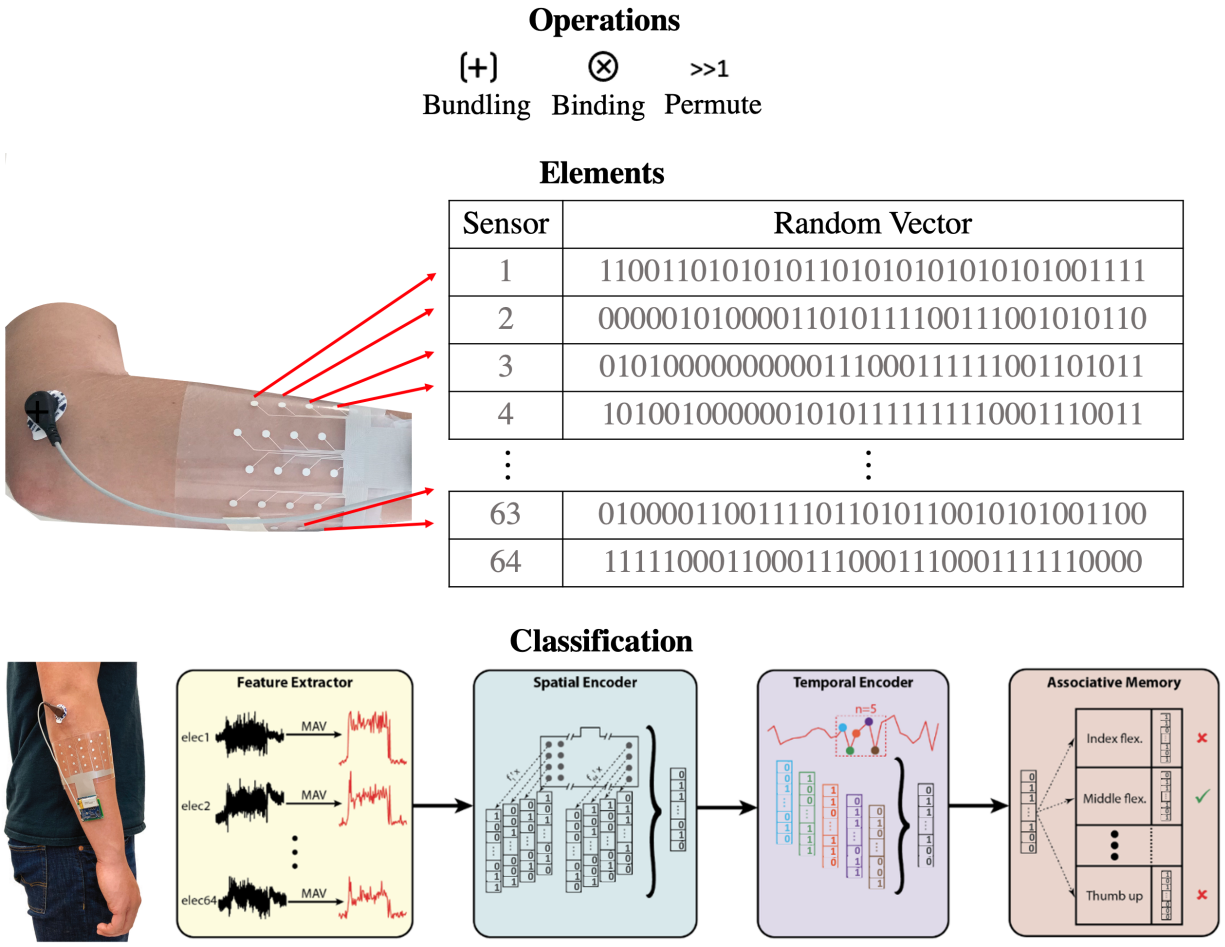


Figure 1.1: Key operations, elements, and blocks of HDC classification, shown for a 64-electrode EMG gesture recognition task [55].

elements and classification blocks for an example EMG gesture recognition task. The first step is assigning each electrode channel to a random hypervector; the full set of which is referred to as the item memory. Feature extraction is done on the signals measured on each electrode, such as a mean absolute value over a certain window [55]. Information about the electrode's feature value for the window at time t is projected onto its random hypervector by scaling the corresponding item vector by it. For example, if the first electrode has a value of 50mV, the full first item memory hypervector is scaled by 50. All the scaled vectors are then bundled: summed together into one hypervector and then thresholded to assign each bit to either a 1 or 0 and return to the binary representation. Because the mean absolute value changes for each window, the output of the spatial encoder varies over time. A temporal encoder is used to capture patterns that may arise over several windows by permuting spatially encoded hypervectors for a set of n windows according to occurrence

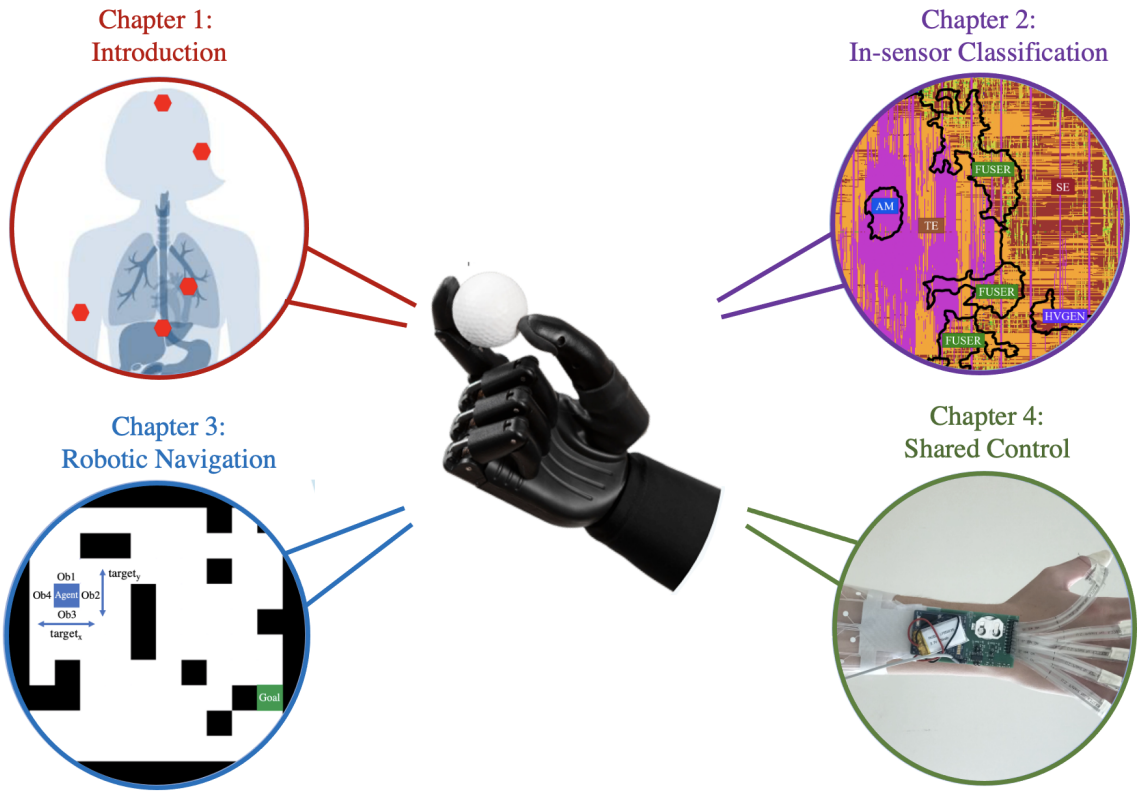


Figure 1.2: Illustration of the scope of this thesis.

and then binding. For the gesture recognition example, during training, the subject will be asked to demonstrate gestures such as middle flexion. The temporal encoder output for that data will represent EMG patterns for a middle flexion; this gets stored as the representative hypervector for that class in the associative memory. During inference, if the user is doing a middle flexion, the output of the temporal encoder once again contains that EMG pattern. The pattern is classified by checking for correlation with the trained hypervectors which would indicate similarity to the middle flexion.

This powerful pattern recognition algorithm has demonstrated superior performance over other state-of-the-art algorithms for applications such as seizure detection [6], gesture recognition [53], EEG error-related potentials classification for brain-computer interfaces [64] and emotion recognition [7]. HDC even shows high performance under constraints such as limited training data [54] or noise from sensor misplacement, and provides rapid model updates [53]. These properties are especially useful for biosensors where the training data is user-specific and the sensors are vulnerable to displacement between or during extended usage [53]. The paradigm has also been used for probabilistic queries [19], and reactive robotics [42], demonstrating its versatility. However, the primary motivation for the growing usage of this paradigm is its inherently simplicity which promises highly energy efficient intelligence due to its binary representation and minimal models.

The full scope of this thesis is shown in Figure 1.2[1]. Towards a wireless biosensor node with improved sensor lifetime through ultra-low-power on-board intelligence, the first section of this thesis introduces a highly energy-efficient HDC processor. Prior HDC processors are dominated by costly hypervector memory storage, which grows linearly with the number of sensors. However, when observing biosignals such as EMG and EEG, large arrays of sensors are used to collect spatial information [53] which significantly limits the efficiency of this paradigm for these applications. To address this, the memory is replaced with a light-weight cellular automaton for on-the-fly hypervector generation. The use of this technique is explored in conjunction with vector folding for various real-time biosignal classification latencies in post-layout simulation on an emotion recognition dataset with >200 channels. The proposed architecture achieves 39.1 nJ/prediction; a 4.9x energy efficiency improvement, 9.5x per channel, over the state-of-the-art HDC processor. The processor is compared against an optimized support vector machine (SVM) processor designed in this work for the same use-case. HDC is 9.5x more energy-efficient than the SVM, paving the way for it to become the paradigm of choice for high-accuracy, on-board biosignal classification.

The second section of this thesis explores how robot navigation tasks can leverage the advantages of the hypervector representation, including robustness to noise, hardware efficiency, and high performance with limited training data - all as useful for robotics applications as for biosensing. The paradigm enables behavioral prioritization through a weighted encoding of heterogeneous sensor information. Experiments over 100 trials in each of the 100 randomly generated obstacle maps demonstrate that the proposed weighted sensor encoding scheme boosts the success rate of the navigation task by over 30% compared to an unweighted sensor encoding, even while using a neural network. A hybrid scheme using the HDC weighted scheme at the input of a deep feed-forward neural network achieves the highest success rate. The hybrid scheme furthermore is more robust when reducing the HDC dimension by 50%. However, the simple HDC implementation remains the most hardware efficient, making it desirable for resource-constrained systems.

The final section of this thesis pulls together the elements described in prior sections to address the burden and limitations of prosthetic devices. There is increasing interest in shared control for assistive robotics with adaptable levels of supervised autonomy. In this work, we present a user-adaptive multi-layer shared control scheme for control of assistive devices. The system leverages the advantages of HDC for classification, probabilistic queries, and reactive robotics to execute actuation based on the user's goal while alleviating the burden of fine control. A multi-modal dataset is collected for various activities of daily living; the controller first recognizes the user's most recent behaviors, then predicts the user's next action based on their habitual action sequences, and finally, determines actuation through shared control with HDC-based reactive robotics which intelligent deliberates between the predicted action and sensor feedback-based autonomy. Each layer independently achieves >92% accuracy. Through error correction, the system is able to eliminate and correct accumulated mistakes after integration of all the layers to achieve an overall accuracy of 93%.

[1]Zeus Bionic Hand showcased as an example prosthetic hand

Chapter 2

Brain-inspired Biosignal Processor for Ultra-energy-efficient Classification

2.1 Introduction

Specialized biosignal classifiers have been designed to efficiently classify biosignal information streams using various algorithms for a variety of applications. Prior art includes application-specific integrated circuits (ASICs) for: EEG-based seizure event classification using support vector machines [21] and logistic regression with stochastic gradient descent [9], decision trees for intracranial EEG-based brain state classification [59], and ECG-based cardiac arrhythmia classification through an artificial neural networks [82]. ASIC classifiers have also been designed to efficiently handle multiple types of biosignals using decision trees [70] and a reconfigurable neural network [39]. All of these works aim for energy efficient usage in ultra-low-power multi-channeled biosignal monitoring devices.

The HDC paradigm has been efficiently implemented in hardware as demonstrated by several examples of specialized processors [13], [16]. However, the energy efficiency has been limited for applications with many channels, such as biosignal monitoring, because HDC processors require as many pseudo-orthogonal channel identification HVs as number of input data sensor channels. Datta et al. synthesized a generic HDC processor for which the channel identification HV storage, using ROMs, contributed the most (over 40%) towards total power consumption [13]. This demonstrates the large cost of the necessary hypervector storage. Eggiman et al. generated the channel identification HVs and the adjustably pseudo-orthogonal feature vectors with a similarity manipulator during the encoding process to alleviate this issue [16]. This led to a 3x energy-efficiency improvement in post-layout simulation over [13]. However, this process is still overly complex, especially for applications with sequential channel access patterns. Therefore, the ability for HDC to efficiently classify for applications with large numbers of channels remains limited.

Prior work has explored the usage of in-memory compute to do end-to-end HDC language classification in the item memory vector storage through use of carbon nanotube FETs and

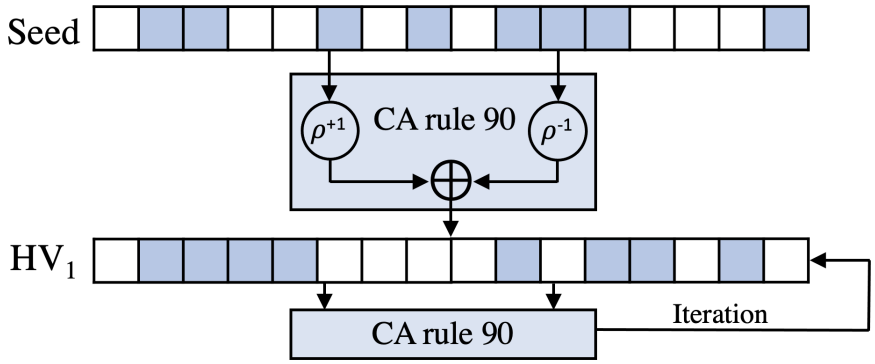


Figure 2.1: Cellular automaton *rule 90* iterative HV generation where ρ is a circular shift and $\oplus$ is a bitwise XOR. HVs are generated through an XOR of the circular right and left shift of the prior iteration.

Resistive RAM (RRAM, an emerging technology for analog memory storage) for bundling and random projection [79]. An alternate implementation focuses on the associative memory, which stores trained HDC language classes, by using an XNOR-based RRAM content-addressable memory to measure similarity between an input query and stored patterns [20]. A third in/near-memory HDC system uses two planar memristive crossbar engines (one for binding, the other for associative memory search) in combination with nearby digital CMOS for bundling applied to language and news datasets [29]. However, these techniques depend on specialized memory elements in order to reduce the burden of the costly hypervector storage, instead of reducing the storage itself.

Algorithm optimizations suggest an alternate and less specialized solution. HDC demonstrates no accuracy degradation when re-using feature HVs across all channels and sensor modalities, making the total number of hypervectors that needs to be stored for feature representation constant regardless of channel count [46]. This is possible because pseudo-orthogonality is maintained through the unique channel identification HVs which are equal in count to number of channels being classified. Prior work has shown that these can be generated on-the-fly using a cellular automaton (CA) instead of being stored in memory [32].

A cellular automata consists of a grid of cells, each of which evolve with complex behavior over time through a set of discrete computations using the state of the current and nearby cells [33]. CA *rule 90* assigns the next state in a method shown to be equivalent to an XOR of the two nearest cells [33]. This involves a single seed hypervector, a cyclical right and left shift, and an XOR across the resulting vectors, visualized in Figure 2.1. Seed vectors of dimensions over 24 have been shown to generate new degrees of freedom, new pseudo-orthogonal vectors, for more than 10^3 iterations before saturating [32]. Hypervectors, with thousands of cells in the grid, provide a linearly larger number of random iterations; sufficient for the majority of applications. Prior work has shown that HDC demonstrates no performance degradation with CA *rule 90* channel vector generation [46]. The architecture proposed in this work aims to take advantage of this.

The proposed designs are demonstrated with the AMIGOS emotion recognition dataset due to its large number of channels which allow for exploration of scaling up the design's input channels [12]. This dataset has 214 channels of multi-modal physiological feature information used to classify human affective states based on two neurophysiological systems, valence (pleasure vs. displeasure) and arousal (alertness). Discrete emotions can be described as a linear combination of these two systems [63], reduced to high and low arousal and valence values for emotion recognition. Previous work on the AMIGOS dataset includes various other machine learning algorithms such as Fisher's linear discriminant with Gaussian Naive Bayes (F1 score of 57.8%) [12], extreme learning machines (83.4%) [71], extreme gradient boosting (74.3%), and a common machine learning algorithm, a support vector machine (SVM), which obtained 67.15% average arousal and valence accuracy [76].

Comparison of HDC hardware against common machine learning algorithms is limited, to the best of our knowledge, to a single prior work which shows that HDC computing achieved 2x faster execution and lower power at iso-accuracy on an ARM Cortex M4 compared to an optimized SVM [56]. However, that architecture used hypervectors with a dimension of 200 bits. In order to maintain pseudo-orthogonality, the key to HDC's high classification accuracy, prior HDC hardware implementations have instead suggested dimensions in the range of 1000 to 2048 bits [53], [13], [16], [46]. Hence, the HDC implementation used for prior comparison is not viable for the majority of applications due to the large accuracy trade-off that results in better hardware performance, but non-competitive algorithm performance [56]. In order to truly make HDC a desirable choice over alternative machine learning algorithms, the hardware efficiency of HDC must be improved at dimensions of 1000-2000 bits to achieve superior performance in both hardware and classification accuracy.

In this work, a 2000-bit HDC processor is designed for always-on, on-board classification, especially for large numbers of channels, to increase the monitoring lifetime of wearable biosignal sensor arrays. CA *rule 90* is combined with vector folding to increase energy efficiency with additional analysis and improved performance at maximum throughput over [44]. Finally, an optimized support vector machine architecture is designed for the same use-case, formalizing the first realistic hardware comparison between HDC and a common machine learning algorithm, explored at a variety of channel counts. All architectures used in this work are demonstrated with emotion recognition on over 200 channels and 3 sensor modalities. This chapter is organized as follows: Section 2.2 proposes an HDC architecture using CA *rule 90* including the development of the HDC algorithm for the emotion recognition dataset. Section 2.3 focuses on results and analysis of the key techniques. Section 2.4 develops alternative architectures to compare HDC's energy efficiency against a traditional machine learning algorithm. Section 2.5 concludes with a summary, comparison against state-of-the-art and potential future directions.

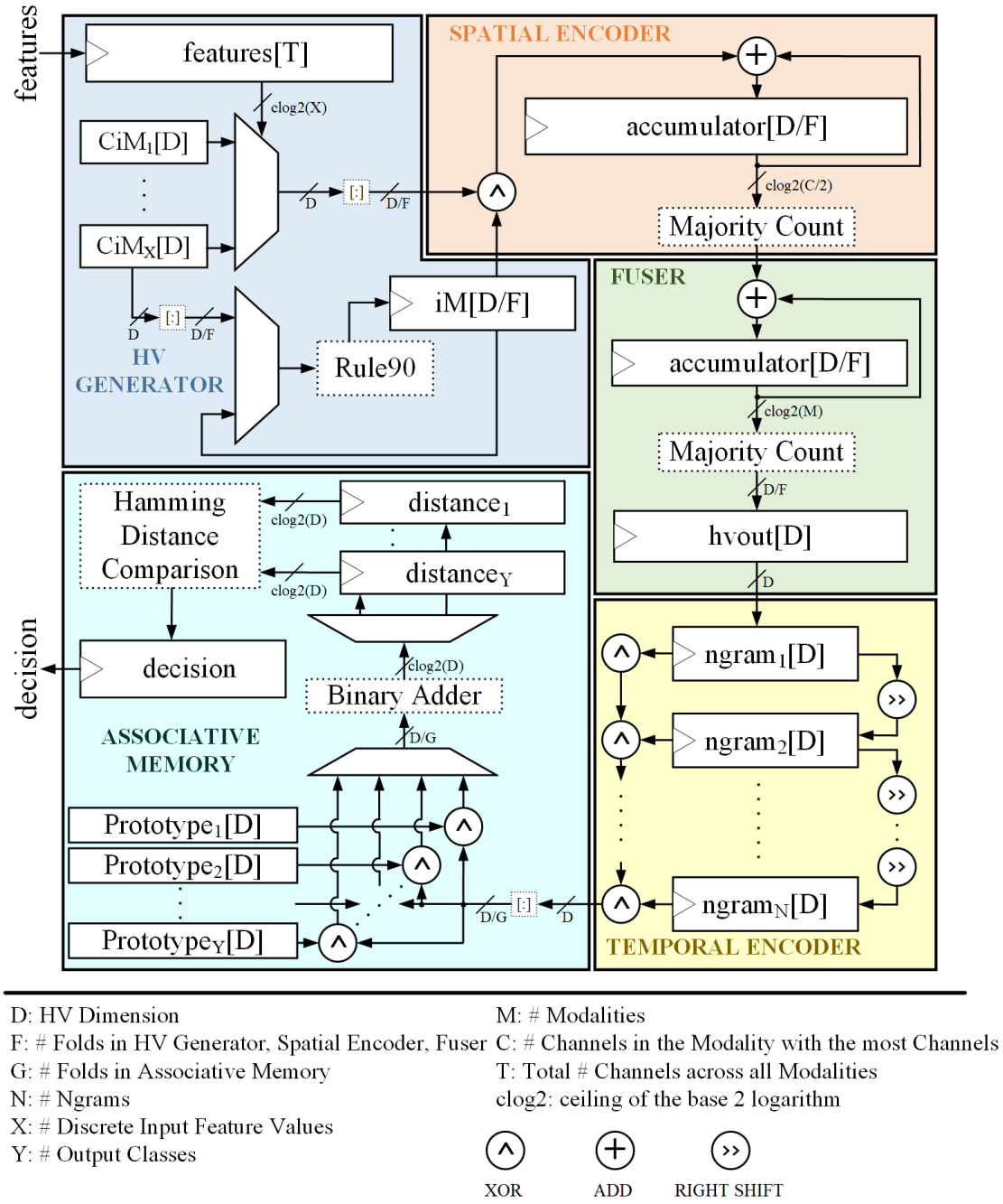


Figure 2.2: Proposed HDC processor with CA *rule 90* hypervector generation.

2.2 Implementation

The HDC encoding architecture is primarily composed of four blocks [54], [7]. The first is mapping into the HD space. Each channel, or item, is assigned a unique pseudo-random HV. The full set of channel identification HVs is called the item memory (iM). A continuous item memory (CiM) is used for encoding of feature values. This comprises a set of HVs that gradually increase in hamming distance (a metric for HV orthogonality) and corresponds to a range of discrete feature values. The feature resolution is set by the number of HVs in the CiM. In the proposed architecture, shown in Figure 2.2, tie-high/tie-low cells are used to store the CiM vectors. These are re-used across all of the channels. The iM HVs, however, are generated on-the-fly instead of being pre-generated or stored in memory. This is done in the HV generator using CA *rule 90*. If $\mathrm{HV_n}$ is the hypervector state at step n, and ρ is the cyclical shift operation (+1 for right, -1 for left), then $\mathrm{HV_{n+1}}$ can be generated by:

$$\mathrm{HV_{n+1}} = \rho^{+1}(\mathrm{HV_n}) \oplus \rho^{-1}(\mathrm{HV_n}) \tag{2.1}$$

These operations are vectorizable and computationally minimal. The iM generation process is further minimized by using the final CiM HV as the seed. Mapping into HD space involves a binding of the channel identification HV generated with CA *rule 90* and its corresponding feature value HV selected from the CiM.

The second block is the spatial encoder which accumulates the mapped vectors across all channels. To minimize leakage power, only one channel is accumulated per cycle. Once accumulation of all the channels in a sensor modality is completed, a majority count is performed to generate a single HV for the modality. This is accomplished by thresholding the accumulator. Then, the fuser bundles these modality HVs providing an equal contribution from each of the modalities. The resulting output HV is an observation across all channels. This is fed to the next block, the temporal encoder, as an n-gram, a single sample in a sequence of n samples for which order is preserved through permutation of each hypervector. This encoding scheme captures time-varying signals, enabling recognition of temporal patterns in time-series data. The final block, the associative memory (AM), stores prototype HVs for each data class. Classification is performed through hamming distance comparison between the incoming HV and each of the stored prototype HVs. The class with the least distance is predicted.

CA rule 90 makes the design flexible to changes in channel count. Since only a single iM vector is stored for the current channel and a new one is generated each cycle, the HV generation architecture does not expand with the number of channels. Instead, the HV generator and spatial encoder take additional cycles to complete. For sequential channel access patterns, each additional channel adds only a single additional cycle. For random channel access patterns, the HV generator will need to restart with the seed and iterate until the specified channel identification HV is generated.

Vector Folding

In this work, vector folding is implemented and the impact on optimizations and trade-offs in area, energy, and classification latency are evaluated for the HDC architecture. Operations within the HDC algorithm are frequently performed on very large vectors. When implemented on hardware, a large amount of fanout is induced on control signals that drive registers across these wide hypervectors, resulting in an increased amount of leakage power dissipation as more buffers are inserted to meet hold timing requirements. To mitigate this problem, the HV datapath can be folded to a fraction of the hypervector dimension. The remaining hardware is then time-multiplexed fold-by-fold. This technique reduces the datapath register count, resulting in a decrease in leakage power and area, while incurring a cost in classification latency. An analysis of these trade-offs is presented in Section 2.3. In the proposed architecture, a hypervector of dimension D is divided into F folds for the HV generator, spatial encoder, and fuser and G folds for the associative memory.

Given this, the iM vectors are generated as follows. The first D/F section of the last CiM vector is used to generate the initial fold F_1 of the first iM. The first fold of each subsequent iM is then generated by simply applying CA *rule 90* iteratively. This process is then repeated for the subsequent folds of each iM, one after another. This entire process is time-multiplexed such that in hardware, only a single iM fold of dimension D/F needs to be stored per cycle, minimizing the footprint of iM generation within the HV generator. However, as the width of the CA reduces, the degrees of freedom – the number of pseudo-random *rule 90* iterations – also scales linearly downward, resulting in a classification accuracy degradation below a fold width of approximately 20 bits, equivalent to 100 folds [32]. This method is applicable until the number of folds is above this point, as shown in section 2.3.

The benefits of vector folding are most prominent in the spatial encoder and fuser, where the accumulator registers within each block can be decreased by a fold factor of F, as observed in Figure 2.2. The temporal encoder can be folded to further reduce the size of the n-grams to a width of D/F. Permutations of each D/F section of the n-grams are then binded and sent to the associative memory for comparison. However, n full spatial and fuser encodings for the fold need to be performed to generate these n-grams. As a result, folding in the temporal encoder requires re-generation of $n-1$ n-grams from prior input samples per fold, increasing latency by a factor of n cycles. In addition, the HV generator would need to buffer n times the amount of features to include the previous $n-1$ feature samples in addition to the current one. On the other hand, the D-width hvout register at the end of the fuser can be removed. This eliminates a main source of high fanout in the design which occurs between control signals in the fuser and the 2000-bit register. The tradeoff between the dynamic and leakage power as a result of folding this block are explored in Section 2.3.

Finally, for the majority of applications, the number of channels is greater than the number of classes. Hence, the AM takes less cycles to execute than the spatial encoder. Even without folding, the number of cycles within the sequential spatial encoder scales with the number of features, while the AM takes only 4 cycles across 4 distance comparisons. As a result, the AM can operate with considerably more folds than the spatial encoder and be

executed in parallel with the rest of the HDC datapath. This pipelining allows the spatial encoder to begin processing the next classification while the AM is still performing distance computation with the current classification. In this work, a separate fold factor, G, is used for the AM, reducing the footprint of hamming distance computations.

Simulation Dataset and Algorithm

This processor is simulated with the AMIGOS dataset from [12] which contains 214 features across 3 modalities: GSR (Galvanic Skin Response: 32 channels), ECG (Electrocardiogram: 72 channels) and EEG (Electroencephalogram: 105 channels). The features were measured for 33 subjects as they watched 16 videos [12]. Each video for each subject was classified to have either led to a positive or negative emotion (valence), and the strength of the emotion was classified as either strong or weak (arousal). From the 3 sensor modalities, Wang et al. selected these 214 time and frequency domain features relevant to accurate emotion classification [76]. With a total of 214 input features, this dataset was especially suited towards demonstrating energy efficiency of the proposed processor as the number of channels scales to large quantities.

The features used include GSR skin response/conductance and skin conductance slow response, ECG heart rate spectral power, variability and heart rate time series, and EEG average power spectral density and asymmetry of theta band, alpha band, beta band, and gamma band. The data for all 33 subjects was appended and a moving average of 15 seconds over 30 seconds was applied. The signals were scaled to be between -1 and $+1$ to meet the HDC encoding process and downsampled by a factor of 8 for more rapid processing and usage of the HDC classification algorithm. Previous work uses the leave-one-subject-out approach to evaluate performance, this was also implemented to evaluate the algorithm in this work [12], [76], [71].

Given the largest modality of 105 features, 6-bits is required to determine a majority across the EEG modality for the saturating counter within the spatial encoder accumulator. The classification result is a binary valence and a binary arousal decision through comparison with four prototype vectors within the associative memory. Transitionary ngrams (those with samples from both classes) were excluded from training and testing. Because emotion recognition involves various sensor modalities, it requires sensor fusion. Previous HDC sensor fusion implementations fused after the temporal encoder [7], but in this work, an early fusion approach is taken, which fuses the modalities directly after the spatial encoding process. Therefore, this architecture requires only a single temporal encoder as opposed to one per modality, as shown in Figure 2.3. This reduces the parallel encoding paths while still allowing each modality to be weighted equally instead of by number of features. The temporal encoder was tuned and an optimal n-gram of 3 feature windows was selected.

Classically, the encoding of 214 features would requires advance storage of 214 iM hypervectors with 2 feature projection hypervectors per iM vector (each feature value is encoded through a hypervector representing a positive value or hypervector representing a negative value) totalling to 642 hypervectors that need to be stored in the item memory. Various

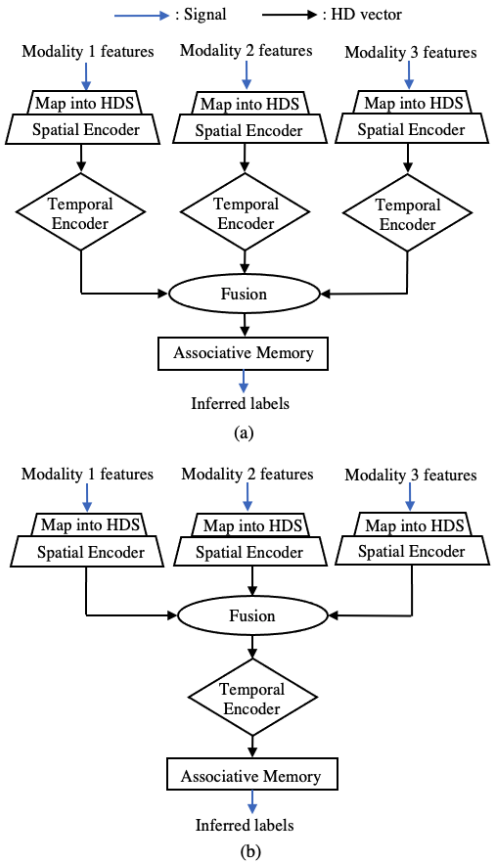


Figure 2.3: HDC (a) late fusion and (b) early fusion architectures for a three-modality emotion recognition system. The late fusion architecture fuses after the temporal encoder, resulting in 3 parallel temporal encoders - one per modality. In comparison, the early fusion architecture fuses before the temporal encoder, resulting in only 1 temporal encoder.

alternate memory optimizations were explored alongside CA *rule 90* to ensure equivalent algorithm performance while using CA *rule 90*.

In the spatial encoder, the iM vector and the FP vector are bound together to form a unique representation containing feature information that is specific to a feature channel. However, both the iM and FP vectors do not need to be unique to the feature channel in order to generate a unique combination of the two. The binding operation will inherently create a vector different, and pseudo-orthogonal to both of its inputs. Therefore, as long as one of these inputs is different for a specific feature channel, the spatially encoded feature channel vector will be unique. Using this idea, a set of optimization parameters were developed:

'iM vectors constant per modality': the iM is replicated across the various modalities. If, between each modalities, the FP vectors are different, then orthogonality and input feature channel uniqueness are maintained even if the iM is the same.

'FP constant per feature channel': though the iM is now the same between each

modalities, each feature channel within a modality still has a unique iM vector. Therefore, it is possible to re-use the same FP vectors for every feature channel within a modality. This requires maintaining 2 unique FP vectors for each modalities, and unique iM vectors within a modality.

'Combinatorial pairs': taking this combinatorial binding strategy to its limit, the 2-input binding operation can be used to generate many unique vectors from a smaller set of vectors by following an algorithmic process. Each feature channel requires a distinct set containing an iM vector, and two FP (positive & negative) vectors: {iM, PFP, NFP}. If the vectors for feature channel 1 are {A, B, C}, then the bound pairs that could result from spatial encoding (iM $\oplus$ PFP or iM $\oplus$ NFP) are:

- A $\oplus$ B
- A $\oplus$ C

B $\oplus$ C will not occur because they are both FP vectors. However, it is a unique pairing that could be re-used for another channel. For example, the set for feature channel 2 could be: {B, C, D}. The encoding process would use the following pairings:

- B $\oplus$ C
- B $\oplus$ D

This re-use strategy is the key to saving memory; it can be applied across all channels using a bank of the minimal required vectors.

Each vector can be paired with every other vector only once in order to maintain orthogonality and paired uniqueness across all feature channel. For a feature channel, one vector (the iM) must have two other available vectors (PFP and NFP) to pair with. With $\lfloor x \rfloor$ defined as the floor function of x, the following equation can be used to calculate the total feature channels, TFC, possible given a bank of v vectors:

$$\text{TFC} = \sum_{n=1}^{v-2} \left\lfloor \frac{v-n}{2} \right\rfloor \tag{2.2}$$

The formula can be derived by looping through each vector in the vector bank and sequentially grouping it with pairs of the other vectors.

'Rule 90 generation': implementation of CA *rule 90* trades off vector storage for vector generation. Two vectors are used for the PFP and NFP vector used for all modalities. These would be maintained throughout training and inference. However, the rest of the iM vectors would be generated on-the-fly for each feature channel during the encoding process, requiring no additional vector storage.

The performance of each of the memory optimization as the hypervector dimension decreases is shown in the Figure 2.4. The final result using CA *rule 90*, at a hypervector dimension of 10,000, was compared against prior work in Table 2.1.

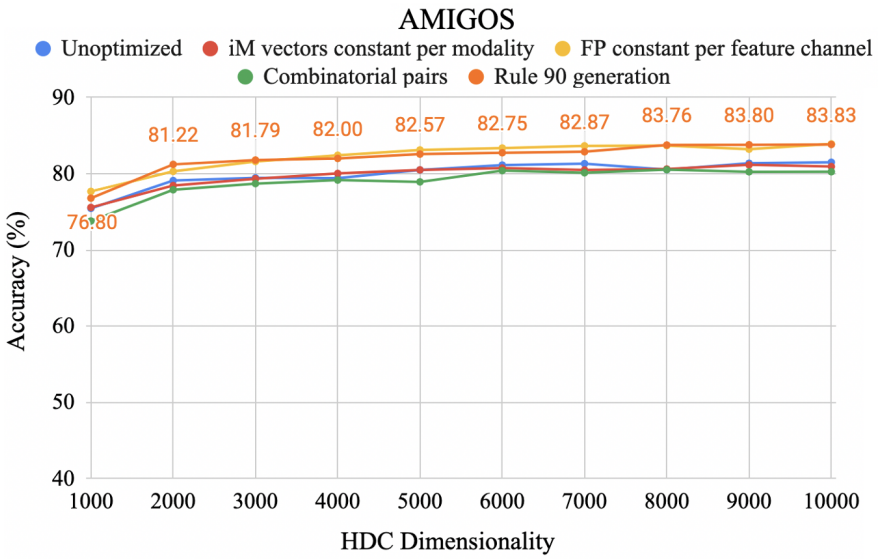


Figure 2.4: Average valence and arousal accuracy for the various memory optimizations proposed as the hypervector length decreases. The data labels are shown for the most effective optimization: CA *rule 90*.

Table 2.1: AMIGOS classification accuracy comparison table

Method	**HV vs. LV (%)**	**HA vs. LA (%)**
GaussianNB[1] [12][7]	57	58.5
SVM [76]	68.0	66.3
ELM [71]	83.9	82.8
XGB [76]	80.1	68.4
HDC late fusion [7]	83.2	70.1
HDC early fusion	87.1	80.5

The performance of various memory optimizations were explored and shown in Figure 2.4. HDC depends on near-orthogonality between different data streams and feature values to ensure that samples from different classes that vary in these ways are encoded into sufficiently orthogonal class vectors. Each optimization reduces the total number of unique vectors that need to be stored in advance, however, there was no decrease in accuracy on AMIGOS between the most unoptimized and most optimized. CA *rule 90* provides a >98% decrease in memory storage while maintaining high performance, making it a viable solution HDC architectures. It maintains the properties required for successful hyperdimensional computing and therefore could generalize to other applications, and will be particularly useful for those with many, varied streams of input information. There was actually an average increase of ∼2.3% at a dimension of 10,000 when using CA *rule 90*; this accuracy change may be attributed to the random element of HDC vector initialization/generation which may result in either beneficial or detrimental random patterns.

Table 2.1 demonstrates the overall improvement in performance on the AMIGOS dataset as compared to the HDC late fusion algorithm, even with CA *rule 90*. The boost in accuracy may come from the fact that different modalities may have different temporal behavior, which may lead to different optimal n-grams. For late fusion, an n-gram of 4 was used for all modalities without individual tuning. For early fusion, an optimal n-gram of 3 was selected for the fused modalities temporal behavior, improving the overall performance. The early fusion method requires tuning of only a single temporal encoder and still achieves higher accuracy even with reduction of the overall encoding complexity. This indicates the potential, benefits, and feasibility of early fusion encoding processes in HDC. Information is retained in the high-capacity vectors even with only a single encoding path after the spatial encoder. Compared to state-of-the-art for this dataset, as shown in Table 2.1, HDC early fusion with CA *rule 90* performed better on average than than GaussianNB[1], SVM, XGB, and ELM. The following sections deeply explore the hardware benefits of the HDC architecture using the proven CA *rule 90* technique and also compare against an SVM hardware implementation.

Simulation Parameters and Tools

Modern recording systems for biological signals use sampling frequencies typically ranging between 200 Hz to 1 kHz. Therefore, new data is sampled every 5ms - 1ms. In this work, the systems were implemented and analyzed to generate a new classification for each incoming data sample targeting real-time classification latencies of 1ms and 5ms, aligning with these typical sampling rates [62].

All designs introduced in this work were implemented in the TSMC 28nm HPM technology. Development was facilitated with the HAMMER framework [75]. CA *rule 90* designs used 0.8V VDD supply, and ROM designs used 0.9V VDD supply, both at the TT corner of 25°C. For minimal leakage power given the low-frequency use-case, only RVT/HVT cells were utilized during synthesis with Cadence Genus. An area utilization of 70% was targeted for place & route with Cadence Innovus. Power measurements were obtained using Cadence Voltus using post place & route Synopsys VCS gate-level simulation, back-annotated SDF delays, and SPEF parasitic resistances and capacitances at the TT corner.

2.3 Results & analysis

As shown in Figure 2.5, generating item memory HVs with a folded cellular automaton shows virtually no impact on accuracy down to a fold width of 20 bits with fold factor $F = 100$. Fold factors larger than this lead to an accuracy of ∼50%. These results align with prior work on maximum degrees of freedom as CA *rule 90* grid size varies; 20 bits provides fewer than 10 pseudo-random iterations, while even 25 bits provides more than 1000, sufficient for most applications [32].

[1]F1 score. Accuracy value not available.

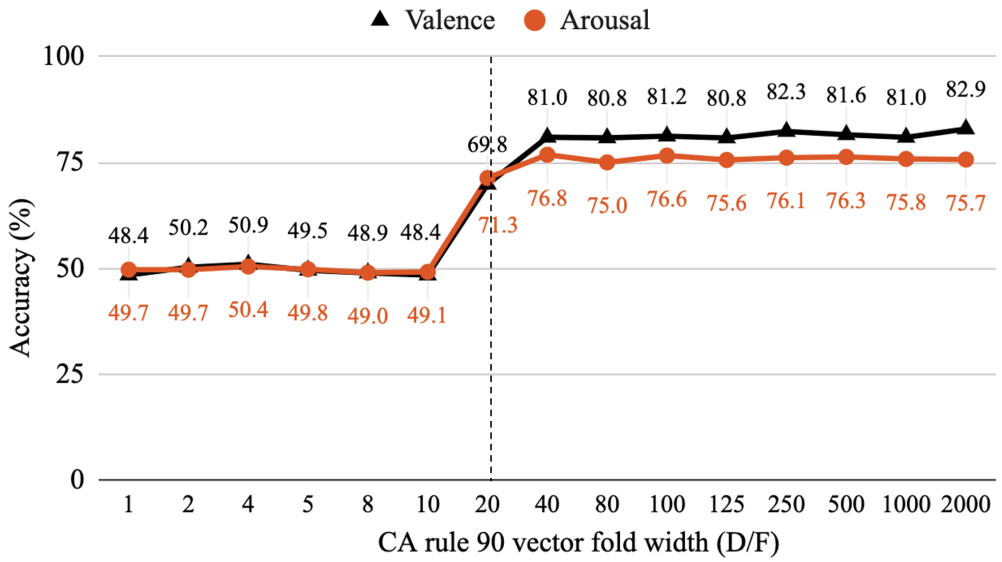


Figure 2.5: Impact of folding the $D = 2000$-bit HDC processor with CA *rule 90* HV generation on classification accuracy.

The design was benchmarked across a range of fold factors while adjusting the clock frequency to realize a 1ms classification latency. The optimal fold factor minimizes power dissipation. The datapath width and accumulator register size in the design decrease as the fold factor increases. However, to meet the target latency, a faster clock frequency, and hence higher switching rate, is required. Hence, the dynamic power increases with the decrease in leakage power. As shown in Figure 2.7, the optimal occurs where the dynamic and leakage power intersect. At 4 folds, ($F = 4$, $G = 200$), the energy efficiency of the design is 39.1 nJ/prediction.

Figure 2.6 shows the generated layout for this design at 4 folds. The minimal size of the HV generator, especially as compared to other blocks, is notable. The hypervector flow is also visible, as exemplified by the HV generator which connects to input channels around the chip, and is encompassed by the spatial encoder. The fuser is wedged between the spatial encoder and temporal encoder, with the small, folded associative memory sitting within. The spatial encoder is significantly reduced due to vector folding, but it remains the largest block at a fold factor of 4.

The impact of folding on the energy consumption of each block is shown in Figure 2.8. The HV generator finds the optimal somewhere between 2 and 4 folds, similar to the rest of the design. The spatial encoder benefits the most from folding. Despite the higher clock frequency, the savings in leakage power from folding the accumulator register is massive. The resulting reduced fanout of control signals in the spatial encoder contributes as well. The associative memory is folded at a faster rate than the other blocks to process in parallel with the spatial encoder and keep the total cycle count down. This allows for a reduced clock frequency across fold factors which helps to keep the dynamic power down in other blocks.

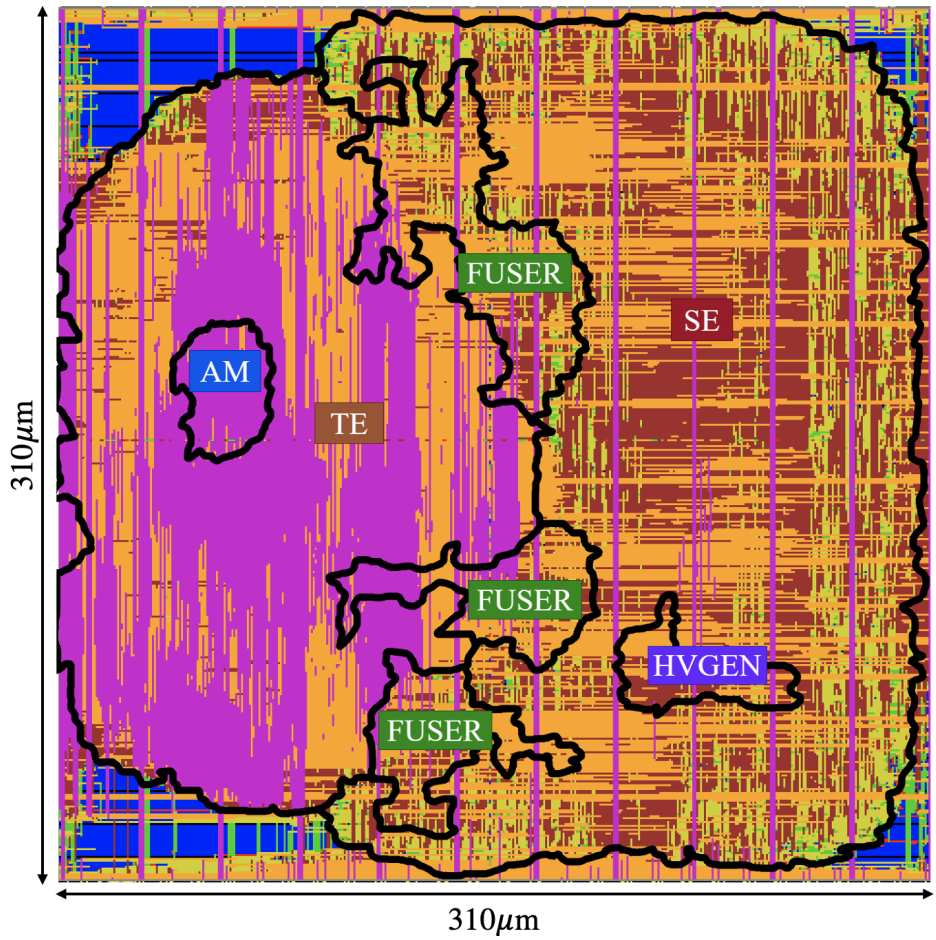


Figure 2.6: Generated layout of the proposed HDC processor using 28 nm CMOS process (without rings and pads)

However, as a result the AM grows slightly as the fold factor increases due to the increased switching rate.

The fuser finds the optimal at 2 folds after which point the dynamic power resulting from the more frequent switching of the same block causes the overall power of the block to increase. The fuser also has an unfolded output register at the input of unfolded temporal encoder that doesn't benefit from the savings in leakage power but suffers from the increase in dynamic power. Similarly, the temporal encoder power consumption grows as folding occurs because it is left unfolded but switches at a much higher rate as each fold makes its way through the block. Overall, folding improves power consumption in the spatial encoder the most significantly. The other blocks see benefits until between 2-4 folds after which point the increased dynamic power due to the faster clock frequency to meet the 1ms target latency outweighs the savings in leakage power. The optimal fold point of 4 occurs as a result of the balance between the benefit to the spatial encoder and the cost that starts to accumulate in

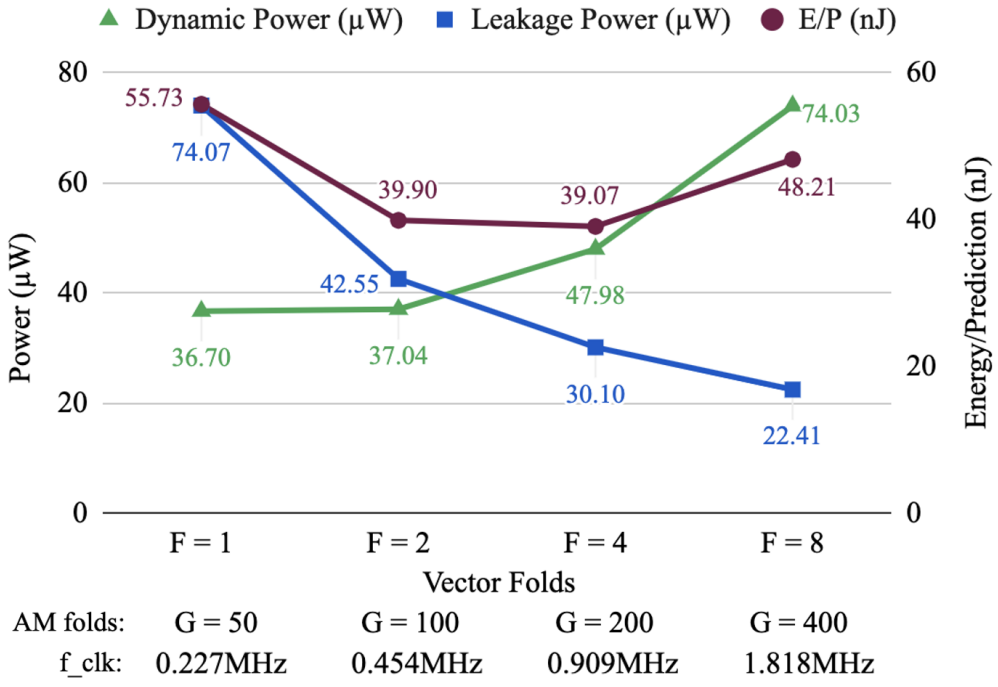


Figure 2.7: Impact of vector folding a $D = 2000$-bit processor with 1ms classification latency on dynamic & leakage power and energy/prediction

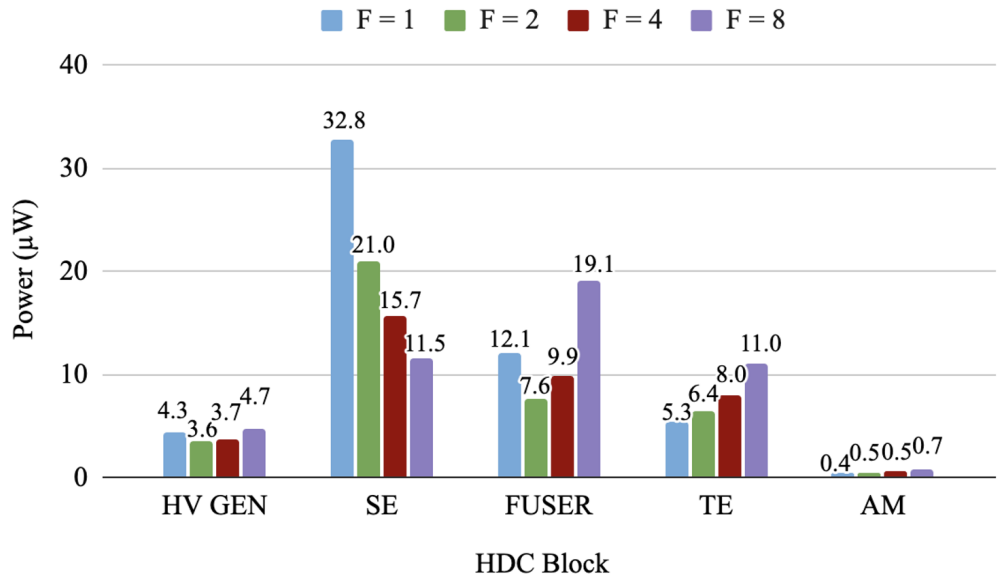


Figure 2.8: Impact of vector folding a $D = 2000$-bit processor with 1ms classification latency on power consumption of each HDC block.

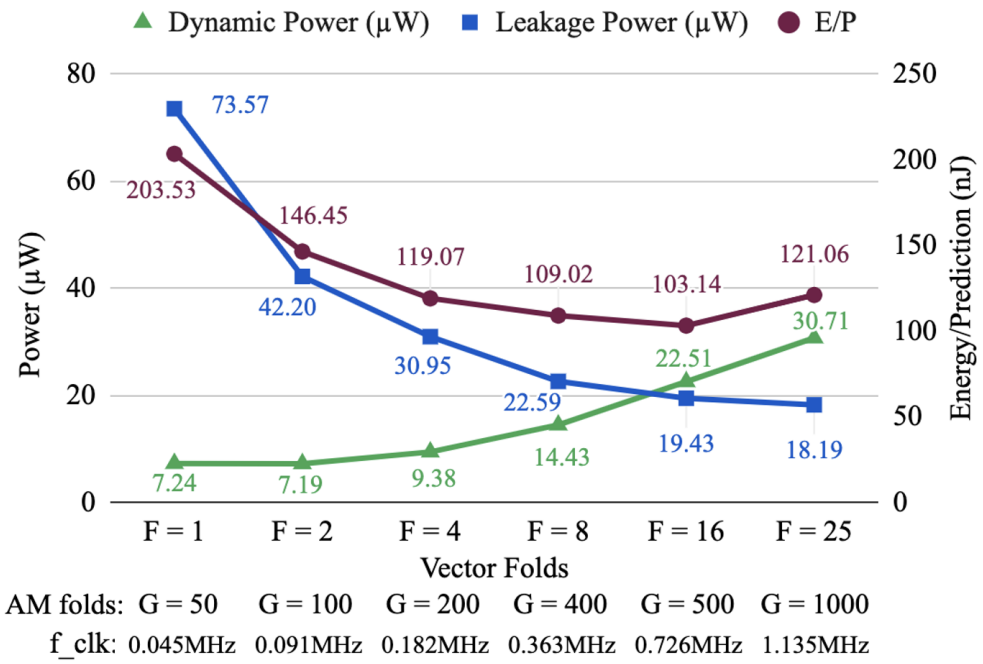


Figure 2.9: Dynamic and leakage power and energy/prediction to observe the optimal folding point when targeting 5ms classification latency.

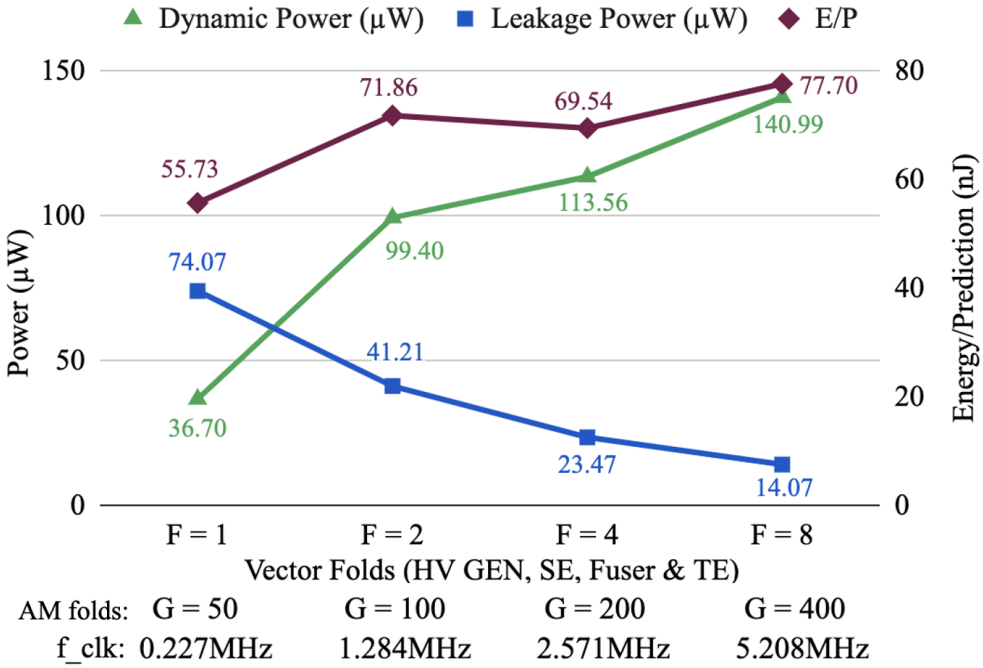


Figure 2.10: Dynamic and leakage power and energy/prediction to observe the impact of folding the temporal encoder targeting 1ms classification latency.

the other blocks.

Folding of the temporal encoder was explored to reduce its rapidly increasing power consumption as the fold factor increases (and in the fuser due to the unfolded output register). The results are shown in Figure 2.10. Temporal encoder folding necessitates re-generation of $n-1$ prior n-grams to form the n-length sequence, scaling latency by n. Hence, the processor needs to run n times faster to meet the target latency of 1ms which increases the dynamic power. Over the same time period, the resulting energy/prediction is much higher. Based on this, it can be concluded that folding of this block is too costly.

Hence, the temporal encoder is left unfolded. However, as a result, as the rest of the design is folded, the temporal encoder becomes the largest contributor to the area. By folding the temporal encoder, its size, and the output register in the fuser, drops along with the rest of the design, allowing the overall design to be nearly 45% smaller at the fold factor of 4. If there is a biomedical application that requires and prioritizes miniaturization, such as for implanted or distributed sensor nodes, folding the temporal encoder is an option to rapidly reduce the logic area.

The parameters in this architecture may find a new optimal given different design priorities or application specifications. For example, other applications may have other desired classification latencies as a result of varying sensor sampling rates, lower frequency biosignals, or low classification usage. Targeting highest energy efficiency for a different latency specification changes the optimal vector folding point. A 5ms classification latency was used to demonstrate the effect. The results are shown in Figure 2.9. Because of the decreased clock frequency, the overall dynamic power of the design has decreased while the leakage is very similar to the 1ms design. As a result, the intersection of the two has shifted to the right and the minimum energy/prediction occurs at a fold factor of 16. It follows that, if, instead, the classification latency was selected to be lower than 1ms, the dynamic power would increase to meet the target and hence the optimal fold factor would be smaller.

Finally, the design was pushed to it's maximum throughput to observe the performance at its extreme. A potential use-case is post-processing of data that has already been collected, providing local analysis of longer-term patterns/sequences of classified human response during the observed period, that the system can adapt to in the future. The results are shown in Figure 2.11. Without folding, the design meets timing at a clock frequency of 357MHz. The energy/prediction is the product of the total power and the classification latency, which is no longer constant since we are aiming for maximum throughput and not a target latency. The overall switching power is massively increased from the 1ms latency design. However, the drop in latency is so large that the energy efficiency is much higher. We hypothesized that folding the design would allow for further reduction of the critical path, and hence greater speed-up, due to reduced fanout. The design was evaluated at various fold factors, each running at the corresponding maximum throughput. By folding, we get reduced power, but the cycles per classification increases approximately by the folding factor. For the folding to be an improvement in energy/classification, the overall frequency (latency) needs to increase (decrease) by the same factor so as to not negate the decrease in power consumption. In this architecture, beyond a fold factor of 2, the critical path reductions that occur due to folding

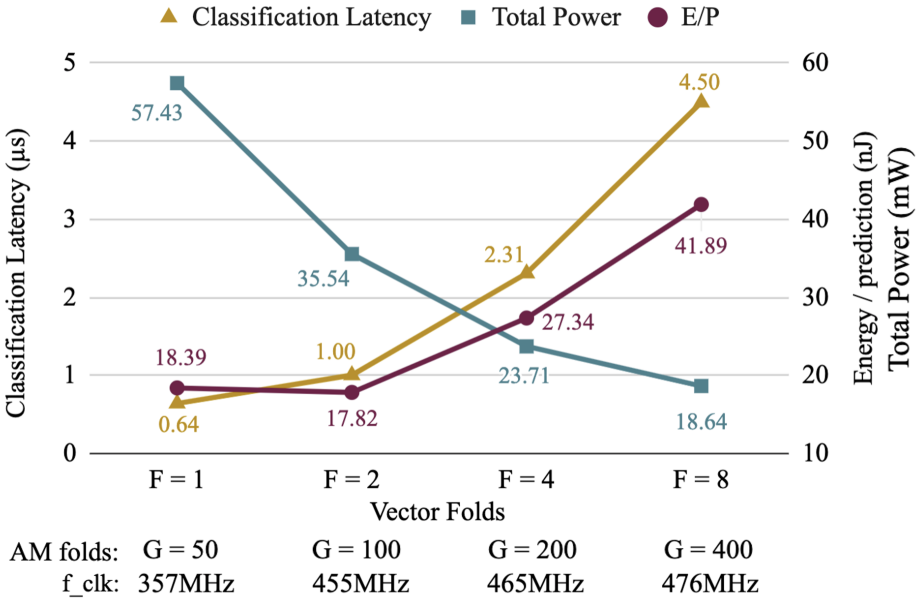


Figure 2.11: Energy/prediction when running at the maximum throughput for each fold factor.

and the corresponding decreases in latency are not large enough to benefit from the reduced power. However, at a fold factor of 2, this architecture runs at a frequency of 455 MHz and achieves an optimal energy efficiency of 17.8 nJ/prediction.

2.4 Alternate Implementations

In this work, architectures for alternate approaches to the same task, biosignal classification, especially for large numbers of channels, were also designed for comparison against HDC processor using CA *rule 90*.

The first is a RISC-V CPU implementation. Many IoT applications utilize microprocessors rather then application-specific ICs due to design time and fabrication costs. We explore optimizations to accelerate HDC on these platforms including utilizing commonly accessible vector accelerators. The heavily optimized energy/prediction is an expected several orders of magnitude larger than for the HDC ASIC processor due to the generality of the RISC-V processor, but several orders of magnitude improved over prior HDC CPU implementations, even those using accelerators with multiple cores [56]. The improved CPU performance can be used for tradeoff analysis to determine the platform of best fit by designers interested in taking advantage of HDC for their applications.

The second is an HDC processor ASIC with vector folding, but with ROMs instead of CA *rule 90*. This implementation specifically analyzes the contributions of vector generation and provides a parallel to prior HDC processor implementations.

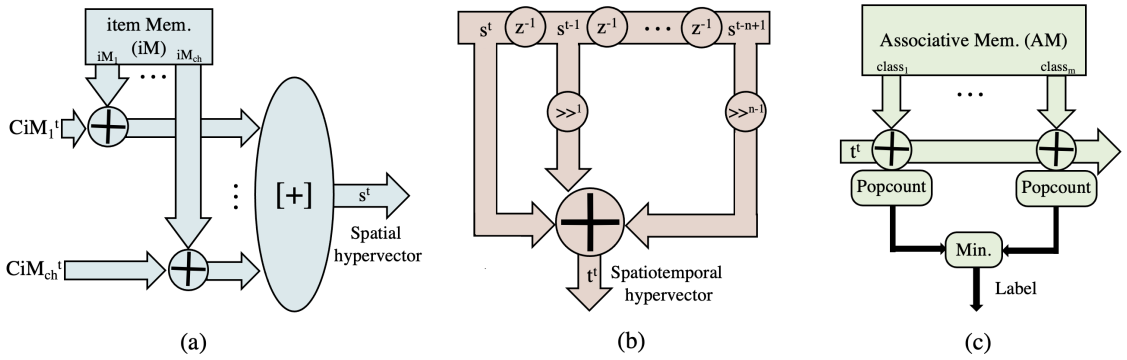


Figure 2.12: Key CPU operations for HDC encoding in the (a) spatial encoder (b) temporal encoder (c) associative memory.

The final approach is an optimized support vector machine architecture. Other models have achieved higher performance on the AMIGOS dataset. However, these algorithms are uncommon, especially as compared to an SVM, and therefore any comparisons or conclusions wouldn't generalize beyond this specific dataset. Hence, the SVM was implemented, allowing a hardware comparison between the proposed 2000-bit HDC processor and a common machine learning algorithm. The SVM implementation also enables a controlled analysis and comparison across various channel counts.

CPU Implementation

Modern machine learning workloads use computationally-intensive matrix multiplication, convolutions and accumulation that are addressed by integration of custom accelerators containing systolic arrays [27],[73]. As opposed to custom hardware accelerators, vector accelerators are commonly accessible, requiring little-to-no design time, and can even serve multiple applications with the same hardware instance [80]. Because HDC training and inference computation is done entirely with vectors of high dimension, vector accelerators have the potential to significantly improve HDC performance. Specifically, of the various HDC blocks, the spatial encoder is the most expensive in terms of cycles ($\sim$6$\times$ greater) and energy ($\sim$4$\times$ greater) due to the computationally-intensive bundling operation across input channels [56],[43]. As a result, classification of more than 128 channels requires >1.5 million cycles, even on an accelerator platform with 8 cores [56]. Hence, there is potential for significant optimization. Towards the acceleration of HDC on these platforms, in this section we explored HDC's energy consumption on CPUs, shown with the open-source Rocketchip RISC-V CPU [2] and Hwacha vector accelerator [37].

First, we propose a novel bit-serial word-parallel approach to accelerate the spatial encoder on any CPU. Then, we describe methods to optimize each block in the HDC classification algorithm for a vector accelerator, focusing on the key operations shown in Figure 2.12, and demonstrate the impact on execution time in terms of clock cycles and on power

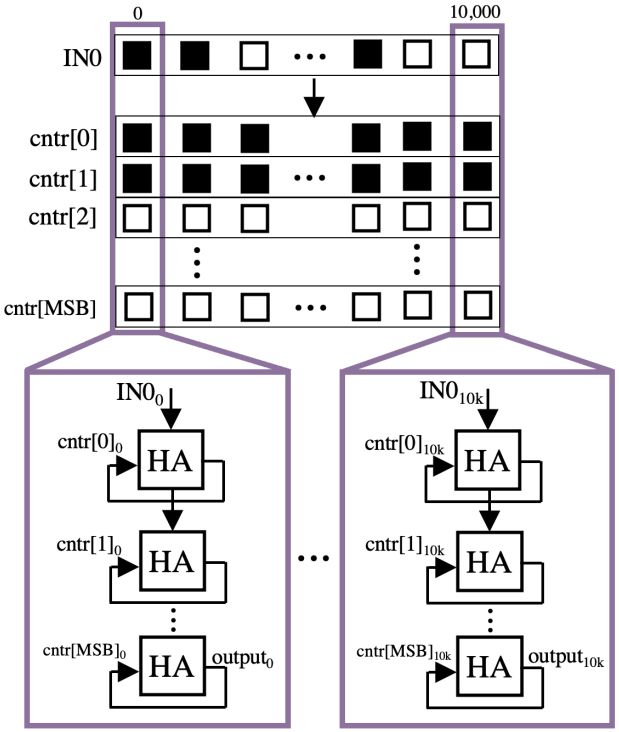


Figure 2.13: Bit-serial word-parallel technique to accelerate the key HDC bundling operation.

consumption. Finally, we explore additional hardware to improve execution time, specifically lanes in the vector accelerator, and show that there are diminishing returns in execution time resulting in an optimal configuration for energy efficiency, after which increased hardware complexity outweighs the reduced latency. We demonstrate the proposed techniques using the AMIGOS emotion recognition dataset used in prior sections due to the large number of input channels which allows for extensive exploration of expensive spatial encoding block.

Accelerating the Spatial Encoder on a CPU

The spatial encoder finds the bit-wise mode across all input channel HVs, visualized in Figure 2.13. The prior method for this involved looping through each word in the hypervector, extracting every bit into an unsigned integer and then executing a popcount [56]. While sufficient for an implementation with only 4 input channels, it scales very poorly due to the 3 nested loops and excess memory accesses [56]. The AMIGOS dataset used in this work contains 214 input channels, making this method infeasible for reasonable latencies. Hence, we moved to a transpose-popcount approach that involves breaking the HVs up into bit matrices (i.e. 64-element array of 64-bit unsigned integers), transposing them, and then performing a popcount on each row. While this removes 1 loop, it still requires a transpose and concurrent storage of unsigned integers equal in number to input channels.

To further reduce execution time we eliminated the transposition by implementing a

technique similar to the transition counting implemented in [36]. Visualized in Figure 2.13, a set of 157-element arrays of unsigned integers, hypervectors, were used to represent a counter. The counter was set to 7 bits for our dataset, large enough to count a majority across all channels. This resulted in 7 counter hypervectors; each hypervector represents a bit in the counter. The counters ripple-carry a bit in the incoming channel hypervector to corresponding bit in the counter MSB hypervector. The counters are initialized to a value equal to $2^7 - 1 - channels/2$. If a majority occurs for a certain bit, the corresponding bit in the counter MSB hypervector will flip. Thus, the output of the spatial encoder is simply the 7th counter hypervector. This process can occur in a single loop. Using this bit-serial, word-parallel method, the memory footprint in the spatial encoder stays constant as channels increase until another counter bit is needed at which point another hypervector would be added. This technique also improves the vectorizability of the bundling operation as all computation can occur across the entire vector in parallel.

Outside of the spatial encoder, the permutation operation in the temporal encoder requires carrying the overflow bit from every 64-bit unsigned integer in a ripple-shifting scheme. To speed up this operation, it was noted that the point of permutation is to remove correlation. A CPU-friendly optimization logically shifts each unsigned integer without carrying over, thereby injecting 1 zero every 64 bits. As this maintains sufficient orthogonality, the accuracy is not impacted, and several operations can be eliminated.

Utilization of a Vector Accelerator

Targeting Hwacha requires that vector assembly code be written inline with the original C code. The datapath was broken into blocks that could maximally utilize vector registers. For the spatial encoder, the bit-serial word-parallel technique was crucial to minimizing register footprint in the vector unit and loads/stores. In total, only 10 vector registers (1 per counter bit and 4 for intermediate storage), 4 address registers, and 5 shared registers were used. The ripple-carry through the counter involving ANDs and XORs from the input hypervector to MSB hypervector was manually unrolled for maximum register reuse. Thus, only 2 vector loads were required for each channel and 1 vector store at the end of spatial encoding.

For the temporal encoder, a set of N spatial hypervectors are logically right shifted and then bound to the output vector. The N parameter was three for the algorithm used; this loop was unrolled to eliminate redundant loading/storing of the output for each sample. The final code mapped to just three loads, one store, two shifts, and two XORs. In the associative memory, the class comparisons were unrolled with each class loaded into vector data registers. Hamming distance is calculated with an XOR which was done in the vector accelerator, and a popcount. The vector popcounts could not be implemented on Hwacha due to the inexistence of a vector accumulation instruction; instead, this was done on the Rocket core.

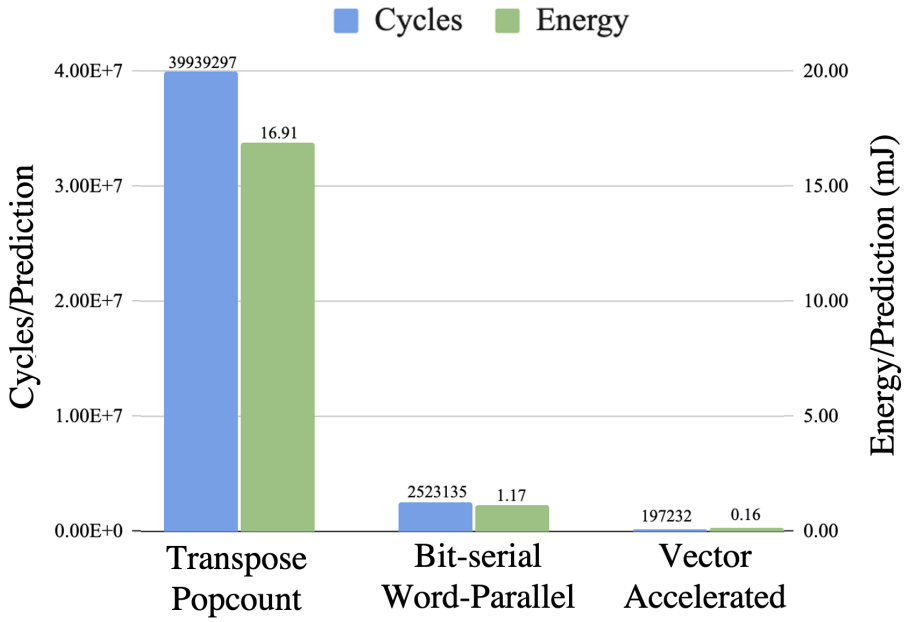


Figure 2.14: Execution time and energy efficiency with transpose-popcount bundling, bit-serial word-parallel bundling, and after implementation on the vector accelerator.

Optimization of the Vector Accelerator

Hwacha vector lanes contain two major components: the Vector Execution Unit (VXU) and the Vector Memory Unit (VMU) [38]. Vector instructions are steered to these decoupled lanes during the decode stage. Hwacha is parameterized to support multiple (powers-of-2) identical lanes. We simulated the classification task with 1 (default), 2, 4 and 8 vector lanes. Each additional lane enables increased parallel processing of elements in each vector which reduces execution time but increases average power consumption.

Results

The execution times in clock cycles and energy efficiencies of each acceleration effort are shown in Figure 2.14. The bit-serial word-parallel approach to spatial encoding improves the total number of cycles per prediction by 15.8× and the corresponding energy-efficiency by 14.5×. The transpose-popcount implementation had an average power consumption of 42.3mW. As the bit-serial word-parallel approach does not add any hardware complexity, there is minimal increase in average power consumption (3.9mW) and thus the total energy/prediction decreases by a similar factor as the execution time. The impact is shown for each HDC classification block in Figure 2.15. Using the transpose-popcount to bundle, the spatial encoder, for the AMIGOS dataset with 214 input channels, takes 2860× more cycles than the other blocks combined. By switching to the bit-serial word-parallel approach, the

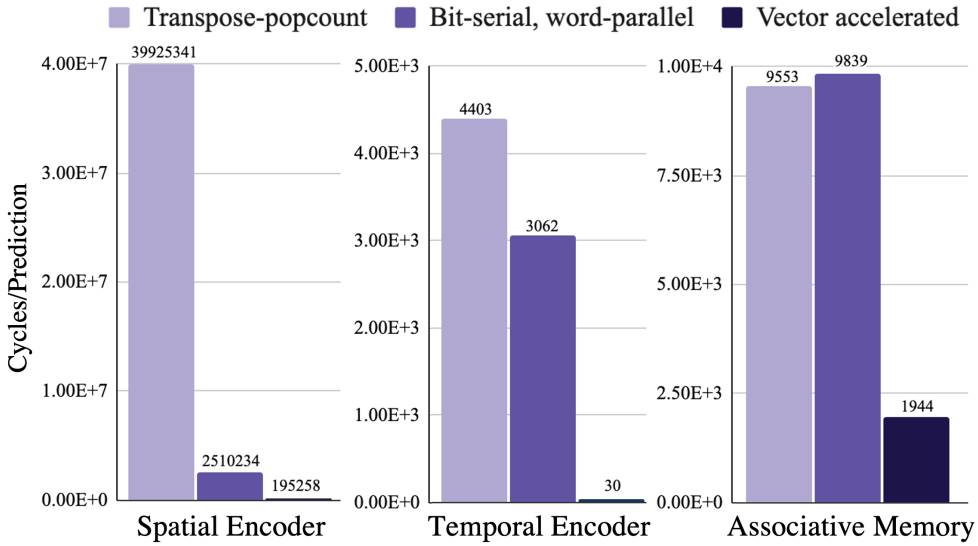


Figure 2.15: Impact of the optimizations on execution time in clock cycles for the spatial encoder, temporal encoder and associative memory.

spatial encoder is accelerated by a factor of 15.9×, the main factor in the net reduction in execution time. An 1.4× acceleration is also seen in the temporal encoder due to the elimination of overflow between array elements during circular shifts.

Implementing the algorithm on the Hwacha vector accelerator (with the bit-serial word-parallel technique) achieved an additional 12.8× reduction in algorithm execution time due to a speedup of 12.9× in the spatial encoder, 102.1× in the temporal encoder and 5.1× in the associative memory. With the vector accelerator and the bit-serial word-parallel approach, the spatial encoder was totally accelerated by 204.4×, now only 98× slower than all other blocks combined while processing >107× more vectors. The average power consumption increased by 33.2mW to 79.4mW due to the added hardware complexity, after integrating Hwacha onto the Rocket Chip. However, due to the reduced execution time, the energy efficiency still improved by a factor of 7.4.

Adding hardware acceleration through vector lanes continues to improve the execution time. However, as shown in Figure 2.16, the returns on this are diminishing. Adding one vector lane results in a 1.6× speedup, 2 to 4 lanes leads to a 1.4× speedup and 4 to 8 lanes only results in a 1.15× speedup. Simultaneously, the average power consumption grows more than linearly; one more lane causes a 1.3× increase, 2 to 4 lanes leads to a 1.4× increase and 4 to 8 lanes results in a 1.6× increase. While the optimal number of lanes may vary with input channels and hypervector length, this behavior will be consistent. For this dataset, the power increase for more than 4 lanes outweighs the reduced execution time; the net energy/prediction with 8 lanes is larger than for even 1 lane. As a result, the optimal number of lanes is 4 which achieves a 22.4% improvement in energy efficiency.

As compared to HDC implementations on prior accelerators with multiple cores [56],

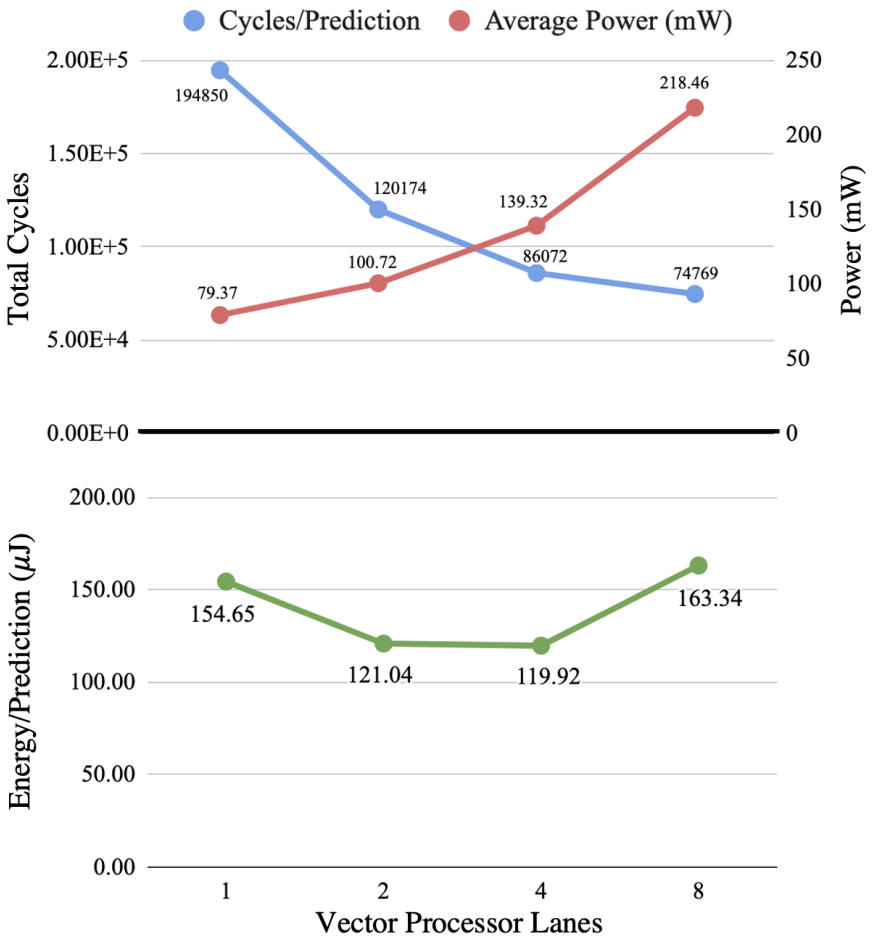


Figure 2.16: Change in execution time per prediction, average power and overall energy efficiency as the number of lanes in the vector accelerator increase.

the bit-serial word-parallel bundling technique on the vector accelerator offers several orders of magnitude improvement in clock cycles. With 4 vector lanes and a clock frequency of 100MHz, the proposed system achieves a latency of ∼0.86ms (>10x prior work), significant especially given the 214 channels being processed.

Overall, the bit-serial word parallel approach is a promising method to significantly accelerate the bundling operation that is key to HDC algorithms. The full classification algorithm can receive orders of magnitude of improvement in cycles and energy/prediction through integration of a vector accelerator to speed up HDC operations. Taking advantage of the available parameters of the vector accelerator such as vector lanes can further improve the overall energy per prediction until the relative optimum after which the diminishing improvements in cycles/prediction make the added hardware acceleration too expensive. In this section, we demonstrated the massive advantage provided to hyperdimensional comput-

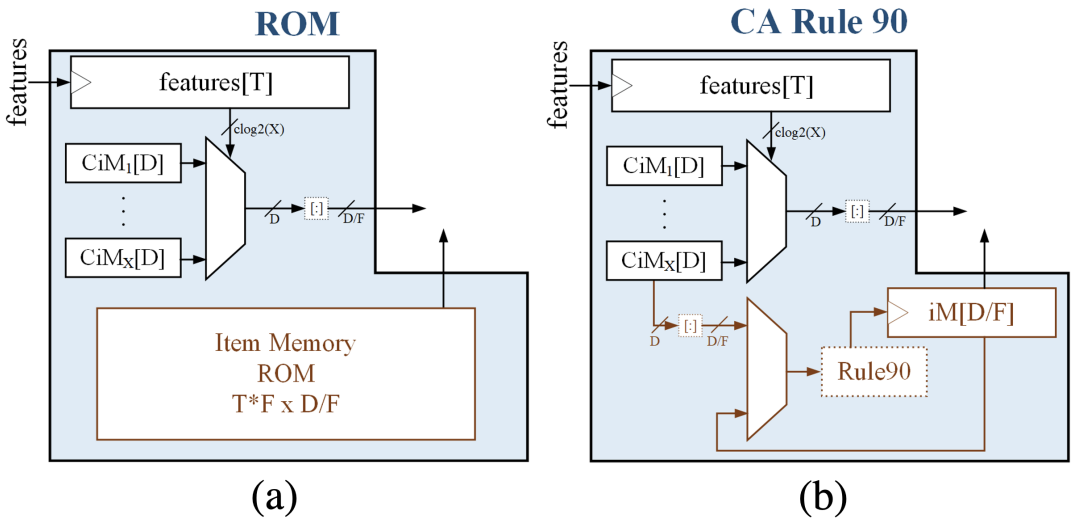


Figure 2.17: HV generation block with (a) ROMs vs. (b) CA *rule 90*.

ing operations by vector accelerators which, unlike custom accelerators, are accessible for integration into hardware instances without additional design effort and can provide utility for various other workloads. The combination of a vector accelerator with HDC-based classification provides a promising direction for highly efficient machine learning on the edge. While the optimized energy/prediction is an expected several orders of magnitude over the HDC processor with CA *rule 90* (119μJ vs. 39.1nJ), the CPU platform offers benefits such as reduced design time and functionality beyond HDC classification. For some applications, a CPU implementation may be preferable; the techniques introduced in this section provide a methodology to improve HDC performance on those platforms.

Future work for HDC CPU implementations can address strategies for data re-use. For example, if the required output rate allows for it, batch processing of multiple accumulated samples would allow for re-use of the same item memory vector multiple times before moving onto the next sample. This would necessitate multiple counters for multiple samples in parallel, but would significantly reduce memory transactions for all of the vectors involved.

HDC Processor using ROMS

The HDC ASIC processor with the ROM architecture is minimally altered from the baseline CA *rule 90* design; only the CA *rule 90* unit was replaced with an item memory ROM containing $214 \times F$ rows and D/F columns, each row containing one fold of a pre-computed item memory vector, as shown in Figure 2.17. Figure 2.18 shows a comparison of area utilization between storing hypervectors in ROM, as opposed to generating HVs through CA *rule 90*, for both the iM only and for the entire design. The area scaling in iM generation is minimal with CA *rule 90*, with an increase of only 3.11mm^2 from 3 channels to all 214 channels, predominantly attributed to an increase in counter overhead. The ROM iM on the

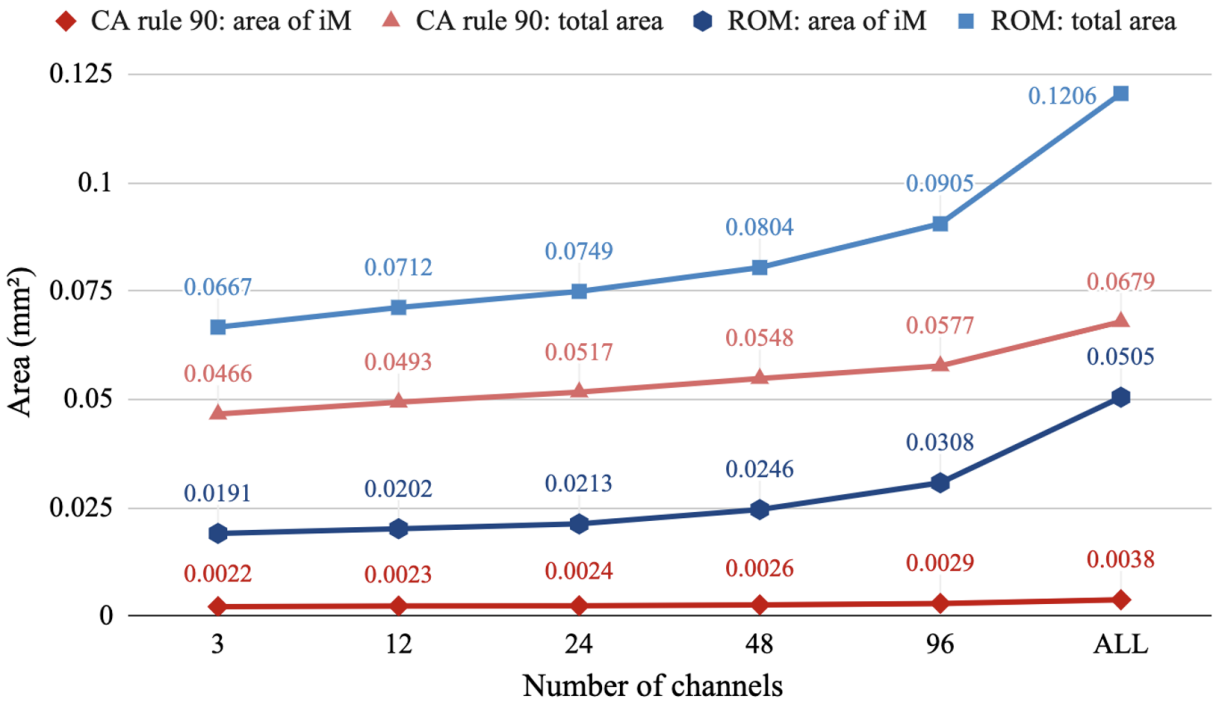


Figure 2.18: iM and total area as channels increase with CA *rule 90* HV generator vs. ROMs evaluated with $F = 4$ and 1ms classification latency.

other hand, increases by 30.84mm^2 as every new channel requires an additional 2000-bit HV to be stored. This growth in memory induces a significant 68.07mm^2 increase in total area for the ROM implementation, while in the CA *rule 90* design, only a 28.75mm^2 increase is observed, primarily originating from increases in the spatial encoder accumulator register bit-width.

SVM Processor

The SVM architecture was designed based on the algorithm in prior work for AMIGOS [76]. The goal of the SVM is to find a hyperplane in N-dimensional space that distinctively separates data points into their respective classes. For this architecture, two separate hyperplanes for valence and arousal classification are needed. The classification decision can be performed using the equation below.

$$\sum_{i=1}^{m} \alpha_i y^{(i)} K(x^{(i)}, x) + b, \tag{2.3}$$

α_i represents the support vector weights, y^i are the corresponding labels of the support vector, K is the kernel function, $x^{(i)}$ are the support vectors, x is the new data point, and b is the intercept. Given 214 input channels, a total of 120 support vectors for the valence

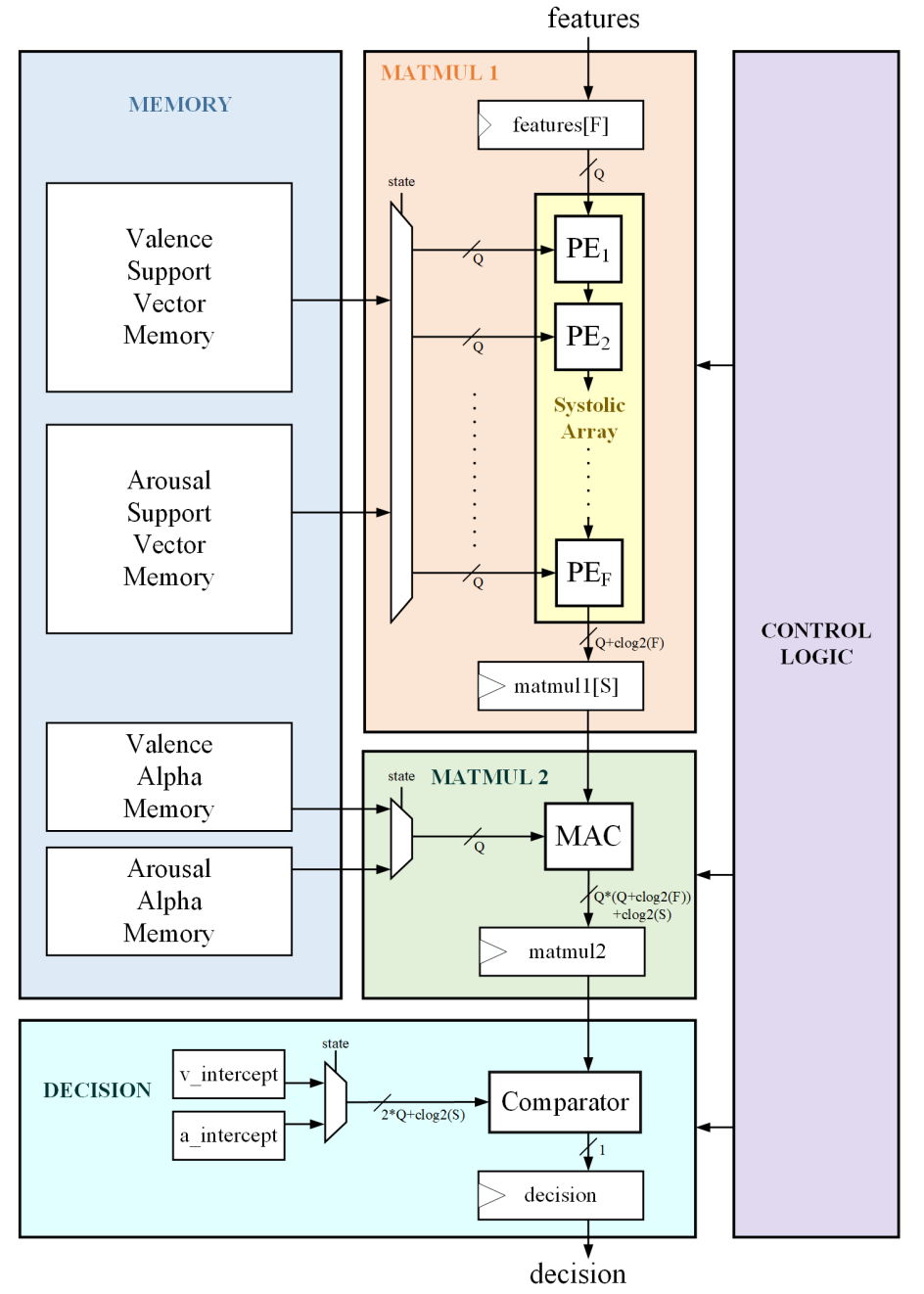


Figure 2.19: SVM architecture with systolic array

classification and 155 support vectors for the arousal classification are required. The number of support vectors remains relatively constant regardless of channel count. The length of each support vector scales with the number of channels, so the memory requirement for both valence and arousal support memories scales with the number of channels as well. In terms of algorithm performance, the SVM achieved accuracies of 68% for valence and 66.3% for arousal while HDC achieved 87.1% and 80.5% [76], [46].

Figure 2.19 depicts the overall system diagram of the SVM architecture. All input data is quantized to a minimum data-width of 9-bits to limit average accuracy loss to <5%. The classification process is divided into 3 steps.

- *MATMUL 1*: The first step is a matrix multiply between the support vector matrix and a set of 214 incoming features. To optimize cycle count, a systolic array with weight-stationary dataflow is implemented, consisting of 214 total processing elements (PEs). An entire set of features are first loaded into the systolic array, after which support vectors can then flow into each PE and propagate downwards. Each PE is also double buffered, allowing a new set of features to be sent in while the current classification is still running. Within each PE in the systolic array, input support values are multiplied with the stored feature, quantized back to 9-bits with minimal accuracy degradation, then accumulated with the partial sum from the PE above. The result of this matrix multiplication is an $S \times 1$ vector, where S is the number of support vectors.

- *MATMUL 2*: The second matrix multiplication is a single MAC, performing the multiplication between the $1 \times S$ α vector, and the $S \times 1$ result from *MATMUL 1*.

- *Decision*: The result of *MATMUL 2* is compared with the intercept b to generate the final classification. The classification is 1 if the result of *MATMUL 2* is greater than $-b$, and 0 otherwise.

To store the large set of support and alpha vectors, this SVM architecture contains a memory block, implemented as a set of ROMs. The data storage is optimized for minimal retrieval time and minimal control logic overhead. The entire set of required supports per cycle at the west face of the systolic array are stored as a single row in memory, resulting in single-cycle data retrieval from memory. The intercept value for the two valence and arousal classifications are stored as constant tie-high/tie-low cells. The overall power contributions from the SVM ROMs were scaled based on the ROM utilization.

Figure 2.20 compares the SVM with the 2000-bit HDC architectures, a dimension that is 10x larger than used in comparisons in prior work [56]. Overall, in terms of energy efficiency, HDC CA *rule 90* scales extremely well with the number of channels. For applications with more than a dozen channels, HDC outperforms the SVM. For the HDC ROM design, this is only true for channel counts greater than 96 due to the large memory cost. SVM does perform better at low channel counts because the supports and vectors scale directly with the number of channels. HDC, however, uses 2000-bit hypervectors regardless of the number of channels which makes the baseline single-channel architecture more expensive. This is,

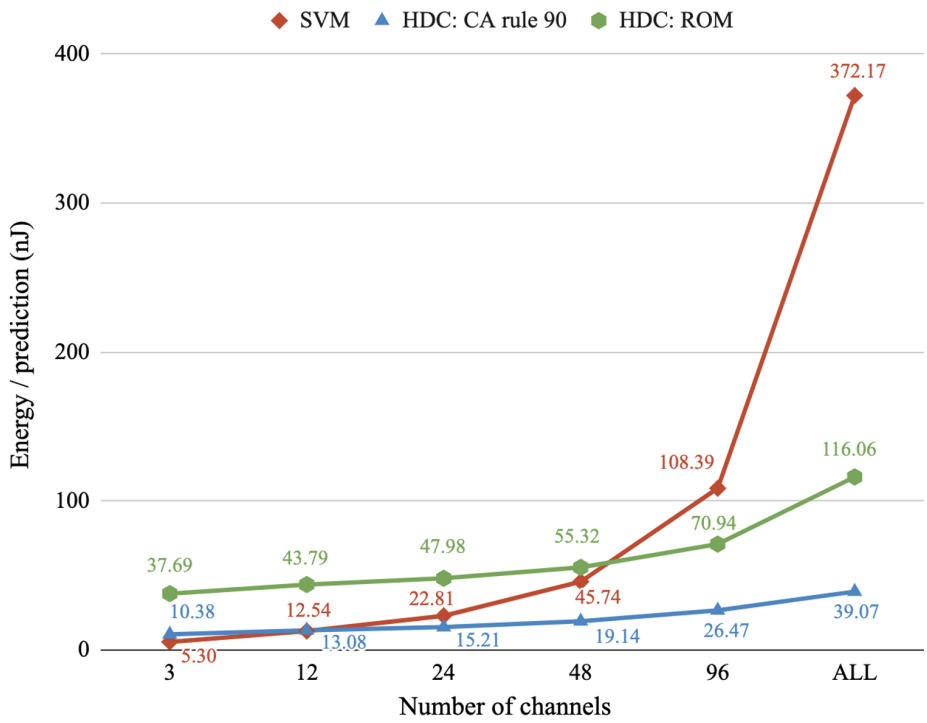


Figure 2.20: Energy/prediction comparison as the number of channels scales between the SVM, HDC with CA *rule 90* and HDC with ROM.

however, helpful at higher channel counts where the hypervector size is still constant instead of increasing like the SVM. A comparison is shown in Table 2.2. By fully exploiting the paradigm's simplicity and minimal computation, without specialized memory blocks and only standard cells, HDC achieves 9.5x better energy efficiency than SVM (372.2 nJ/prediction).

2.5 Conclusion

A comparison between the HDC processor proposed in this work and prior work (Table 2.3) presents a 4.9x improvement in energy efficiency over state-of-the-art [16] HDC processors for the implementation targeting physiological signal frequencies running at 909 kHz. At maximum throughput, 455 MHz, the architecture achieves an even larger improvement of 10.7x over the state-of-the-art HDC processor by minimizing leakage power. Since this processor was designed specifically targeting sensor arrays where the channel count is large, the energy efficiency per channel is also compared (64-channel implementations were used for fair comparison). An improvement of 9.5x and 33.5x over the state-of-the-art HDC processor can be seen for the 909 kHz and 455 MHz implementations, respectively.

Based on this, we conclude that CA *rule 90* is a highly viable choice for HDC processors, especially for applications with large channel counts, as it enables elimination of expensive

Specifications	SVM	HDC: CA rule 90
Supply voltage	0.9V	0.8V
Frequency	1.19MHz	909 kHz
Logic Area	0.334 mm^2	0.068 mm^2
Channels	214	214
Latency	1ms	1ms
Energy Efficiency	372.2 nJ/prediction	39.1 nJ/prediction
Energy Efficiency per channel *64 channel implementation for comparison	1.000 nJ/prediction	0.313 nJ/prediction

Table 2.2: Comparison between the HDC architecture proposed in this work using CA *rule 90* with vector folding and the SVM developed in this work both targeting physiological signal classification.

memory storage and overly complex generation. Vector folding alleviates the high fanout caused by hypervectors to improve energy efficiency. Though the most energy-efficient fold factor was 4 for a 1ms classification latency, this varies for different throughput requirements. Shorter latencies will find an optimum at lower fold factors, longer latencies at higher. With no accuracy loss down to widths of 20 bits, vector folding and CA *rule 90* can sucessfully be used concurrently. The techniques used in this work are not specific to the emotion recognition application; they generalize to other biosignal classification applications as well. Overall, reducing energy/prediction enables in-sensor classification with increased battery life which improves the usability of biosignal monitoring devices.

An SVM implementation was designed in this work for the same biosignal application to do a controlled analysis across a variety of channel counts for comparison. After applying the techniques proposed in this paper, HDC achieves better energy efficiency than the SVM for applications with more than a dozen channels. This is achieved even while maintaining high classification accuracy with a 2000-bit processor, making this the first, to our knowledge, viable HDC hardware comparison. This work's contributions establish HDC as a desirable paradigm for high-accuracy, extremely energy-efficient on-board biosignal classification, especially for large numbers of channels.

Table 2.3 also summarizes where this work fits within the biosignal classification ecosystem, including comparisons with previous work dealing with similar biosensor signals. In general, this work performs each classification with very low latency and energy/prediction per channel despite the large cumulative channel count, including when compared with similar work at comparable technology nodes and supply voltages. Overall, only Chua et al. demonstrates a lower energy/prediction, achieved with highly domain-specific features/algorithm and by aggressively scaling the supply voltage down to 0.5V [10]. Future works fabricated with this work's architecture would allow for power measurements at lower supplies, including tradeoff analysis between the accuracy and the lowest functional supply voltage as datapath bits begin to flip. Integrating this work with feature extraction hardware

	HDC Classifiers			Alternate Biosignal Classifiers					
Specifications	**This work**	**Eggiman et al. [15]**	**Datta et al. [14]**	**Huang et al. [8]**	**Chua et al. [9]**	**Shoaran et al. [12]**	**O'Leary et al. [10]**	**Liu et al. [13]**	**Zhao et al. [11]**
HV Dimension	2000 bits	2048 bits	2048 bits	-	-	-	-	-	-
Stored HVs (iM + AM)	3 + 4	2 + 5	1024 + 32	-	-	-	-	-	-
Folds	4 / 2	1	1	-	-	-	-	-	-
Application	Emotion recognition	Generic	Generic	Seizure detection	Seizure detection	Medical Devices	Brain state classification	Biomedical AI	Arrhythmia classifier
Dataset	AMIGOS	EMG [7]	EMG [7]	CHB-MIT	CHB-MIT	ieee.org	EPILEPSIAE	MIT-BIH / Bonn / NinaPro DB1	MIT-BIH
Sensors: type (channels)	214 from GSR (32), ECG (77), EEG (105)	EMG (64)	EMG (64)	EEG (16)	EEG (8)	iEEG (32)	iEEG (8)	ECG (2) / EEG (16) / EMG (10)	ECG (2)
Classifier	HDC	HDC	HDC	SVM	SOUL	Decision Trees	Decision Trees	Reconfigurable Neural Network	Artificial Neural Network
Technology	28nm	22nm	28nm	40nm	28nm	65nm	65nm	65nm	0.18μm
Fabricated vs. Simulation	Simulation (Post-layout)	Simulation (Post-layout)	Simulation (prior to layout)	Fabricated	Fabricated	Fabricated	Fabricated	Fabricated	Simulated
Supply voltage	0.8V	0.6V	0.8V	0.58V	0.5 V	0.8V	1.2V	0.75V	1.8V
Frequency	909 kHz / 455 MHz† †maximum throughput for this architecture	100 kHz	416 MHz	130 kHz	8 kHz	-	1 MHz	1 MHz / 500 kHz / 500 kHz	10 kHz
Area	0.068 / 0.093 mm^2 (logic)	-	3618 mm^2 (logic)	2.55 mm^2	0.1 mm^2	1 mm^2	1.95 mm^2	1.74 mm^2	0.92 mm^2
Latency	1 ms / 1μs	6.78ms	-	0.71s	1.6s – 2.6s	1.79s	-	-	630ms
Energy Efficiency (nJ/prediction)	39.1 / 17.8	191	610	170,900	1.5	41.2	36	2250 / 2060 / 5250	6770
Energy Efficiency per channel (nJ/prediction)	0.313* / 0.089* *64 channel implementation for comparison	2.98	9.53	10,681	0.1875	1.2875	4.5	1125 / 128.8 / 525	1605

Table 2.3: Comparison of this work at 909 kHz, targeting real-time physiological signal classification, and at 455 MHz, the maximum throughput, against state-of-the-art HDC processors and biomedical classifiers.

and sensor IO would also provide a more holistic view on the entire energy impact of this processor.

Future work can also explore the application of these techniques on different benchmarks with various throughputs requirements, particularly non-physiological time-series tasks such as keyword recognition or anomaly detection, where the feature resolution and hence CiM may need to be scaled to larger sizes. Comparing against other traditional machine learning algorithms, such as convolutional neural networks, in terms of both algorithmic performance and hardware efficiency for other benchmarks would further inform the advantages and limitations of the HDC paradigm. As more classes are added, the AM storage will run into same scaling issue as the iM. HDC vector compression techniques can be explored to address this. Without the iM memory storage, future work should explore algorithmic changes focusing on the spatial encoder which is now the largest block in the architecture due to the accumulator. This could include exploration of feature reduction or elimination for HDC. This has been used successfully for other machine learning algorithms and would reduce the overall number of channels and therefore spatial encoder cycles and register size.

Chapter 3

Design of Reactive Robotics with High Performance & Efficiency

3.1 Introduction

For robotic applications, deep learning faces challenges with learning from limited amounts of training data, integrating prior knowledge, generalizing to new environments, and producing interpretable models [74]. Using hypervector representation and operations, [58] proposes a fully-HDC algorithm for robot learning and recall of reactive behaviors (HDC RORB) to take advantage of HDC's interpretability, representational power, and robustness against noise to address these existing challenges. Previous works hypothesize the advantages of this approach, but do not measure its performance/limitations in a well-defined environment using standard metrics [58], [57].

A major motivation for recent interest in HDC is efficient hardware implementation [53], [44], [13], [16]. HDC has achieved 11.6x greater ASIC energy efficiency than an optimized support vector machine (SVM) [43] and 2x faster execution and lower power at iso-accuracy on an ARM Cortex M4 compared to an SVM [56]. It has high applicability for battery-operated systems, relevant to robotics. Though the HDC RORB algorithm is different from classification, the simple operations and minimal model size are maintained suggesting similar hardware efficiency.

In this work, we explore the ability of this technique to learn from human demonstrations and emulate those behaviors in new runs/environments for a large number of trials. The problem we aim to solve is prioritizing behaviors based on prior knowledge about the task through encoding of heterogeneous sensor information. Prior work on sensor fusion with HDC has shown its ability to represent heterogeneous sensor data in a unified form regardless of the type or complexity of the inputs [46], [7]. Each sensor channel is assigned a unique random hypervector (e.g. sensorID_n in Fig. 2). The discrete value of each of these channels is also assigned a random hypervector (e.g. sensorVal_n in Fig. 2) [18]. The two hypervectors are then bound together (through the XOR operator $\oplus$) to represent the relevant variable-value

correspondence. The sensor channel values can be obtained through feature extraction or other suitable discretized mappings. Thus, any type of information stream can be encoded into this representation, which lends HDC well to sensor fusion. After assigning hypervectors to sensor channels and values, they can be combined into rich representations through the elementary HDC operators; these representations enable weighting or prioritization of the different types of information. Previous work fused modalities with equal weight in order to create a sensor summary for classification [46]. In this work, we build on this premise but, exploring robotics applications, we propose uneven sensor weighting as a method to integrate prior knowledge about the task to influence behavioral priorities and demonstrate the ability to leverage the high-dimensional representation and operations to achieve this. This can be accomplished through modifying the impact a specific sensor has on the overall sensor hypervector representation, affecting the weight of that sensor on the recalled actuation. Various encodings are demonstrated on the 2-D robot simulated navigation task, shown in Fig. 3.1, to significantly impact the ability to reach the goal. To our knowledge, this is the first implementation of a fully-HDC algorithm for a robot navigation task.

In this work, we are primarily exploring the HDC RORB paradigm to determine if key HDC properties can be leveraged for robotics. State-of-the-art algorithms for similar 2-D navigation tasks include, but are not limited to, NNs and Reinforcement Learning (RL) approaches such as policy gradients [24], actor critic [50], and deep Q-learning networks (DQN) [51], [78], [1], [68]. To explore the role that HDC can play in tandem with state-of-the-art approaches, we develop a HDC-NN hybrid. Previous hybrid approaches use HDC representation with NNs [34], [30] and long short-term memory networks [47] for classification tasks, or for spatial representation [35]. In this work, we propose a hybrid approach to leverage the HDC weighted sensor encoding and noise robustness for robotics applications and observe the impact of the weighted sensor encoding on non-HDC models. We implement three versions: a standard NN, a NN with the HDC sensor encoding at the input, and a NN with the HDC sensor encoding at the input but trained to generate a hypervector as the output followed by the inherent HDC noise clean-up process. The weighted encoding significantly boosts the NN performance, demonstrating an avenue for future research on integrating HDC into other machine learning paradigms.

Prior work improves efficiency primarily through hypervector dimensionality reduction [53], [46], [13], [56]. We use this technique for in this work to observe the efficiency-performance trade-offs. HDC RORB, shown in Fig. 3.2, depends heavily on the program hypervector's ability to represent all sensor-actuator training pairs; this is heavily correlated to dimensionality of the hypervectors [60]. We hypothesize and experimentally demonstrate that the HDC-NN hybrid, which doesn't rely directly on hypervector dimensionality, performs well for a 50% smaller dimension than HDC RORB. Based on this, we compare the hardware efficiency for the two implementations based on first-order memory and operation count and show that HDC RORB is still more efficient, highlighting it's potential for resource-constrained systems.

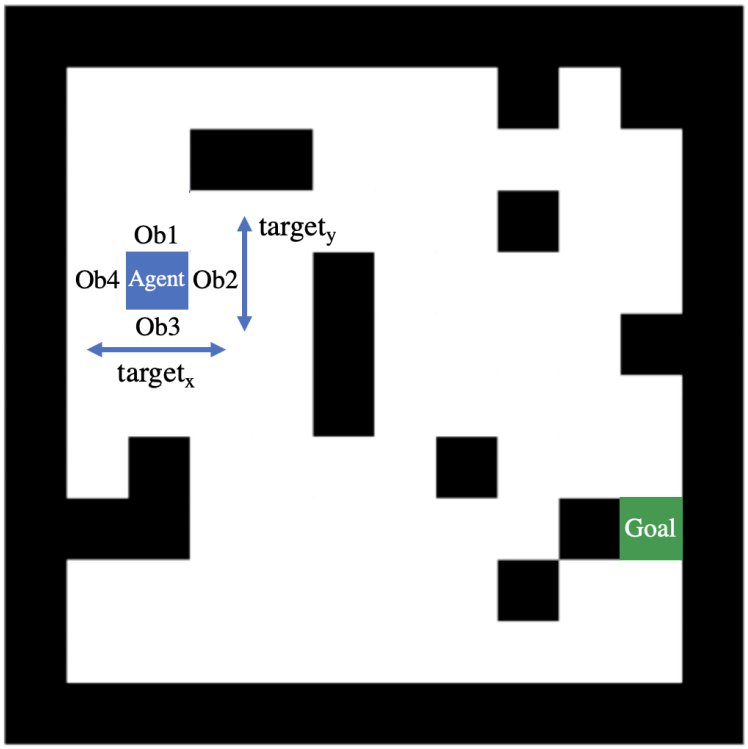


Figure 3.1: 2-D simulated navigation environment with obstacle (Ob1-Ob4) and target goal x and y direction sensors. The robot can move in the four directions (up, down, left, right).

3.2 Problem Formulation

We formalize the navigation and obstacle avoidance setting in a simulated 2-D 10×10 grid with 15 randomly placed obstacles as shown in Fig. 3.1. The blue square indicates the robot's current location while the green indicates the goal's location. The robot's task is to traverse to the goal location while avoiding any obstacles in its path. At each time-step, the robot moves in one of four directions (up, down, left, right) based on its environmental sensor data.

To mimic a real-world scenario, the environment is partially observable: the robot does not know its own state (grid location), but has access to sensor data for the presence of an obstacle in each of four directions around it, the x and y direction of the target goal, and its previous movement. Since HDC RORB is a supervised algorithm, training data was collected from human demonstration of moving the robot to the goal location through various randomly generated obstacle maps with random initial and goal locations.

We use this setup to explore the ability for HDC RORB to encode behavioral strategies for navigation of a 2-D maze. Using this testing environment, we aim to achieve compact encoding of heterogeneous information, prioritizing the different pieces of information in a way that is relevant or useful for the task. The same environment is used to study the hybrid

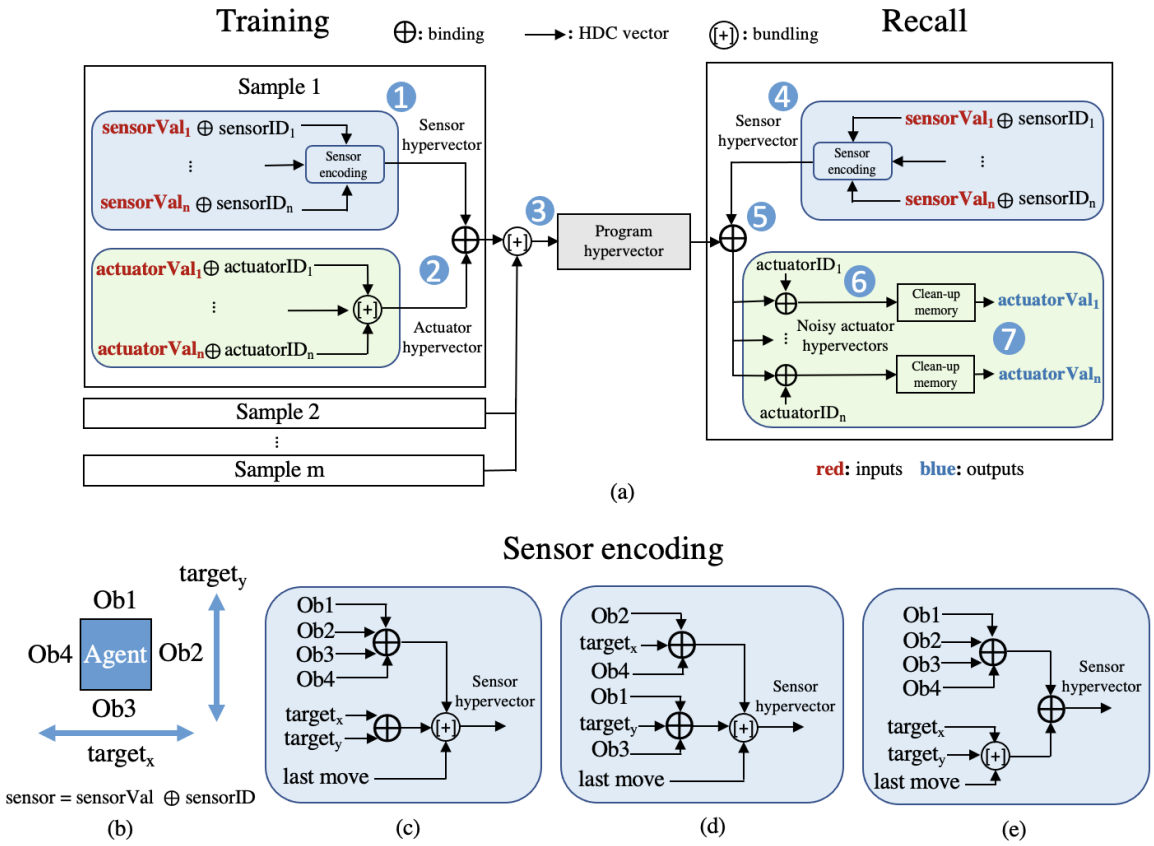


Figure 3.2: (a) Visualization of the sensor-actuator pairs generated during training of the program hypervector, and the query and clean-up done during recall in the HDC RORB algorithm. (b) A sensor hypervector is computed from the various sensors information through (c) modality-based encoding (d) directional encoding (e) constraints vs. goals.

HDC-NN schemes.

3.3 HDC RORB

All HDC algorithms rely on three fundamental vector operations: bundling (bitwise majority count), binding (bitwise XOR), and permutation (cyclic shift) [28]. The binding and permutation operations return a hypervector that is pseudo-orthogonal to its inputs. The bundling operation outputs a hypervector that is similar to each of its inputs. Functionality results from the following three concepts: (i) due to their very large dimension, randomly generated hypervectors are pseudo-orthogonal [28]; (ii) HDC is inherently high capacity from an information-theoretic perspective – as the hypervector dimension grows, the capacity of a single hypervector increases – more orthogonal hypervectors can be bundled together while

remaining individually identifiable [60]; (iii) the distance between two hypervectors can be measured using a similarity metric such as cosine similarity.

The HDC RORB algorithm uses these basic HDC operations and concepts to encode behaviors into a single program hypervector which can be later queried to recall behaviors, as shown in Fig. 3.2(a). For the 2-D navigation task, each of the 7 sensors and the actuator (in this case, the robot movement direction) is assigned a random ID hypervector, and random feature value hypervectors. These hypervectors are fixed and stored. During training, the sensor data is encoded into a single sensor hypervector using binding and bundling operations. For each training sample, the sensor hypervector is bound with the corresponding actuator hypervector, creating sensor-actuator pairs. The sensor-actuator hypervectors, if sufficiently different from hypervectors already available in the program hypervector until that point, are then bundled into the program hypervector.

During the testing phase, the same encoding process is performed on the input sensor data. The sensor hypervector is bound to the trained program hypervector, which effectively results in the 'unbinding' of a noisy actuator hypervector. This hypervector undergoes a clean-up process by comparing against the stored hypervector representations of possible robot actuations (up, down, left right). The actuation corresponding to the hypervector with the highest cosine similarity to the noisy actuator hypervector is provided as the output to the robot. This clean-up process pushes the output hypervectors into different parts of the hyperdimensional space and is the key to HDC's robustness to noise.

Heterogeneous Sensor Encodings

For the 2-D navigation task, the heterogeneous sensors (obstacle sensors, target sensors, and previous movement) are bound with their corresponding sensor ID hypervectors and then fused to create a single sensor hypervector. Recall that the binding operation creates an output pseudo-orthogonal to its inputs and the bundling operation creates an output similar to its inputs. Using these operations, the weight or relevance of specific sensors in the overall sensor hypervector can be manipulated. The fusion process was designed for three different strategies using this idea. The first is *modality-based encoding*, shown in Fig. 3.2(c), in which modality hypervectors are generated through binding of channels for each modality, and then bundled together. Each sensor has approximately an equal weight on the fused sensor hypervector.

The second method is *directional encoding*, shown in Fig. 3.2(d), which creates representations for the x and y directions through binding the horizontal obstacle data to the horizontal target data and the vertical obstacle data to the vertical target data. These directional representations are then bundled with the last move sensor. This method was inspired by how a human approaches this navigation task, by combining information on constraints/goals in the possible movement directions together.

The final method is *constraints vs. goals encoding*, shown in Fig. 3.2(e), which bundles the target direction sensors and last move sensors and binds these to the combined obstacle representation. This builds on the previous method with the difference that obstacles (con-

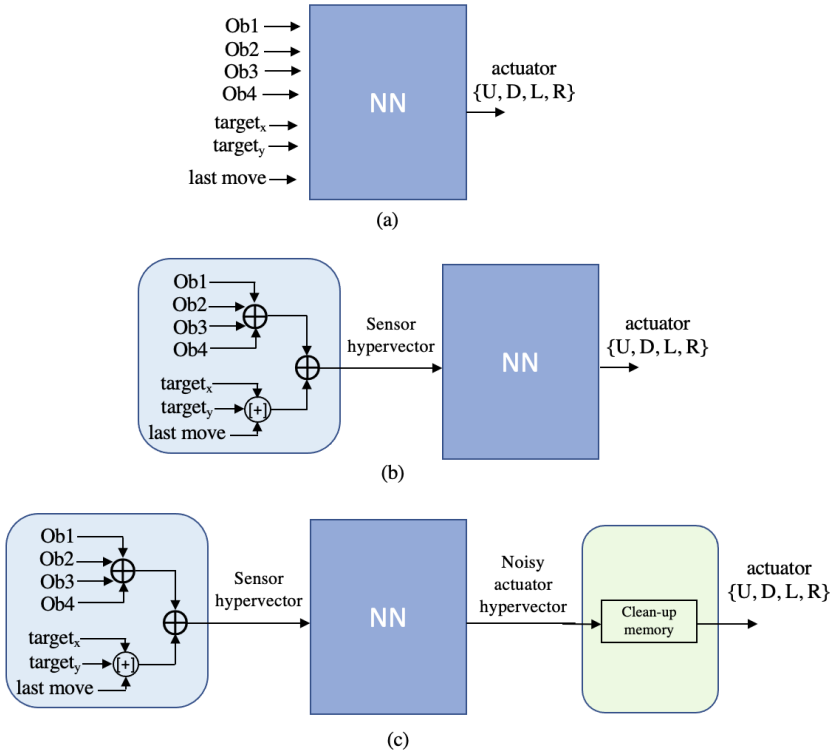


Figure 3.3: The implemented models are (a) NN: input sensor values, output actuation (b) HDC-NN1: input constraint vs. goals sensor hypervector where each bit is a feature for the NN, output actuation (c) HDC-NN2: input constraint vs. goals sensor hypervector, output actuator hypervector with the HDC clean-up process.

straints on movements towards task completion) and goals are combined directly instead of within each direction. This is equivalent to projecting the bundled direction and last move data into one of obstacle subspaces depending on the constraint data. This ensures that movement decisions are based first on obstacle avoidance; a different choice will be made for two different obstacle scenarios even if the bundled target direction and last move data is identical. Within a certain obstacle sensor combination, decisions are made based on the goal and last move.

The impact of the different encoding schemes on the success rate in the navigation environment are presented in Section 3.5.

3.4 Hybrid HDC and NN

To further explore the effect of the behavioral prioritization through weighted sensor fusion technique in non-HDC models, we integrated the HDC sensor hypervectors as inputs to a NN and compared against a standard NN with raw sensor data as inputs. The NN models

used in this work were developed in Python using the TensorFlow Library [40]. The first model, shown in Fig. 3.3(a), is a deep feed-forward NN with an input layer, 2 hidden layers, and an output layer. The input size is 7, the first and second hidden layers have 16 and 8 nodes, respectively, and both use ReLU activation functions. The output has 4 nodes, one for each of the 4 movement directions. The loss function used for learning was sparse categorical cross entropy. The stand-alone NN learns on raw sensor data as features and robot movements as labels. The HDC-NN1 hybrid, shown in Fig. 3.3(b), uses the same deep feed-forward NN but the size of its input layer is equal to the dimension of the sensor hypervector. As such, we are able to train the NN with the HDC-encoded dataset of human demonstrations; sensor hypervectors are the features, while the corresponding movements are the labels. The comparison results are shown in Section 3.5

To improve the performance of the hybrid model, we further designed the HDC-NN2 hybrid, shown in Fig. 3.3(c). This version also contains a deep feed-forward NN, however, the output is a hypervector. In this case, the input size is the hypervector dimension, but the first and second hidden layers have 50 and 20 nodes, respectively, and both use tanh activation functions in order to output a bipolar hypervector. To achieve this, the loss function is selected to be cosine similarity, the metric used for measuring distance between hypervectors. The neural network is trained on HDC-encoded sensor hypervectors as features with corresponding actuator hypervectors as labels. During inference, the NN output hypervector is returned to HDC representation by thresholding to bipolar values. Then, a prediction is made by computing the maximum cosine similarity against the 4 HDC actuator hypervectors. Note that this is the same actuator clean-up used by HDC RORB; it is inherent to the process of returning from the HDC representation to the selected robot actuation. The performance of this system is compared with the NN, HDC-NN1 and HDC RORB in Section 3.5

A softmax function is applied to the outputs of HDC-NN1 (probability), HDC-NN2 (cosine similarity) and HDC RORB (cosine similarity) to add a probabilistic element that prevents an endless loop of repeated movements. This tends to occur when there is conflicting sensor information, such as when the robot finds itself stuck inside a dead-end (in this situation, obstacle avoidance can conflict with goal reaching). The softmax function is not implemented for the modality-based or directional HDC RORB sensor encodings or the stand-alone NN for which the crash rates are too high for this to occur.

3.5 Experimental Results

Testing was done with 100 trials on a set of 100 randomly generated obstacle maps with different initial and goal locations (100,000 runs in total). The same set of maps was used to compare between algorithms. Successful trials are those where the robot reaches the goal within a reasonable number of steps without crashing into an obstacle. The reasonable number of steps was defined with a timeout set to 100 iterations. This limit was hit only if the robot entered a endless loop of repeated movements due to conflicting sensor information,

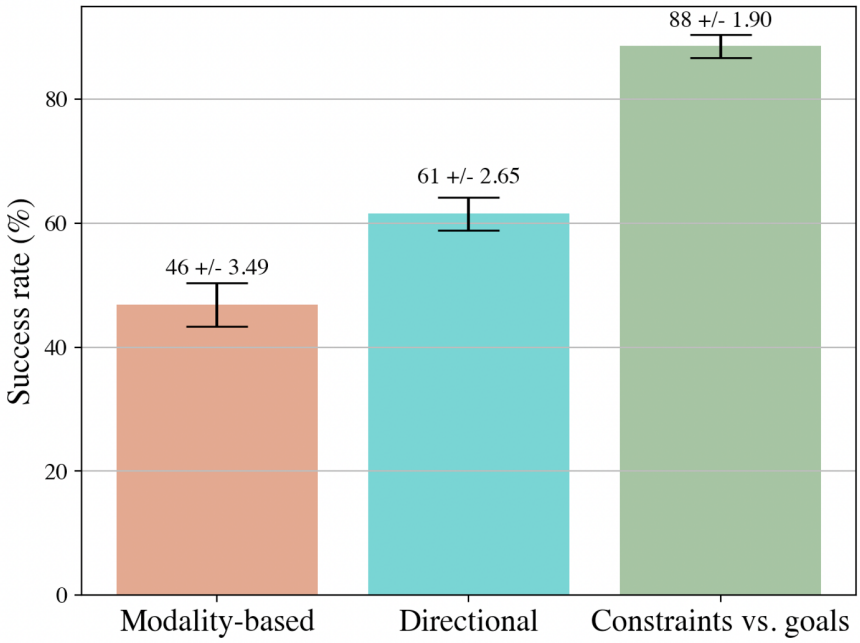


Figure 3.4: The impact of sensor encoding strategy on success rate for HDC RORB. The error bars indicate standard deviation.

neither crashing nor reaching the goal; this is considered a failed trial. The robot fails immediately if it crashes into an obstacle. Success rate is defined as the mean percentage of success in the set of 100 maps over all 100 trials. Standard deviation is computed for the percentage of success in the 100 maps over the 100 trials.

Behavioral Prioritization through Sensor Fusion

Success rate of the 3 different encoding schemes used with HDC RORB across the 100 trials in each of the 100 testing obstacle maps is shown in Fig. 3.4. The modality-based encoding performs the worst at 46.89%, the directional encoding does ∼15% better at 61.53%, and the constraints vs. goals encoding does the best at 88.57%. The results demonstrate the ability to integrate prior knowledge about the task into the encoding process, typically a challenge for deep learning [74]. Sensors that only go through binding operations into the final sensor hypervector have a strong impact on the output. Sensors that go through the bundling operation will have a weaker impact and lower association with specific output movements. Therefore, the binding operation allows for prioritizing the impact of changes in specific sensors, and through this, the ability to prioritize certain behaviors.

For this navigation task, hitting an obstacle causes a failed trial. Therefore, the first priority should be to avoid obstacles, even if temporarily moving away from the goal. The ability to make that choice requires balance between the decisions different sensors suggest.

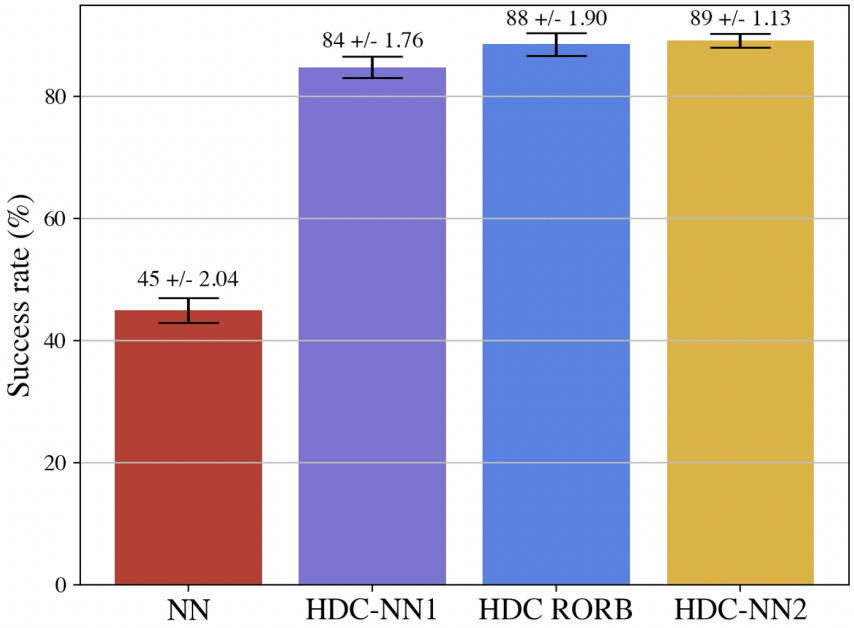


Figure 3.5: Success rates for the NN on its own, the HDC-NN1 with constraints vs. goals sensor encoding, and the HDC-NN2 with constraints vs. goals sensor encoding and actuator clean-up. Error bars indicate standard deviation.

As demonstrated by the results, HDC RORB can learn this behavioral priority by binding the obstacle sensors at both stages in the sensor encoding process while including the target goal and last move sensors through a bundling operation in the first stage, as shown in Fig. 3.2(e). The constraints vs. goal encoding method is the most suitable for this task. Using only 3 operations, the results showcase the simplicity and expressiveness of weighted sensor fusion in HDC representation.

Integration of HDC with NN

The success rates of the 3 instances of NN-based models are shown in Fig. 3.5. The NN alone has a 45% success rate which is similar to the HDC RORB with modality-based encoding. This shows, interestingly, that performance is equally low for the two algorithms when the sensor information is provided to the model without intelligent weighting based on prior knowledge about the task.

Because of its high performance with HDC RORB, constraint vs. goals sensor hypervector encoding was integrated into the HDC-NN hybrids. With each bit of the weighted sensor hypervector provided as a feature to the NN, HDC-NN1 shows ∼40% improvement in success rate to 84.81%. This is a significant boost and demonstrates the value of the expressive, weighted HDC sensor fusion technique, even for non-HDC learning schemes. When NNs

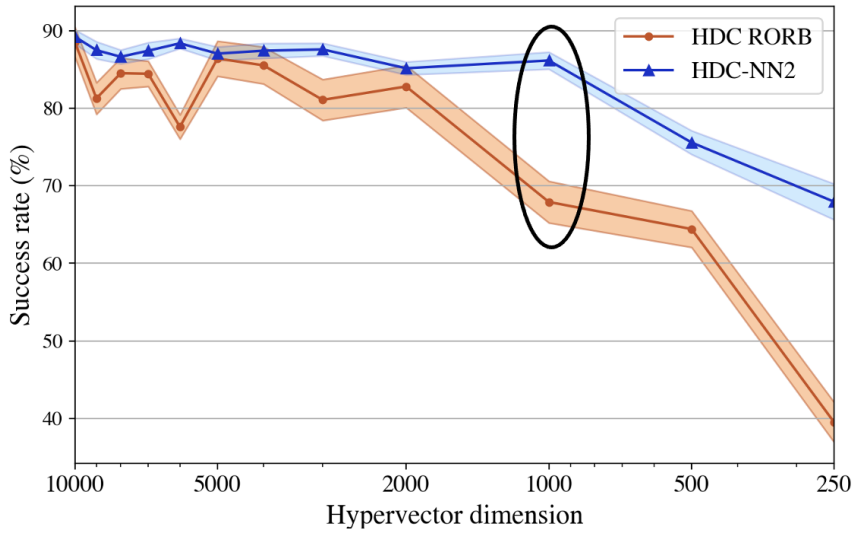


Figure 3.6: Impact of hypervector dimension on HDC RORB and HDC-NN2 success rate. At 1000, HDC RORB degrades by more than 20% while the HDC-NN hybrid remains within 3% of the peak. Shaded area indicates standard deviation.

are trained with the original sensor data, they are not capable of prioritizing application-relevant information because they lack knowledge encoding capabilities, they simply learn statistically-relevant patterns found in that data. A hybrid allows the NN to receive some of the knowledge encoding through representing training data as weighted fused sensor hypervectors.

However, even when using the constraints vs. goals sensor hypervector bits as the input features to the NN, it still has a ∼4% lower success rate than HDC RORB. The only remaining difference between the two is that the HDC-NN1 outputs an integer representing the selected robot actuation while HDC RORB outputs a hypervector that undergoes the clean-up process, pushing it closer to an existing actuator hypervector before translating down to an robot actuation.

To include this key step, as explained in Section 3.4, the HDC-NN2 hybrid uses actuator hypervectors as labels and the cosine similarity-based clean-up process on the output of the NN. The success rate improves by ∼4.5% when using this system, demonstrating that the clean-up process contributes to the overall performance. With this boost, the HDC-NN2 hybrid achieves 89.22% success rate, actually better (∼0.7%) than HDC RORB. The deep feed-forward networks used in this work underwent a hyperparameter search to select layer sizes, but the models were kept simple (input, output, 2 hidden layers) for the purposes of integration exploration. We hypothesize that additional time spent designing a NN for the specific task and for the hypervector input and output layers could result in success rates with a larger margin above the HDC compressed recall algorithm.

Considering hardware efficiency (a key benefit of the HDC paradigm), the most com-

mon optimization technique is reducing the hypervector dimension. This reduces the size of the entire datapath affecting operation count and memory storage approximately linearly. However, due to the correlation between hypervector dimension and ability to store information [60], this reduction has an impact on algorithm performance. For HDC RORB, a reduction in dimension affects how many training examples the program memory can learn and effectively unbind into recognizable actuator hypervectors during recall. This effect is confirmed in Fig. 3.6 where HDC RORB significantly degrades for dimensions below 2000. We hypothesized that the HDC-NN2 hybrid would not face as significant performance effects, because the training samples are used to train weights rather than be bundled into a compressed hypervector. This is demonstrated in Fig. 3.6. The hybrid is able to maintain performance to within ∼3% even with a 90% reduction in dimension. By 1000 bits, HDC RORB has already degraded by more than 20%; the HDC-NN hybrid allows for a 50% smaller hypervector dimension than HDC RORB.

For this task, HDC RORB requires storage of around 17 hypervectors including the binary program and ID/values hypervectors. The computation includes XORs, additions, comparisons and hamming distance operations, totalling to around 20 operations per hypervector bit. To first order, if D is the hypervector dimension, this results in $20\times D$ operations per recall and $17\times D$ bits in memory. For the HDC-NN2, all the weights in the neural network are stored instead of the program hypervector. There are $D\times 50$ feed forward weights between each input node and hidden layer node and $20\times D$ between each second hidden layer node and output node. Hence, the hybrid network storage scales with D by a factor of 70. The operation count of the NN includes the multiply and additions which in total, to first order, scale with D by a factor of 140. With 32-bit weights, this is approximately $2256\times D$ bits in memory and $159\times D$ operations, excluding the middle layers. Because layers of the NN are now HDC dimension, the hardware cost has significantly expanded. Even with 50% smaller hypervector dimension for the hybrid, the HDC RORB would still be more efficient.

3.6 Conclusion

In this work, we explored the use of hyperdimensional computing for robotics navigation. Using prior knowledge about a 2-D navigation and obstacle avoidance task, we proposed a sensor encoding scheme that uses only 3 operations to prioritize selected behaviors through sensor weighting. The effectiveness, simplicity and noise robustness of HDC encoding and representation for weighted sensor fusion tasks showed promising results for both HDC RORB and non-HDC paradigms. The hybrid model with a hypervector input and output is highly robust against dimension reduction. However, the simple HDC RORB algorithm has the highest hardware efficiency of the explored models, demonstrating its potential for resource-constrained robotics systems. The hybrid schemes provide avenues for future research in integration schemes that leverage the advantages of HDC alongside other machine learning algorithms to improve performance for robotics applications.

Future work includes observing performance of HDC RORB as more sensors and actuator

degrees of freedom are available. In this work, the success rate metric was used to compare ability to navigate while avoiding obstacles, but additional observation of optimality of the taken paths to the goal would further inform the model's performance, particularly as more variables are added. Transitioning the model from the simulation environment to a physical environment and measuring the ability to handle noise from non-ideal sensors would provide insight into the input noise robustness. A key property of HDC is the ability to perform well with limited training data, useful for robotics. Future work could explore this for HDC RORB. Finally, in this work we show that HDC facilitates hybrid models; future work should consider these results as a baseline. The hybrid techniques in this work provide a basis on which integration of HDC with state-of-the-art robotics learning algorithms such as DQN-based reinforcement learning is possible. Exploring that for more complex robotics applications is left as future work.

Chapter 4

Shared Control for Neural Prosthetic Devices

4.1 Introduction

Current myoelectric upper limb prosthetic devices aim to bridge the gap between users and their prosthetic with a functional residual limb that performs common grips for activities of daily living including pinching, pushing, or grasping objects [4]. However, assistive functionality on commercial hands is limited to a set of programmable grips [52] decided based on the user's daily activities [77]. The terminal device of myoelectric prostheses is capable of rotating through a set of programmable grips - the myoelectric control generally involves two sensors on a flexor and extensor muscle which are used to trigger this. While noninvasive, the myoelectric control relies on the consistent accuracy of sensor placement on the forearm when donning the prosthetic device, involves a long training phase and faces signal degredation even during a single wear session [11]. As a result, some companies take extra measures such as the Grip Cycle Button in the TASKA hand which provides the necessary means to change a grip or return to a neutral position while bypassing the use of EMG signals.

Due to the inconvenience and unreliability of switching between grips, users stick to one grip 70% of the time [4], resulting in low voluntary usage of even the existing limited functionalities. Additionally, while performing a grasping task, users focus around 90% of their visual attention on either their prosthetic or the area near the object [17]. This, and factors such as dexterity limitations and unintuitive design ultimately lead to up to a 75% abandonment rate of prosthetics [22]. EMG-based control also puts excessive burden on the user to over-emphasize grasping muscle patterns, causing early fatigue [85]. Overall, the lack of intuitive assistive functionality makes it difficult for users to be motivated to continue dealing with the practical issues and mental load involved [17].

Existing approaches to robotic control that eliminate the mental/physical burden of EMG-based control include optimal control algorithms such as reinforcement learning (RL). These are fully automated through a policy that minimizes a cost function [69]. However,

the policies are limited to known scenarios/goals [81] and are slow and data hungry [67] - particularly an issue for myoelectric control where limited training data exists for a specific user/wear session and latency impedes proper functionality. Additionally, the system selects an optimal trajectory so the user can not naturally execute on their intents, make decisions or anticipate prosthetic behavior. Hence this alone does not address prosthetic needs.

"Shared control" is an approach that intelligently arbitrates between human control and prosthetic autonomy. This strategy executes on the user's intent while alleviating the burden of fine control. Existing shared control schemes include a binary system that switches between a multi-layer perceptron for myoelectric control to autonomous object contact maximization once contact is made [85]. However, in this case, the human's intent is lost once the automated control takes over. Also, the system does not generalize to daily tasks that do not need, or may even be hindered by maximum contact. A different shared control scheme uses a weighted sum of the intent-specific optimal trajectory learned through inverse RL and the direct human-controlled trajectory [86]. However, the involvement of RL limits it to specific trajectories - a variation in object location would require re-training under a new intent.

In this work, we design & evaluate a shared control system that maintains user autonomy at all times, but displaces the burden of fine control. The system is user-adaptive, generalizes to new tasks without large amounts of training data, and aims for computational simplicity for in-device implementation to lower latency and avoid power-hungry data streaming. It has the potential to quickly update its model, avoiding performance degradation due to time or sensor placement inconsistency. Towards these goals, the system takes advantage of the benefits of Hyperdimensional Computing discussed in prior sections such as energy efficiency and small memory footprint [43] as shown in in Chapter 2, high accuracy on limited training data [46] shown in Chapter 2, and quick model updates for new classes/contexts - especially useful for user/session-specific EMG systems [54]. Embedded HDC enables low latency due to its simplicity, enabling in-device feedback for rapid, natural prosthetic response [55]. HDC can also be used to prioritize reactive robotic behaviors relevant to specific user intents through intelligent sensor encoding [42] as shown in Chapter 3. HDC is a very versatile paradigm that, in addition to the classification mentioned before, can encode probabilistic queries [19], address sensorimotor mappings [48], enable model superposition [8] for reduced memory footprint [83] and achieve efficient symbolic reasoning [49].

These various schemes can be used in combination to leverage the benefits of HDC to address the versatile needs of prosthetic shared control. This work utilizes the appropriate algorithms to achieve decoding, interpretation, prediction and reaction in a multi-layer fashion in order to achieve prosthetic shared control.

4.2 Data Collection

We collected a dataset of EMG, accelerometer and force sensor data while subjects completed a set of 6 ADLs. These include "folding laundry", "writing with pen", "opening a jar",

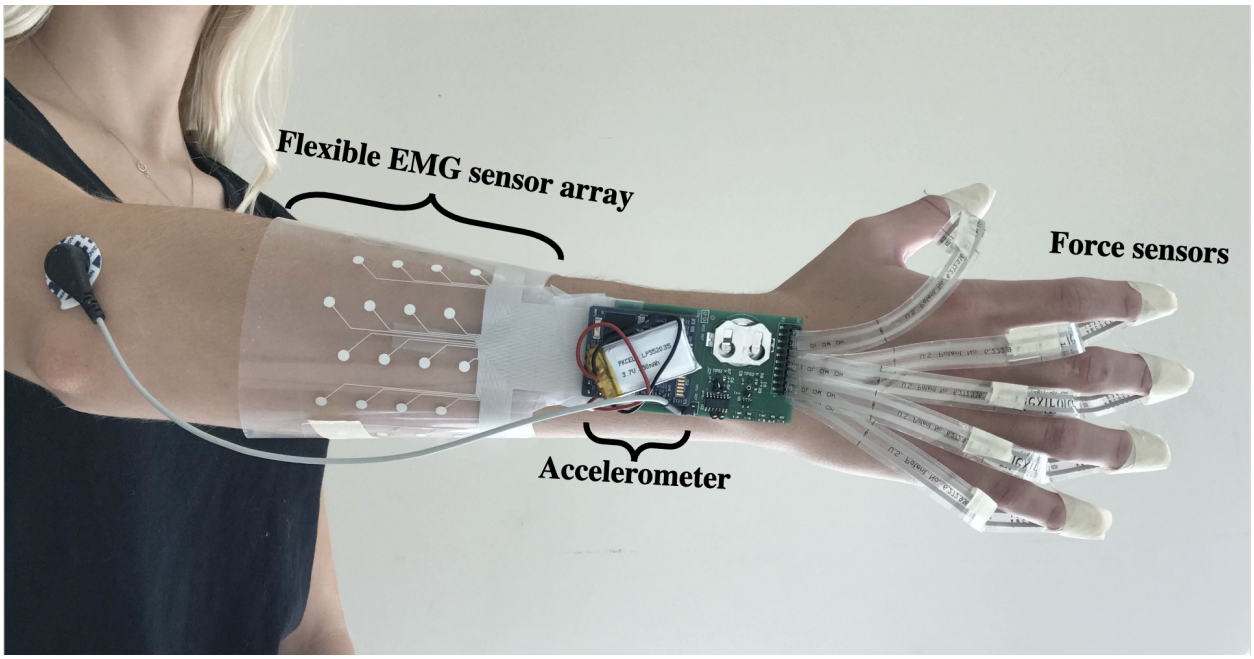


Figure 4.1: The data collection setup integrating flexible force sensors into the EMG and accelerometer acquisition system.

"screwing lightbult", "combing hair", and "tying shoelaces", selected to cover a variety of hand motions and positions [45]. The EMG sensor array was fabricated by screen-printing connective traces and circular electrodes (diameter r, 4.3mm) using conductive silver ink (NovaCentrix FG57b) in a uniform 4×16 array on a flexible PET substrate [55]. Each row was spaced 14.3mm apart, and each column was spaced 17.8mm apart. A dielectric encapsulation layer (NovaCentrix DE-SP1) contains via holes to expose the electrode pads but insulate the connective traces from the skin. Overall dimensions (29.3cm×8.2cm) allow the array to wrap around an above-average-sized forearm, enabling observation of extrinsic flexor and extensor muscle activity [55].

The flexible EMG sensor array interfaces with the WAND device - an 8-layer PCB that accommodates an accelerometer and components for processing/telemetry [84]. This is done using a flat flexible connector (FFC) on a two-layer adapter board re-designed in this work to also interface with 5 force sensors. Tekscan FlexiForce A201 111N resistive force sensors were selected for their precision and replicability. These were placed in a voltage divider with the reference voltage and unity-gain buffer to output a signal that changes with the force applied. The additions to the PCB include force sensor connectors, op-amps for unity gain buffers and resistors for the voltage dividers for each sensor, and a 3V coin cell battery to power the voltage supplies and reference voltages. To improve the replicability and stability of the signal for weights with larger radii, a bumper was added to centralize the force onto the sensor area. The force sensors had a maximum sensor length of 190.5 mm that allowed

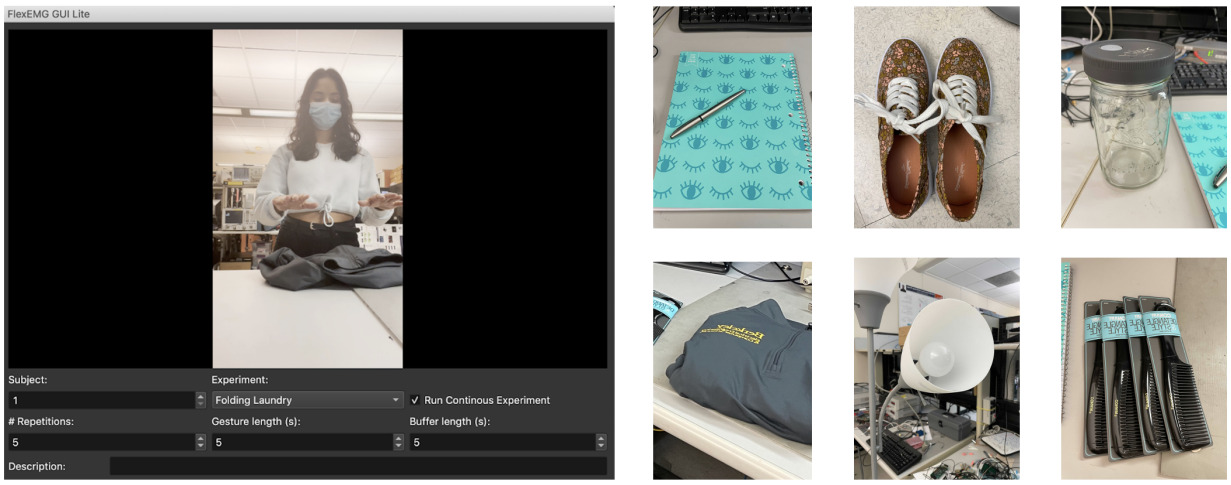


Figure 4.2: The data collection GUI and objects for common activities of daily living.

for attachment to the tips of the subject's fingers.

As the system is setup for 64 channels, 5 of the channels were re-routed from EMG channels on the edge of the array to the 5 force sensors. All 64 channels (59 EMG, 5 force sensor) were sampled and digitized at 1 kS/s using the custom-designed sensing and digitizing neural interface IC35 that maintains very low power consumption (700µW) [26]. In operating WAND, the wide input voltage mode with a range of 400 mV was used to allow for a larger range of force measurements. The device wirelessly connects to a base station computer with a 2.4 GHz radio SoC for data streaming and re-configuring. The neural interface was powered using a single 3.7 V, 240 mAh lithium-ion battery weighing 4g. The full system, shown in Figure 4.1, tracks motion, sEMG signals, and force exerted on each fingertip.

The current dataset includes 4 able-bodied, adult subjects[1] [45]. Data was collected using a graphical user interface (GUI) to guide subjects through the 6 ADLs shown in Figure 4.2. For each ADL, the subject first watched a tutorial. Then, the subject did 5 repetitions of the ADL in a controlled time frame during which they were guided through the sub-tasks comprising the ADL. Subsequently, they did 5 repetitions of the ADL at their own discretion, unguided and with unspecified sub-task timing.

During the controlled time frame, each of the 6 ADLs were broken down into sub-tasks guided to last 5 seconds. The sub-tasks were selected to correspond to changes in prosthetic actuation (e.g releasing, lifting, changing of grip, etc.). "Folding laundry" consisted of 8 sub-tasks to lift and fold a jacket. "Writing with a pen" consisted of 15 sub-tasks to use a two-finger pinch to write "hello" on a paper. "Opening a jar" consisted of 14 sub-tasks to untwist and remove a jar cap. "Screwing lightbulb" consisted of 13 sub-tasks to screw a lightbulb into a lamp. "Combing hair" consisted of 16 sub-tasks to pickup and comb through hair 4 times. "Tying shoelaces" consisted of 9 sub-tasks to use a mirrored L grip and closed pinch

[1]Dataset & scripts available at https://github.com/alisha-menon/ADL-data

to tie the laces. During the variable time frame, the subject was instructed to complete each ADL under 20 seconds freely, without guidance. All experiments were performed in strict compliance with the guidelines of IRB and were approved by the Committee for Protection of Human Subjects at University of California, Berkeley (Protocol title: Continuous EMG for Learning Activities of Daily Living Study, Protocol number: 2021-11-14838).

Most existing sEMG datasets are limited to specific movements or static gestures [61, 41, 31]. Prior sEMG datasets that were collected during ADLs do not include feedback information, which a controller could heavily benefit from for object manipulation, or labels for various parts of the ADL, which could enable more specific grasp selection and timing [23]. This dataset collects sEMG and accelerometer information during ADLs along with the critical feedback force sensors, includes labels for sub-tasks and also adds the controlled timing scenario in which executed tasks are guided under specific timing [45]. The controlled time frame serves as a foundation for analyzing data collected in the variable, unguided time frame which is also included. The two levels of timing complexity aid in control scheme exploration and development.

The proposed shared control system is evaluated on this multi-sensor dataset for 6 ADLs (with 75 total sub-tasks or short-term behaviors) for 4 able-bodied, adult subjects.

4.3 Proposed System

Overview

The multi-layer shared control scheme designed and built in this work is shown in Fig. 4.3. The lowest layer acquires feed forward sensor data: EMG and accelerometer data on the x, y and z axes collected on the forearm. This information is used to determine the user's long-term goal (ADL) using an HDC classifier. A time series classifier recognizes the most recent sub-task completed by the user within the selected ADL using the accelerometer data. Because the classification layer classifies on prior sensor data, it determines what has already happened and therefore can not direct actuation that corresponds to the user's actions at the present. To solve this, we include a probabilistic layer that encodes the user's habitual sequences of sub-tasks used to accomplish an ADL in order to predict sub-tasks. This information is used to predict the sub-task that the user intends to do at the present based on their most recent sub-task recognized by the time series classifier. Prior work has successfully projected probability tables to HDC vectors [19]. While we use a traditional probabilistic model in this work, we make initial steps towards an HDC representation through taking input sensor data in the form of HDC vectors instead of raw data. Future work will fully replace this layer with an HDC representation making the full multi-layer control scheme majority HDC and able to take advantage of HDC properties like computational efficiency, low latency, and quick model updates. The output of this layer is provided to the shared control layer as the human-driven control.

Currently, when being fit for a myoelectric terminal device, patients are asked to describe

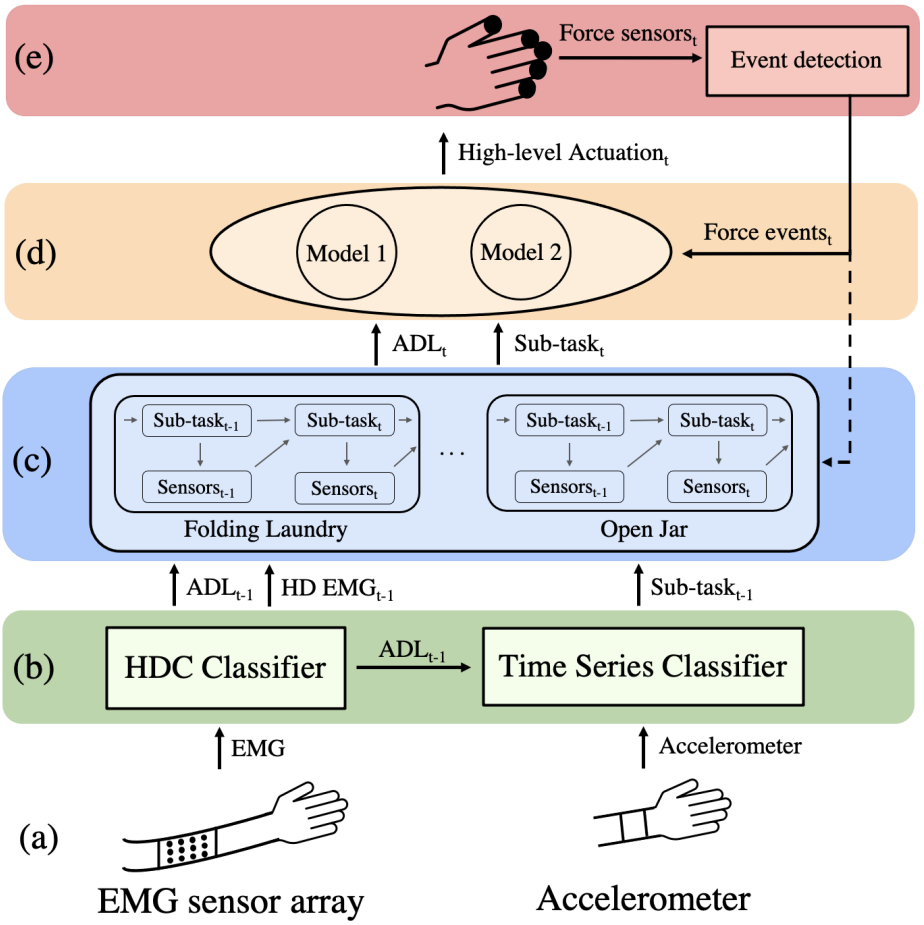


Figure 4.3: Multi-layer shared control scheme with (a) sensor inputs (b) user behavior recognition (c) user intent prediction (d) shared control with HDC RORB (e) actuation & feedback.

their activities of daily living (ADLs) – this determines the order and range of grips preprogrammed into the hand. ADLs can include a wide range of activities, each with its own unique set of necessary grips. For instance, if a patient enjoys working on their car, they would need a more durable hand, capable of being scratched and waterproof in case oil leaks on it. Others are more widely selected such as brushing one's hair or feeding themselves, each of which requires specific grips.

Some examples of commonly used prosthetic grips in commercial devices include the power grasp, closed pinch, open pinch, key grip, and finger point with the pinch grips and power grasp being the most used overall [52]. For the power grasp, all 5 digits begin to flex, surrounding the object of interest [72]. This grip is useful when the user would like to hold something or pick up an object. During the closed pinch, digits 3-5 are flexed while the first and second digits function by opening and closing. Conversely, during the open pinch, digits

3-5 remain extended while the first and second digits open and close [72]. These grips are examples of the high-level prosthetic behavior that must be outputted by the shared control system.

The shared control also depends on feedback from force sensors on the fingers of the end-effector during actuation. The shared control is achieved through HDC RORB which is designed to intelligently use both the human-driven control and the feedback-based automatic control to determine the appropriate reactive high-level prosthetic actuation. In this work, each layer was implemented individually and developed for high accuracy. During integration, an error correction scheme was included to eliminate accumulated error or performance degredation. Thus, high performance functionality is enabled for prosthetic shared control.

User Behavior Recognition

The lowest layer of the system recognizes the user's ADL. The HDC classifier used includes a spatial encoder, temporal encoder and associative memory as outlined in Figure 4.4.

We used the controlled time frame dataset collected for the 4 subjects and designed an HDC framework for task recognition. The algorithm is shown in Figure 4.4 for the example ADL "screwing lightbulb". The user reaches for a lightbulb, moves it towards the lamp, then proceeds to screw it in with periodic rotations of the wrist. During data collection, the data was labeled by ADL and sub-task. In the controlled time frame, there were 5000 samples per sub-task. Feature extraction involved a mean absolute value over 50ms non-overlapping windows selected based on [55]. Hence, each sub-task is represented by 100 data points per EMG channel.

The proposed algorithm then maps these features into the HD space. The basic building block of the algorithm is hypervectors (HVs), bipolar vectors with dimensions in the thousands. An item memory (iM) consists of a set of randomly generated HVs (randomly placed equal number of -1's and $+1$'s). These are pseudo-orthogonal in the HD space due to the high dimensionality [28]. Using point-wise multiplication/addition, scalar multiplication, and permutation, HVs can be combined while still preserving information due to the orthogonality. The classification procedure involves spatially encoding by assigning each channel an iM HV, scaling by the channel feature value and then summing across channels and bipolarizing to generate a single HV per time step [55]. The temporal encoder uses 2-grams, selected by parameter sweep. This involves point-wise multiplication of 2 consecutive samples with the later being permuted to encode patterns over a 100ms window.

All the information within the relevant sub-task time window is then summed, bipolarized and then compared against stored classes in the associative memory. The closest class by cosine similarity determines the prediction. Leave-one-out cross validation was used for the 5 examples of each ADL to determine task recognition accuracy. This process allows us to determine the long-term intent of the user based on their continuous motions and corresponding EMG signals involved in accomplishing it.

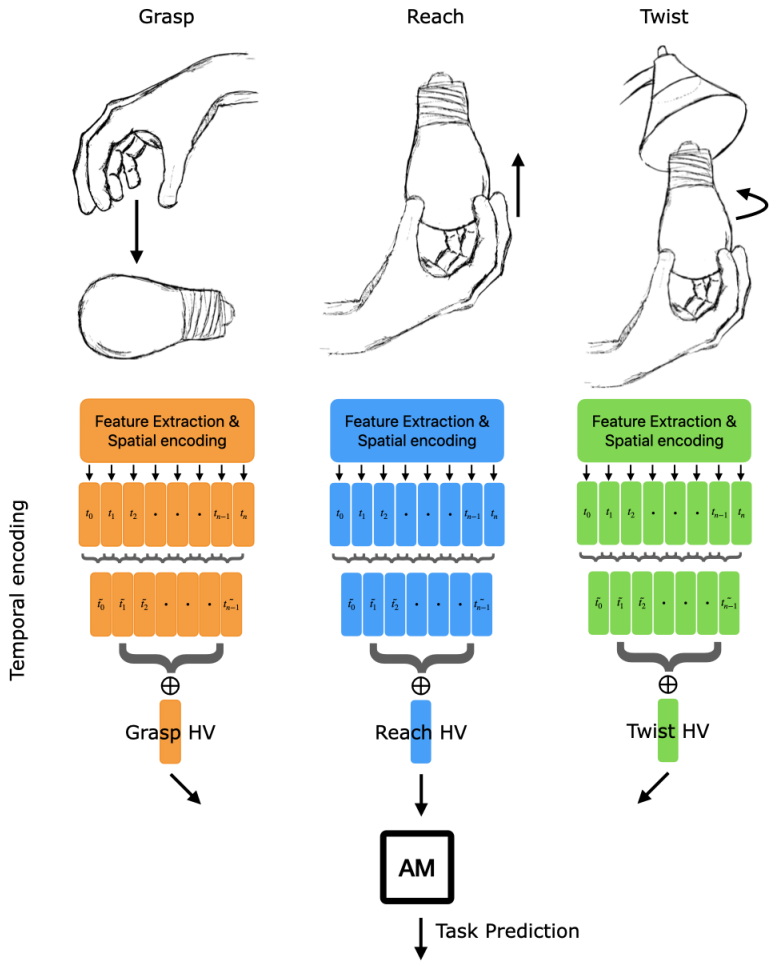


Figure 4.4: Task prediction of continuous motion, shown for "Screwing lightbulb". Raw data is spatially and temporally encoded, bundled across a sub-task, and classified in the associative memory.

ADL recognition provides long-term information to the system, but does not provide information on the short-term actuation the user is doing. While EMG primarily reflects arm position and hand gestures, accelerometer signals reflect arm movements and hence sub-tasks. To recognize the temporal accelerometer sub-task patterns, we used ROCKET, a method that transforms time series through random convolutional kernels and uses these transformations to train a linear classifier [14]. ROCKET performs similarly to state-of-the-art time series classifiers while requiring far less computational resources [15]. We train and test ROCKET with the 5 examples of each sub-task (each is 5s, 100 samples) using leave-one-out cross validation. The repetitive sub-tasks in ADLs like *opening jar* and *combing hair* such as *untwist* or *comb through hair* are given the same label. Through iteration, we included additional features for each channel on top of the raw data including standard

deviation over a moving 10-sample window and jerk (change in acceleration) to improve sub-task recognition.

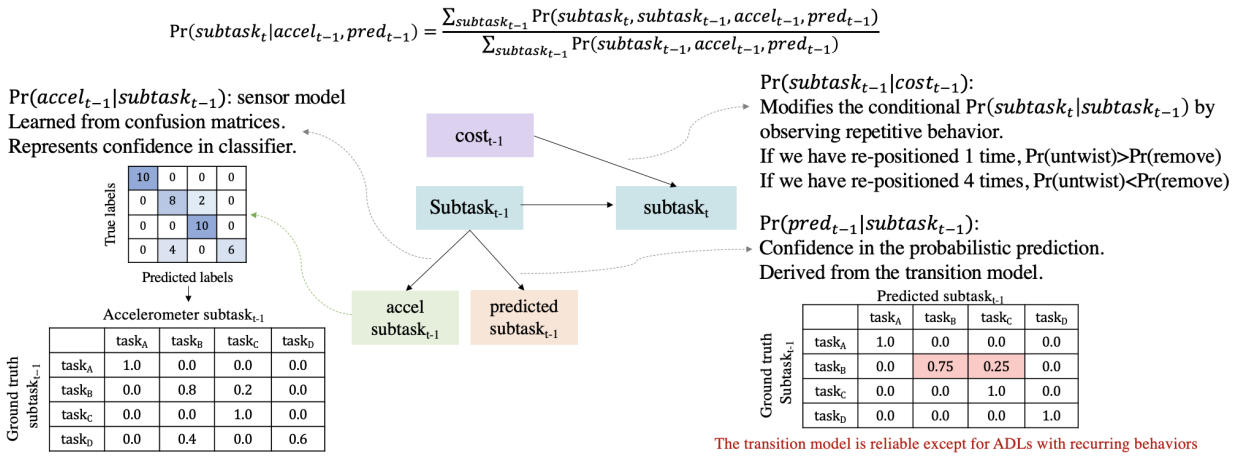
User Intent Prediction

The second layer comprises a collection of Dynamic Bayesian Networks (DBNs), one for each ADL. The goal is to execute short-term intent prediction, as shown in Fig. 4.5 which depicts a singe slice of the DBNs, by taking into consideration the most recently recognized sub-task and the prior prediction made by the DBNs. This allows the system to predict the sub-task the subject is currently aiming to complete so that the shared control algorithm can assist in real-time instead of waiting to recognize the sub-task from prior user behaviors at which point the subject has already completed sub-task's arm motions. A naive model makes a prediction only based on the prior accelerometer classification and the known sequences. However, the sub-task recognition does not perform with 100% accuracy which can cause errors if purely making predictions based on that.

Similarly, each DBN relates the sub-tasks probabilistically over time, for example in the *open jar* ADL, *untwist* (cap) follows (hand) *realign* most frequently, except when the user has finished unscrewing in which case *place cap* follows. This results in errors as well if the user is following an unlikely behavior which primarily occurs during ADLs with recurring behaviors: *opening a jar*, *screwing in a lightbulb* and *combing hair*. Hence, even given 100% correct sub-task classifications, the probabilistic layer can't be relied on to always correctly predict the next sub-task. Worse, while using the inevitably imperfect sub-task classifier, errors from misclassifications will accumulate with errors from the probabilistic layer resulting in lower performance than either layer alone.

To adjust for this, a confusion matrix is generated during training based on error in accelerometer classification and the transition model for the probabilistic prediction as shown in Figure 4.5. During testing, this confusion matrix is used to generate the likelihood of a sub-task at time t based on the prior sub-task as recognized through accelerometer data and by the prior prediction made by the DBN. This query favors the more reliable source of information and hence corrects for errors commonly made by either the classifier or the prior prediction by selecting the next prediction based on the more reliable guess of the prior sub-task. This significantly improves the performance over predicting only based on the prior classified sub-task or only based on the prior DBN prediction because the accelerometer classification and probabilistic prediction are generally not unreliable at the same time. Additionally, this model adjusts to potentially unpredictable behaviors by keeping up with the most recent accelerometer data instead of only following a set sequence.

To address the challenge of correctly predicting future sub-tasks in ADLs with recurring behaviors to improve the prediction of the probabilistic layer, we introduce an additional cost variable, which modifies the transition model depending on the number of prior recurrences. For example, if the user has only realigned their hand one time, untwist is very likely to follow, but if the user has realigned their hand already 4 times, then remove should be more likely. Thus, the cost variable enables modification of the appropriate conditional



	task$_A$	task$_B$	task$_C$	task$_D$
task$_A$	1.0	0.0	0.0	0.0
task$_B$	0.0	0.8	0.2	0.0
task$_C$	0.0	0.0	1.0	0.0
task$_D$	0.0	0.4	0.0	0.6

	task$_A$	task$_B$	task$_C$	task$_D$
task$_A$	1.0	0.0	0.0	0.0
task$_B$	0.0	0.75	0.25	0.0
task$_C$	0.0	0.0	1.0	0.0
task$_D$	0.0	0.0	0.0	1.0

Figure 4.5: Probabilistic model depicting a single slice of the DBNs, integrating error correction and a cost variable for recurring behaviors. The query favors the more reliable source of information.

distribution, and inference is made. In general we infer future sub-tasks by evaluating the query $P(subtask_t|accel_{t-1}, predicted_{t-1}, cost_{t-1})$ on the relevant DBN at each time step.

Shared Control with Recall of Reactive Behavior

The previous layer predicts the current sub-task and the current ADL is passed all the way up from the HDC-based ADL classification in the lowest layer. We combine these with feedback force sensors on the end-effectors fingers to determine the desired actuation of the prosthetic device. This occurs via an HDC RORB scheme in which the ADL, sub-task and force information are all equally weighted in determining the actuation. The primary benefit of using HDC for this layer is the ability to fuse the various sensor input streams efficiently [46], [7].

The force data is processed to detect one of four states: no contact, touch, grip with slipping, or gripping. Each of these is mapped to a pseudo-orthogonal hypervector (HV) that gets bound (via the XOR operator $\oplus$) to the output of the ADL–sub-task binding and then to the associated actuation HV for that state. The fused vectors for the various ADL, sub-task, and force combinations get bundled together with equal weighting into a hypervector we call the program memory or program vector. This process is explained in more detail with several variations in [42]. However, there are several sub-tasks for which all 4 force states will have the same actuation. This occurs in force-independent sub-tasks, such as the L grip actuation which has a prescribed final position.

Given 75 unique sub-tasks across all ADLs and 4 force states, there are 300 sub-task-force-actuation combinations that get bundled together with equal weight into the Program

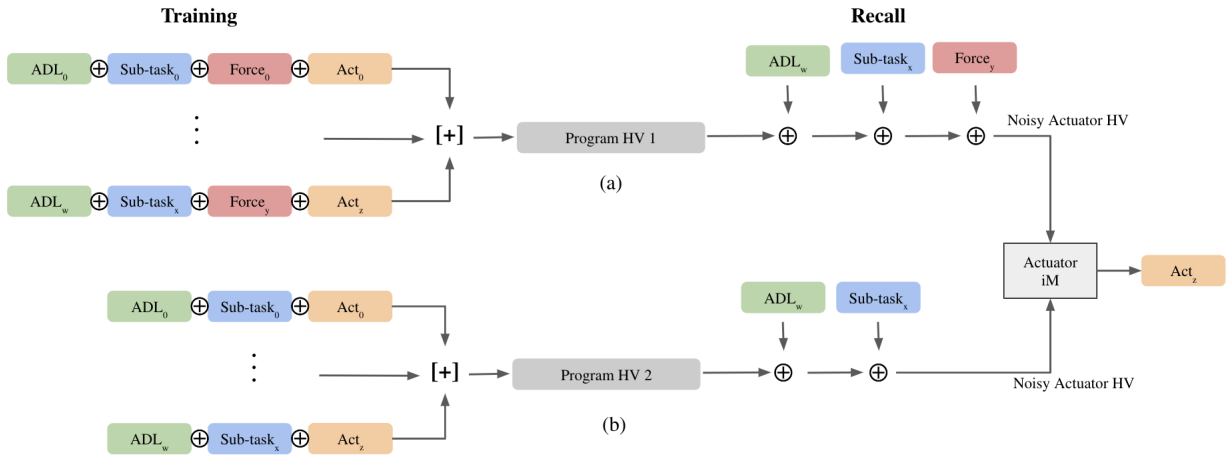


Figure 4.6: The adopted HDC RORB algorithm with (a) the originally proposed algorithm combining all training data into a single PV, representing both the naive method and the version in which force-independent cases are pruned to prevent redundancy and (b) the added second PV for the force-independent subset of the data which results in higher recall performance.

Vector (PV) as in Eq. 4.1. The appropriate actuation can be extracted in real time by sequentially binding each of the HVs associated with input values to the PV. The resulting HV is a noisy approximation to one of the actuation HVs and the appropriate actuation HV can be extracted by comparison with an item memory. See Fig 4.6.

$$PV = \sum_{w,x,y,z} ADL_w \oplus S_x \oplus F_y \oplus Act_z \tag{4.1}$$

However, this naive implementation doesn't take into consideration the 22 sub-tasks that are force-independent. Each of these sub-tasks will have 4 training samples in the current training scheme, 3 of which are redundant. Exploiting this, we store only the training sample for the no contact state for force-independent sub-tasks. Similarly, at test time we query only the no contact state for the force HV binding. This reduces the number of training samples from 300 to 234 which decreases the magnitude of noise in the recovered noisy actuation HV, increasing the probability that we can correctly identify the right entry in the item memory.

Given that each training sample in the bundle contributes equally to the noise during extraction, we can further increase our accuracy by adding a second PV. We split our model into two parallel channels for the force-dependent and independent sub-tasks. Our primary channel is the same as our original naive model in which we bind ADL, sub-task, force, and actuator HV together and bundle those together into a program vector (now PV1). However, for our force-independent sub-tasks, we bind only the ADL, sub-task, and actuation HV together – without the force HV – and bundle all of these into PV2. This further decreases

the number of training samples stored in PV1 to 212, with only 22 samples stored in PV2. This third model alleviates noise during extraction at the expense of slightly more processing. We implement and evaluate all three models in Section 4.4. There are 19 unique high-level actuation outputs that are passed to the end-effector ranging from *Wrist flex* to *2 finger closed pinch*. This full set of 19 prosthetic behaviors was compiled based on the needs of the ADLs included in the dataset [45], commonly used prosthetic grips and gestures, and prosthetic responses during interactions with objects.

User-adaptivity

This system is designed to be able to adapt to users through personalization based on the user's activities of daily living, similar to commercial prosthetics. Though data was collected for and performance evaluated on a set of 6 ADLs, the control scheme can adjust to a new user's ADLs and sub-tasks. The probabilistic layer stores information on these sequences and the likelihood of a next sub-task given the most recent user behavior and prior predictions. The actuation mapping for the range of potential sub-tasks is stored in the HDC RORB layer. Both of these would require a one-time setup when the user is fitted for their prosthetic. The user would then need to provide demonstrations of each ADL/sub-task with corresponding EMG and accelerometer data in order to train the classifiers to recognize them. The dataset used in this work included 5 examples of each, though the classifiers can theoretically be trained on just one example instead of four. Future studies are needed to validate the one-shot learning property for the full system.

4.4 Experimental Results

EMG & Accelerometer Classification

The results of task recognition with HDC on the collected continuous motion EMG are shown in Figure 4.7 for each subject per ADL and overall. The task recognition accuracy shows the ability to map the 74 sub-tasks to the correct ADL, averaged across 5-folds for leave-one-out cross-validation. The average accuracy across all 4 subjects was 92.5% with subject 3 performing the highest at 93.6%. The performance difference among subjects is attributed to variation in arm sizes/hair resulting in discrepancy in sensor contact and signal quality. Additionally, subjects vary in task emphasis and consistency which can affect the ability to recognize the task.

Most tasks are recognized with greater than 90% accuracy although certain tasks for subjects show lower performance such as "screwing lightbulb", "writing with pen" and "folding laundry" for which subject 1 performs 16% worse, subject 3 performs 10% worse, and subject 2 performs 10% worse than overall, respectively. Because these activities require changes in arm angle, some subjects' low performance is hypothesized to be due to a combination of reduced sensor contact in certain arm positions, which varies per subject, and inconsistency

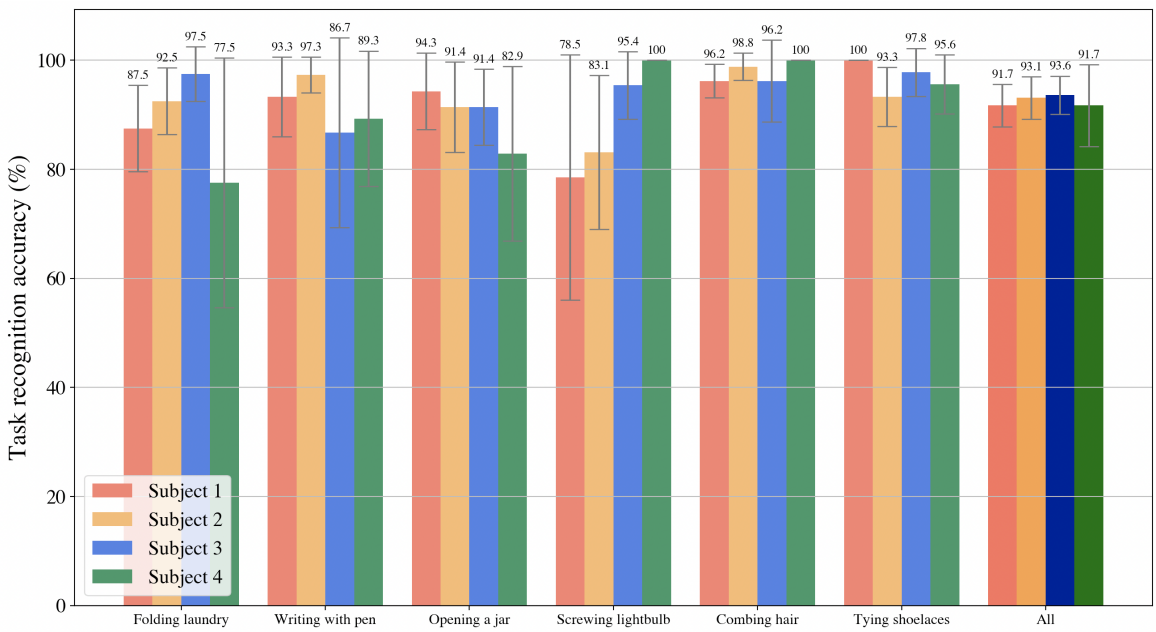


Figure 4.7: EMG-based task recognition accuracy with HDC per ADL and overall for the 4 subjects. The error bars indicate standard deviation.

in task completion. However, overall, the ability to distinguish between the different tasks is satisfactory, especially given the 74 different sub-tasks over the 6 ADLs that are being recognized and correctly mapped. Several of the sub-tasks are similar between different ADLs as well since more than a few include sub-tasks such as "reach out" and "pick-up", increasing the significance of this result.

Figure 4.8 shows the similarity between sub-tasks measured with cosine similarity between the sub-task hypervectors with each row/column corresponding to a sub-task, shown for the highest performing subject. Overall, there is a clear separation between ADLs while the sub-tasks within each ADL are very similar. However, within specific ADLs there is some dissimilarity, such as in "combing hair" where the initial and later sub-tasks are dissimilar. This can be explained by the fact that each ADL occurs at varying arm positions; this is known to heavily change the EMG signals that are produced [25]. This explains the similarities shown for an ADL such as "combing hair" where the initial sub-tasks are done with the hand at waist-level, while the later ones are done at head-level. Hence, an important factor in detecting the ADL is the change in arm position that can be observed from the EMG signals.

These results show that this simple and efficient algorithm can successfully predict the long-term intent regardless of the part of the task the subject is engaging in, even the first sub-task. This provides valuable information to the controller which can now select an intent-specific model to anticipate behaviors and assist accordingly. In the proposed multi-layer control scheme, this outcome is provided to the time series classifier so that it is recognizing

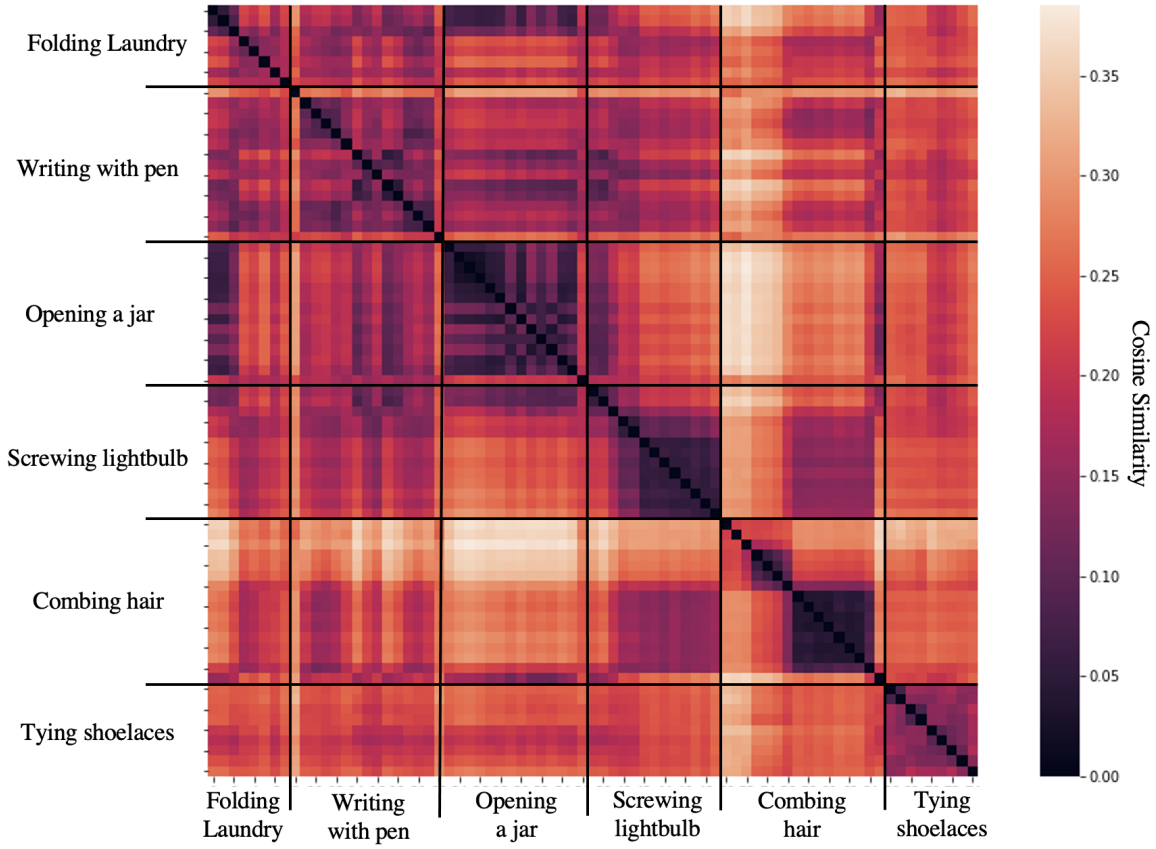


Figure 4.8: Similarity matrix of the hypervectors for each sub-tasks within the ADLs measured with cosine similarity.

from the more specific set of sub-tasks within the recognized ADL.

The time series classification of the accelerometer sensor data achieves an average sub-task recognition accuracy within each ADL of 91.7% across all the subjects with leave-one-out cross-validation over 5 examples, shown in Fig. 4.9. This algorithm also sees the same varied performance across tasks between different subjects which, in this case, is attributed to inconsistencies in the exact method of accomplishing each ADL between examples. Overall, the performance of both classifiers is very high, especially given the 75 different possible sub-tasks, and showcases the utility of both the EMG and accelerometer information in recognizing the user's behaviors.

Probabilistic Model

Fig. 4.10 shows the ideal prediction accuracy of the naive DBN, calculated over the set of all sub-tasks for each ADL starting at $t = 1$ assuming the prior accelerometer classification is correct. This figure demonstrates the performance of the probabilistic layer alone. Note that, unlike time series sub-task classification, we consider repeated sub-tasks because one goal of

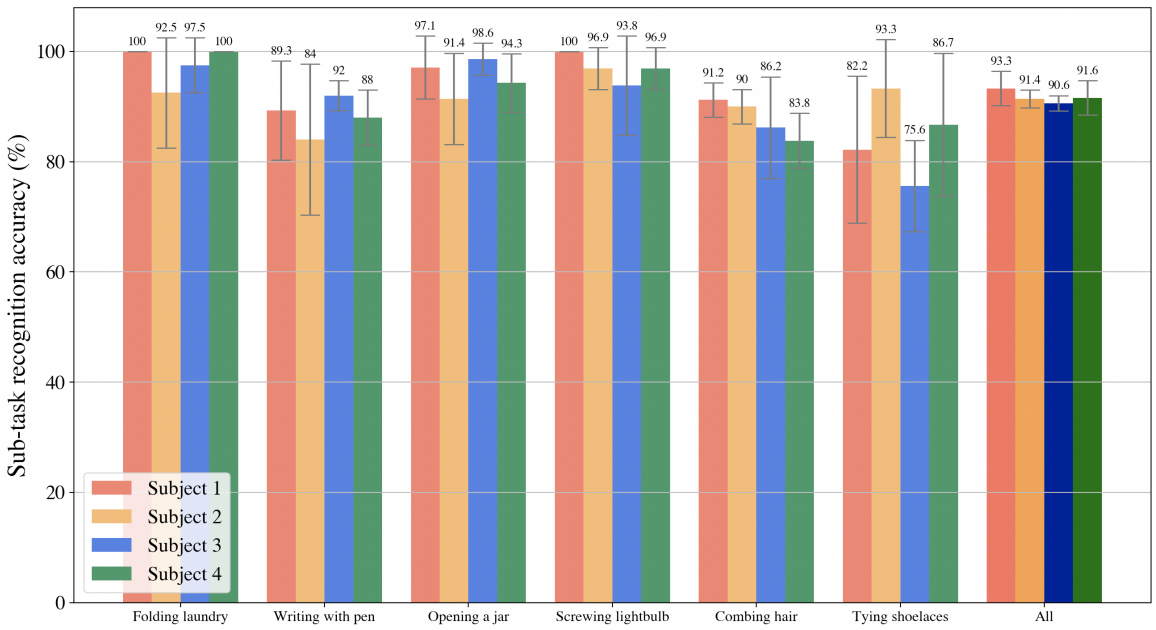


Figure 4.9: Sub-task recognition accuracy with a time-series classifier, given an accurate ADL from the HDC classifier. The error bars indicate standard deviation.

this layer is to correctly predict future sub-tasks in ADLs with recurrent behavior despite the number of times this behavior is repeated. This allows the user to repeat the sub-task as needed (e.g. prediction must be successful whether the user opens a jar by twisting the cap 3, 4, 5, etc. times). Using just the predictive algorithm, ADLs that present recurrent behaviour attain lower accuracy as the repetitions of this recurrence is non-deterministic, the rest achieve perfect performance.

The results in Fig. 4.4 shows the performance of the probabilistic model when considering the non-ideal output of the accelerometer-based time series classifier from Section 4.4. Overall, the average prediction accuracy is now 82%, a 10% degradation. This end-to-end implementation is affected by the error accumulating throughout the layers. Accuracy degradation is most severe for subject 4 in *tying shoelaces*, which indeed corresponds to the lowest time series classification accuracy shown in Fig. 4.9.

Using the scheme to instead deliberate between the prior accelerometer classification and the prior prediction to make the next prediction based on confidence in each model enables for elimination of accumulated error and instead uses the strengths of each model. This is shown in Figure 4.12 which demonstrates the increase in performance for each subject when comparing the naive integration-based prediction with the error-corrected prediction. The final performance of the predictive layer is recovered to 92.6%.

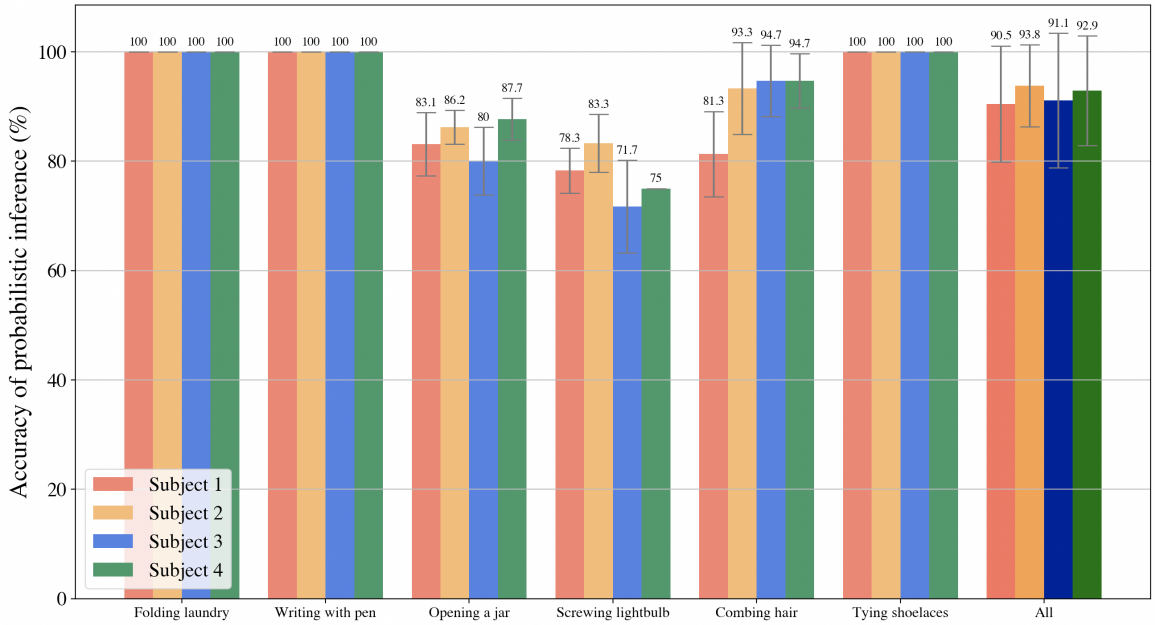


Figure 4.10: Ideal accuracy of sub-task prediction using the probabilistic layer assuming correct prior sub-task classifications.

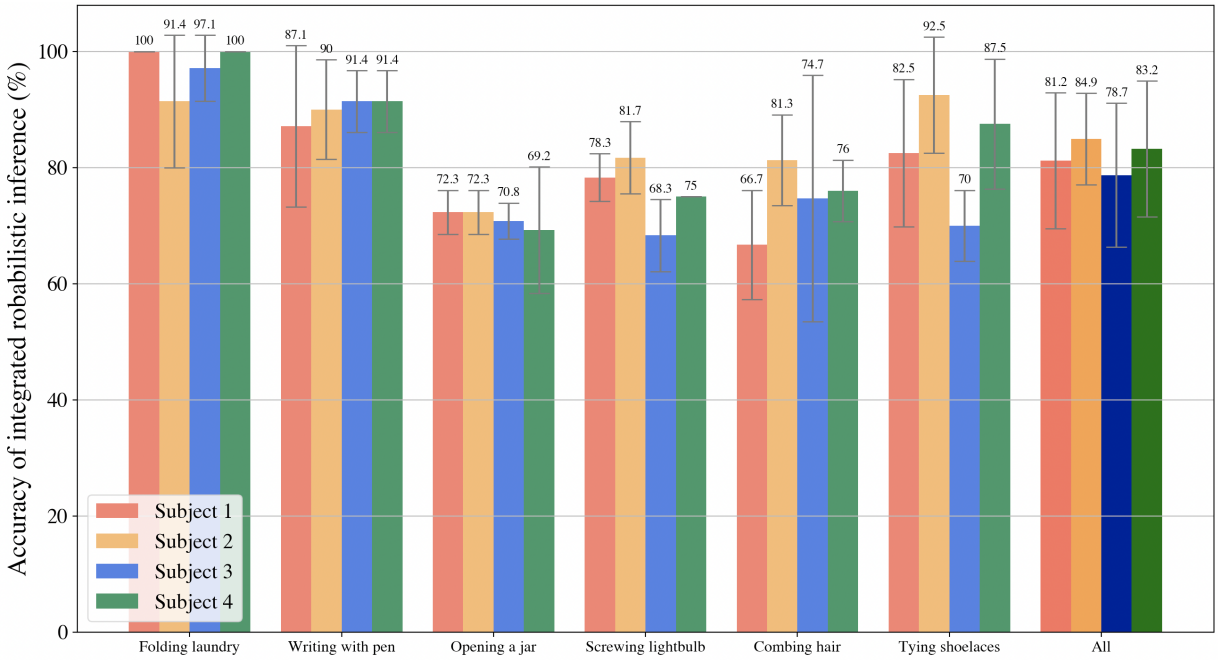


Figure 4.11: Sub-task prediction performance when integrating error from prior sub-task classification.

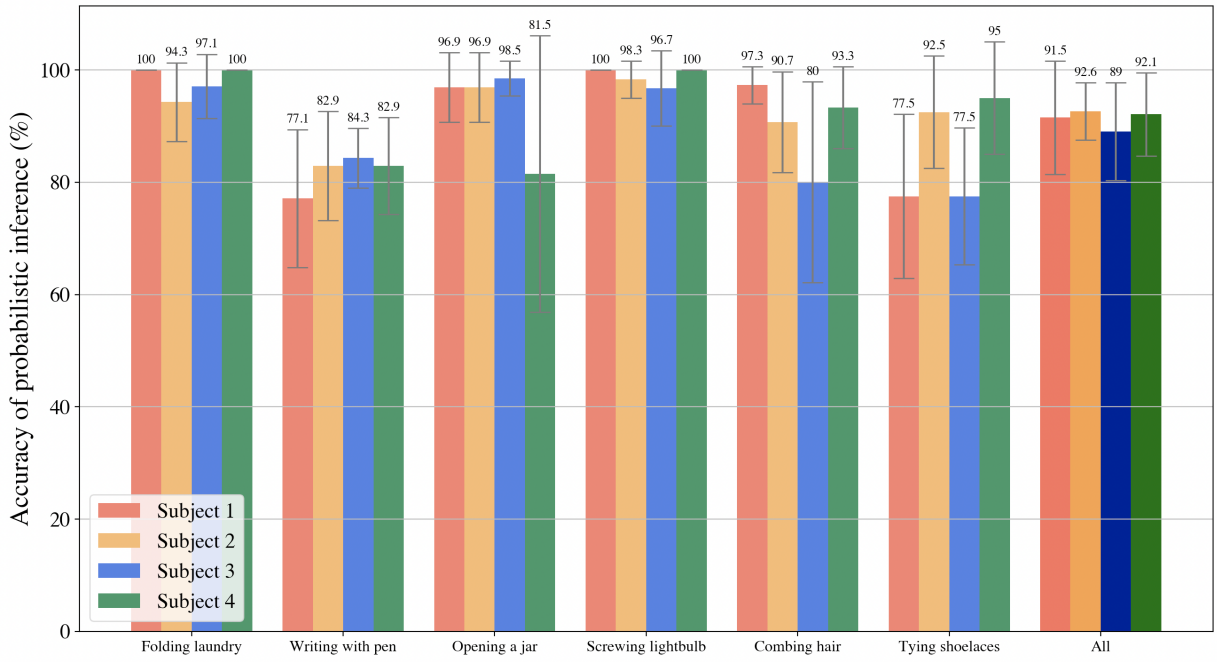


Figure 4.12: Sub-task prediction performance when using the error correction scheme.

HDC Recall of Reactive Behavior

The HDC RORB control system first determines which PV to query based on a flag passed by the probabilistic layer which contains a map of all ADLs and their sub-tasks including their force dependency. We then bind the ADL and sub-task HVs to the appropriate PV. In the case where force feedback is not relevant, we are left with a noisy approximation of the actuation HV which we clean up via an item memory. When force feedback is necessary, we identify the appropriate force state: no contact, touch, grip with slipping, or grip, and bind that HV to the earlier HV. This produces a noisy actuation HV for further clean-up.

Fig. 4.13 shows the accuracy of our adopted HDC RORB method, as well as the two simpler models implemented, as a function of the dimensionality of our HD space. We highlight performance at a dimensionality of 10,000 due to its common usage in the HDC community [28]. The total accuracy of 98.4% in the chosen model is due to the combining of a 100% accurate force-independent model (PV2) and 97.9% accurate force-dependent model (PV1). The difference in recall accuracy between PV1 and PV2 can be entirely attributed to the difference in number of unique psuedo-orthogonal sample vectors stored in each. The force-independent model stores a total of 22 training samples, while the force model stores 212 training samples. This shows that 212 vectors begins to exceed the maximum storage capacity of an hypervector of dimensionality 10,000, additionally demonstrated by the two simpler models which store even more samples into their single PVs. In terms of computational simplicity, though the chosen method uses two program HVs, to achieve the same accuracy, the simpler methods would require an overall increase in HD dimensionality

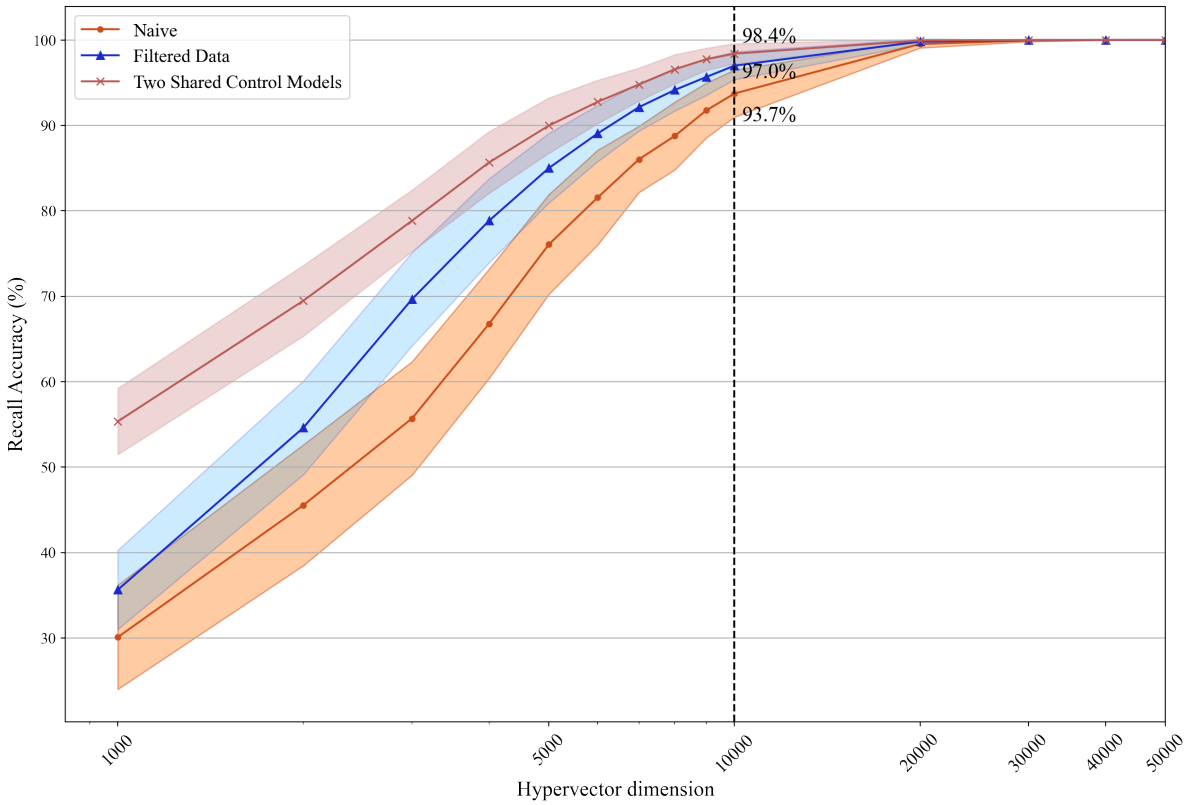


Figure 4.13: Shared control prosthetic actuation recall accuracy for HDC RORB implemented naively, after eliminating redundancy, and with two shared control models. Shaded area shows standard deviation.

which would cumulatively be much more expensive.

Actuation

Given the appropriate high-level actuation from the HDC RORB layer, we are able to implement these actuations in simulation as well as with a physical robot hand. We are currently primarily using the simulation to model the end-to-end system including feedback. The simulation environment uses the PyBullet physics simulation along with the Shadow Dextrous Hand to model a human hand and its movements [66]. This model has 20 actuated degrees of freedom and is able to replicate the movements of a human hand almost entirely (the human hand has 27 degrees of freedom [65]). Objects used in ADLs are instantiated from OpenAI's Roboschool Objects [5]. This allows for testing of shared control with the shadow hand and its interaction with a wide range of objects. The simulation environment provides force feedback at the contact points between objects mirroring the force sensors on the physical setup. The Shadow Dextrous Hand is used in combination with the JACO arm, replacing it's original end effector, which enables maneuvering of the end effector to

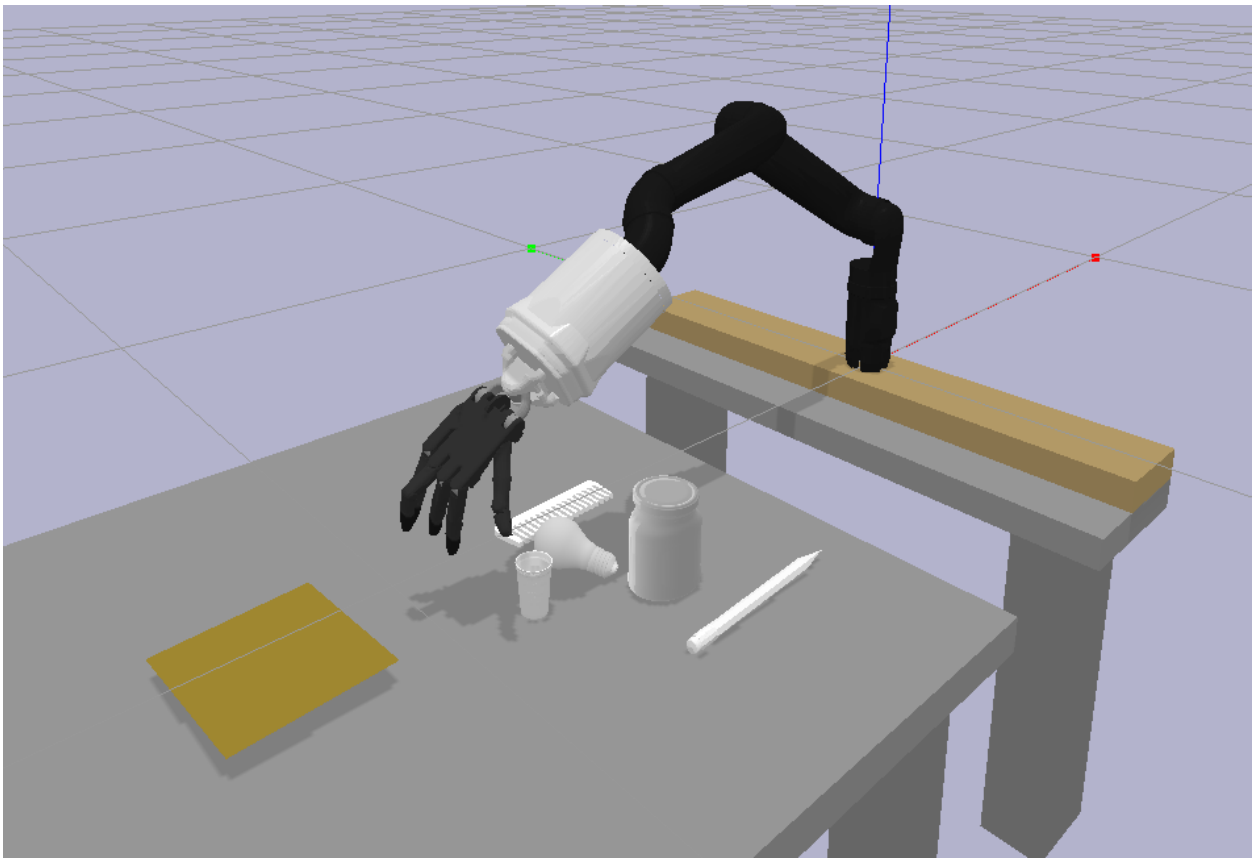


Figure 4.14: Simulation environment with the Shadow Dexterous hand and all the various ADL objects including a cloth for folding, a lightbulb, a jar, a pen, a comb and a cup.

the desired location using inverse kinematics. All the various objects for the 6 ADLs in the dataset collected are also present in the simulation environment, as shown in Figure 4.14. The hand can interact with them to simulate completion of the various ADLs.

Hardware Efficiency

The full shared control system was designed while prioritizing hardware efficiency to enable in-sensor implementation. In this section, we conduct a first-order analysis of the various layers to estimate the total operations and memory.

HDC classification is highly efficient, as demonstrated extensively in Chapter 2. Prior work using HDC for EMG classification has shown usage of 43,384 LUTs and 45,477 DFFs [55], good estimators of the total number of operations and memory, respectively. The algorithm used in the first layer of the prosthetic control scheme is very similar, except that it involves bundling over a longer time window. Therefore, the hardware utilization of this part of the lowest layer will be similar.

The first layer also includes a time series classifier. Traditional implementations require heavy computation in the form of matrix multiplications which would make this block likely to dominate the entire prosthetic control scheme in terms of operations/memory. To alleviate this, we are currently working on an HDC implementation. Preliminary work suggests that alternate, potentially more complex, feature extraction is needed in order to generate equivalent accuracy to the ROCKET classifier. Thus, the feature extraction will likely remain the most expensive part of this classifier, as it is for EMG classification [55], despite the significantly reduced input channels (3 for the accelerometer - one for each axis, compared to 64 for EMG). However, the overall HDC accelerometer classification will be several times more efficient than the HDC EMG classifier due to the reduced input data streams which especially affects the spatial encoder. Even with more complex features, the feature extraction and spatial encoder will be reduced by a factor close to 20x as compared to EMG classification, resulting in an estimated 21,000 LUTs (50% decrease) and 34,000 DFFs (25% decrease).

The second layer of the system is the DBN. Prior work has demonstrated that DBNs can be effectively implemented with HDC representation with minimal performance degredation [19]. A traditional implementation requires storage of a few large tables of floating point probability values in order to generate predictions. With HDC, each table is reduced to a single hypervector through superposition. Probability values can be retrieved from these hypervectors through an unbinding (XOR) and descaling process (cosine similarity and the floating point division), similar to the reactive robotics algorithm. For the probabilistic layer described in this work, the necessary tables include the confusion matrices used to estimate confidence in classifications generated by the time series classifier, and in predictions generated by the DBN described by the transition model. This results in two hypervectors being stored for each ADL, one for the confusion matrix model and one for the transition model. Additionally, each subtask requires a unique pseudo-orthogonal hypervector. However, the technique proposed in Chapter 2, CA *rule 90*, can be used to reduce the hypervector storage to a single hypervector from which the rest can be generated. Therefore, the total storage of 13 hypervectors includes 12 table hypervectors and 1 seed hypervector. Computing a posterior distribution over $subtask_t$ involves retrieving three values from the two probability tables. This is done for each possible $subtask_{t-1}$ to generate a probability of a particular current sub-task, and then for each possible $subtask_t$ to generate a distribution and, from that, a prediction. This process takes an estimated 1536 operations per hypervector bit due to the number of sub-tasks that must be iterated over (the longest ADL has 16 subtasks). To first order, if D is the hypervector dimension, this results in $1536 \times D$ operations per prediction and $13 \times D$ bits in memory for this layer. Implementation with HDC and then further analysis will indicate the minimum hypervector dimension that maintains prediction accuracy which will then inform whether the HDC implementation provides hardware benefits. The alternative would be storage of the probability tables as simply lookup tables which would require greater storage due to the decimal nature of the probabilities, but offer reduced computation for value retrieval.

The HDC reactive robotics algorithm is shown in Chapter 3 to be exceedingly simple.

The specific algorithm used to recall high-level prosthetic behaviors uses 51 unique sensor/actuation pseudo-orthogonal hypervectors, but these can be generated using CA *rule 90*. Therefore, the total storage of 3 hypervectors includes the 2 trained program hypervectors and a single seed hypervector. The overall computation for this algorithm includes XORs, additions, comparisons and hamming distance operations, totalling to around 42 operations per hypervector bit, the majority of which come from cleaning up the retrieved actuation with comparison against all 19 possibilities. To first order, if D is the hypervector dimension, this results in $42\times D$ operations per recall and $3\times D$ bits in storage for this layer.

Overall, each layer of the shared control scheme can be implemented with an algorithm that maintains minimal computation and storage, creating a structure that achieves a broad range of functions whilst remaining lightweight enough for wearable hardware constraints such as hardware resources and power. Future work will involve an FPGA implementation of the full system to measure LUT/DFF usage, latency, and overall power consumption.

4.5 Summary & Future Work

In conclusion, this chapter demonstrates the successful design of a multi-layer shared control scheme, primarily through HDC. Through classification, prediction, and reactive robotics, the full-stack performance was shown to achieve performance greater than 92%. Each layer has been implemented prioritizing hardware efficiency to enable in-sensor implementation of the full shared control algorithm.

In future work, we aim to demonstrate live experiments of the control scheme in simulation to observe the impact of real-time feedback. We also aim to do an FPGA implementation of the algorithm to observe the overall hardware efficiency and latency. Given the prosthetic application, the dataset must also be expanded to amputee studies. By fitting a population of right radioulnar amputees with a socket containing the 64 electrode array, we can compare our control scheme to the amputees' existing device. To standardize the study, we would fit each subject with the same type of socket and myoelectric terminal device. We can also track endurance levels on a subject basis, as our shared control system aims to minimize the fatigue amputees undergo when controlling their device for extended periods of time. By expanding the research to the prosthetic wearing community, future devices can be worn for longer periods of time and provide the user with more accurate and intention al movements.

Chapter 5

Final Thoughts

This thesis comprised multiple elements which came together for a neural prosthetic with a shared control scheme designed while prioritizing hardware efficiency to enable in-sensor implementation. It contains the first significant hardware comparison of the emerging hyperdimensional computing (HDC) paradigm to a traditional machine learning algorithm with results that demonstrate HDC's potential to become a paradigm of choice for tinyML applications. It also tackled HDC's usage for reactive robotics to bring the same property to the robotics field. The exploration of hybrid HDC-NN schemes shows the benefits of the HDC representation and opens the door for further hybrid models to leverage these alongside the benefits of other popular optimal control algorithms such as reinforcement learning. Finally, the neural prosthetic control system demonstrates the ability of the paradigm to provide in-sensor intelligence that addresses the versatile needs of even highly complex systems.

The methods proposed in this work can be leveraged for any closed-loop application involving biosignal processing, understanding, response and feedback. Applications include a broad range of tasks from brain-computer interfaces for restoration of movement, communication, rehabilitation, and memory augmentation, to artificial organs that monitor biological conditions and respond with the necessary enzymes, to chronic pain management systems that observe biosignal indicators of pain and deliver drugs to maintain patient comfort. Such systems generating intelligent responses to biosignals measured on, in, or around the human body have the potential to revolutionize the way we interact with machines, devices, and each other. Through in-sensor implementation, the door is open to untethered advancement of the human-machine interface.

Bibliography

[1] Ali Jaber Almalki and Pawel Wocjan. "Exploration of Reinforcement Learning to Play Snake Game". In: *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE. 2019, pp. 377–381.

[2] Krste Asanovic et al. "The rocket chip generator". In: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17* 4 (2016).

[3] Kenneth C Barr and Krste Asanović. "Energy-aware lossless data compression". In: *ACM Transactions on Computer Systems (TOCS)* 24.3 (2006), pp. 250–291.

[4] Elaine A Biddiss and Tom T Chau. "Upper limb prosthesis use and abandonment: a survey of the last 25 years". In: *Prosthetics and orthotics international* 31.3 (2007), pp. 236–257.

[5] Greg Brockman et al. "OpenAI Gym". In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[6] Alessio Burrello et al. "Laelaps: An energy-efficient seizure detection algorithm from long-term human iEEG recordings without false alarms". In: *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2019, pp. 752–757.

[7] En-Jui Chang, Abbas Rahimi, et al. "Hyperdimensional Computing-based Multimodality Emotion Recognition with Physiological Signals". In: *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE. 2019, pp. 137–141.

[8] Brian Cheung et al. "Superposition of many models into one". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 10868–10877.

[9] Adelson Chua, Michael I Jordan, and Rikky Muller. "A 1.5 nJ/cls Unsupervised Online Learning Classifier for Seizure Detection". In: *2021 Symposium on VLSI Circuits*. IEEE. 2021, pp. 1–2.

[10] Adelson Chua, Michael I Jordan, and Rikky Muller. "SOUL: An Energy-Efficient Unsupervised Online Learning Seizure Detection Classifier". In: *IEEE Journal of Solid-State Circuits* (2022).

[11] Francesca Cordella et al. "Literature review on needs of upper limb prosthesis users". In: *Frontiers in neuroscience* 10 (2016), p. 209.

[12] Juan Abdon Miranda Correa et al. "Amigos: A dataset for affect, personality and mood research on individuals and groups". In: *IEEE Transactions on Affective Computing* (2018).

[13] Sohum Datta et al. "A programmable hyper-dimensional processor architecture for human-centric IoT". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9.3 (2019), pp. 439–452.

[14] Angus Dempster, François Petitjean, and Geoffrey I Webb. "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels". In: *Data Mining and Knowledge Discovery* 34.5 (2020), pp. 1454–1495.

[15] Bhaskar Dhariyal et al. "An examination of the state-of-the-art for multivariate time series classification". In: *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2020, pp. 243–250.

[16] Manuel Eggimann, Abbas Rahimi, and Luca Benini. "A 5 μW Standard Cell Memory-Based Configurable Hyperdimensional Computing Accelerator for Always-on Smart Sensing". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.10 (2021), pp. 4116–4128.

[17] Monica Espinosa and Dan Nathan-Roberts. "Understanding prosthetic abandonment". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 63. 1. SAGE Publications Sage CA: Los Angeles, CA. 2019, pp. 1644–1648.

[18] E Paxon Frady et al. "Computing on Functions Using Randomized Vector Representations". In: *arXiv preprint arXiv:2109.03429* (2021).

[19] Laura Isabel Galindez Olascoaga et al. "A Brain-Inspired Hierarchical Reasoning Framework for Cognition-Augmented Prosthetic Grasping". In: *Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations*. 2021.

[20] Yasmin Halawani et al. "RRAM-based CAM combined with time-domain circuits for hyperdimensional computing". In: *Scientific reports* 11.1 (2021), pp. 1–11.

[21] Shuo-An Huang et al. "A 1.9-mW SVM processor with on-chip active learning for epileptic seizure control". In: *IEEE Journal of Solid-State Circuits* 55.2 (2019), pp. 452–464.

[22] Chul Ho Jang et al. "A survey on activities of daily living and occupations of upper extremity amputees". In: *Annals of rehabilitation medicine* 35.6 (2011), pp. 907–921.

[23] Néstor J Jarque-Bou et al. "A calibrated database of kinematics and EMG of the forearm and hand during activities of daily living". In: *Scientific data* 6.1 (2019), pp. 1–11.

[24] Junior C Jesus et al. "Deep deterministic policy gradient for navigation of mobile robots in simulated environments". In: *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE. 2019, pp. 362–367.

[25] Ning Jiang et al. "Effect of arm position on the prediction of kinematics from EMG in amputees". In: *Medical & biological engineering & computing* 51.1 (2013), pp. 143–151.

[26] Benjamin C Johnson et al. "An implantable 700μW 64-channel neuromodulation IC for simultaneous recording and stimulation with rapid artifact recovery". In: *2017 Symposium on VLSI Circuits*. IEEE. 2017, pp. C48–C49.

[27] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, pp. 1–12.

[28] Pentti Kanerva. "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors". In: *Cognitive computation* 1.2 (2009), pp. 139–159.

[29] Geethan Karunaratne et al. "In-memory hyperdimensional computing". In: *Nature Electronics* 3.6 (2020), pp. 327–337.

[30] Geethan Karunaratne et al. "Robust high-dimensional memory-augmented neural networks". In: *Nature communications* 12.1 (2021), pp. 1–12.

[31] Rami N Khushaba et al. "Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals". In: *Expert Systems with Applications* 39.12 (2012), pp. 10731–10738.

[32] Denis Kleyko, Edward Paxon Frady, and Friedrich T Sommer. "Cellular automata can reduce memory requirements of collective-state computing". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[33] Denis Kleyko and Evgeny Osipov. "No two brains are alike: Cloning a hyperdimensional associative memory using cellular automata computations". In: *First International Early Research Career Enhancement School on Biologically Inspired Cognitive Architectures*. Springer. 2017, pp. 91–100.

[34] Denis Kleyko et al. "Density encoding enables resource-efficient randomly connected neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[35] Brent Komer and Chris Eliasmith. "Efficient navigation using a scalable, biologically inspired spatial representation." In: *CogSci*. 2020.

[36] Ronny Krashinsky et al. "SyCHOSys: Compiled energy-performance cycle simulation". In: *Workshop on Complexity-Effective Design, 27th International Symposium on Computer Architecture*. 2000.

[37] Yunsup Lee et al. "A 45nm 1.3 GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators". In: *ESSCIRC 2014-40th European Solid State Circuits Conference (ESSCIRC)*. IEEE. 2014, pp. 199–202.

[38] Yunsup Lee et al. "The Hwacha vector-fetch architecture manual, version 3.8. 1". In: *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-262* (2015).

[39] Jiahao Liu et al. "4.5 BioAIP: A Reconfigurable Biomedical AI Processor with Adaptive Learning for Versatile Intelligent Health Monitoring". In: *2021 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 64. IEEE. 2021, pp. 62–64.

[40] Martin Abadi, Ashish Agarwal, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[41] Ana Matran-Fernandez et al. "SEEDS, simultaneous recordings of high-density EMG and finger joint angles during multiple hand movements". In: *Scientific data* 6.1 (2019), pp. 1–10.

[42] A Menon et al. "On the role of hyperdimensional computing for behavioral prioritization in reactive robot navigation tasks". In: *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2022.

[43] Alisha Menon et al. "A Highly Energy-Efficient Hyperdimensional Computing Processor for Biosignal Classification". In: *IEEE Transactions on Biomedical Circuits and Systems* (2022).

[44] Alisha Menon et al. "A Highly Energy-Efficient Hyperdimensional Computing Processor for Wearable Multi-modal Classification". In: *2021 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. 2021, pp. 1–4.

[45] Alisha Menon et al. "Brain-inspired Multi-level Control of an Assistive Prosthetic Hand through EMG Task Recognition". In: *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. 2022, pp. 1–4.

[46] Alisha Menon et al. "Efficient emotion recognition using hyperdimensional computing with combinatorial channel encoding and cellular automata". In: *arXiv preprint arXiv:2104.02804* (2021).

[47] Florian Mirus et al. "An investigation of vehicle behavior prediction using a vector power representation to encode spatial positions of multiple objects and neural networks". In: *Frontiers in neurorobotics* 13 (2019), p. 84.

[48] Anton Mitrokhin et al. "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception". In: *Science Robotics* 4.30 (2019), eaaw6736.

[49] Anton Mitrokhin et al. "Symbolic representation and learning with hyperdimensional computing". In: *Frontiers in Robotics and AI* 7 (2020), p. 63.

[50] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[51] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[52] Alireza Mohammadi et al. "X-Limb: a soft prosthetic hand with user-friendly interface". In: *International Conference on NeuroRehabilitation*. Springer. 2018, pp. 82–86.

[53] Ali Moin, Andy Zhou, et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition". In: *Nature Electronics* (2020), pp. 1–10.

[54] Ali Moin, Andy Zhou, et al. "An EMG gesture recognition system with flexible high-density sensors and brain-inspired high-dimensional classifier". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2018, pp. 1–5.

[55] Ali Moin et al. "A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition". In: *Nature Electronics* 4.1 (2021), pp. 54–63.

[56] Fabio Montagna et al. "PULP-HD: Accelerating brain-inspired high-dimensional computing on a parallel ultra-low power platform". In: *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE. 2018, pp. 1–6.

[57] Peeer Neubert, Stefan Schubert, and Peter Protzel. "An introduction to hyperdimensional computing for robotics". In: *KI-Künstliche Intelligenz* 33.4 (2019), pp. 319–330.

[58] Peer Neubert, Stefan Schubert, and Peter Protzel. "Learning vector symbolic architectures for reactive robot behaviours". In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016.

[59] Gerard O'Leary et al. "26.2 A Neuromorphic Multiplier-Less Bit-Serial Weight-Memory-Optimized 1024-Tree Brain-State Classifier and Neuromodulation SoC with an 8-Channel Noise-Shaping SAR ADC Array". In: *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE. 2020, pp. 402–404.

[60] Evgeny Osipov, Denis Kleyko, and Alexander Legalov. "Associative synthesis of finite state automata model of a controlled object with hyperdimensional computing". In: *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2017, pp. 3276–3281.

[61] Mehmet Akif Ozdemir et al. "Dataset for multi-channel surface electromyography (sEMG) signals of hand gestures". In: *Data in brief* 41 (2022), p. 107921.

[62] Vasileios C. Pezoulas, Themis P. Exarchos, and Dimitrios I. Fotiadis. "Chapter 2 - Types and sources of medical and other related data". In: *Medical Data Sharing, Harmonization and Analytics*. Ed. by Vasileios C. Pezoulas, Themis P. Exarchos, and Dimitrios I. Fotiadis. Academic Press, 2020, pp. 19–65. ISBN: 978-0-12-816507-2. DOI: https://doi.org/10.1016/B978-0-12-816507-2.00002-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128165072000025.

[63] Jonathan Posner, James A Russell, and Bradley S Peterson. "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology". In: *Development and psychopathology* 17.3 (2005), p. 715.

[64] Abbas Rahimi et al. "Hyperdimensional computing for noninvasive brain-computer interfaces: Blind and one-shot classification of EEG error-related potentials". In: *10th EAI Int. Conf. on Bio-inspired Information and Communications Technologies*. CONF. 2017.

[65] Akhlaquor Rahman and Adel Al-Jumaily. "Design and Development of a Bilateral Therapeutic Hand Device for Stroke Rehabilitation". In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 405.

[66] Muhammad Tallal Saeed et al. "Comprehensive Bond Graph Modeling and Optimal Control of an Anthropomorphic Mechatronic Prosthetic Hand". In: *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2019, pp. 2006–2011. DOI: 10.1109/ICMA.2019.8816325.

[67] Deepali Salwan et al. "Challenges with reinforcement learning in prosthesis". In: *Materials Today: Proceedings* 49 (2022), pp. 3133–3136.

[68] Alessandro Sebastianelli et al. "A Deep Q-Learning based approach applied to the Snake game". In: *2021 29th Mediterranean Conference on Control and Automation (MED)*. IEEE. 2021, pp. 348–353.

[69] Mohammadreza Sharif et al. "Towards End-to-End control of a robot prosthetic hand via reinforcement learning". In: *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*. IEEE. 2020, pp. 641–647.

[70] Mahsa Shoaran et al. "Energy-efficient classification for resource-constrained biomedical applications". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8.4 (2018), pp. 693–707.

[71] Siddharth Siddharth, Tzyy-Ping Jung, and Terrence J Sejnowski. "Utilizing deep learning towards multi-modal bio-sensing and vision-based affective computing". In: *IEEE Transactions on Affective Computing* (2019).

[72] Patrick Slade et al. "Tact: Design and performance of an open-source, affordable, myoelectric prosthetic hand". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 6451–6456.

[73] Jinook Song et al. "7.1 An 11.5 TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC". In: *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE. 2019, pp. 130–132.

[74] Niko Sünderhauf et al. "The limits and potentials of deep learning for robotics". In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 405–420.

[75] Edward Wang et al. "A methodology for reusable physical design". In: *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE. 2020, pp. 243–249.

[76] Sheng-Hui Wang et al. "Entropy-assisted emotion recognition of valence and arousal using XGBoost classifier". In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer. 2018, pp. 249–260.

[77] Panipat Wattanasiri, Pairat Tangpornprasert, and Chanyaphan Virulsri. "Design of multi-grip patterns prosthetic hand with single actuator". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.6 (2018), pp. 1188–1198.

[78] Zhepei Wei et al. "Autonomous agents in Snake game via deep reinforcement learning". In: *2018 IEEE International Conference on Agents (ICA)*. IEEE. 2018, pp. 20–25.

[79] Tony F Wu et al. "Brain-inspired computing exploiting carbon nanotube FETs and resistive RAM: Hyperdimensional computing case study". In: *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE. 2018, pp. 492–494.

[80] Jason Yu, Guy Lemieux, and Christpher Eagleston. "Vector processing as a soft-core CPU accelerator". In: *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*. 2008, pp. 222–232.

[81] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2020, pp. 737–744.

[82] Yang Zhao, Zhongxia Shang, and Yong Lian. "A 13.34 μW event-driven patient-specific ANN cardiac arrhythmia classifier for wearable ECG sensors". In: *IEEE transactions on biomedical circuits and systems* 14.2 (2019), pp. 186–197.

[83] Andy Zhou, Rikky Muller, and Jan Rabaey. "Memory-Efficient, Limb Position-Aware Hand Gesture Recognition using Hyperdimensional Computing". In: *arXiv preprint arXiv:2103.05267* (2021).

[84] Andy Zhou et al. "A wireless and artefact-free 128-channel neuromodulation device for closed-loop stimulation and recording in non-human primates". In: *Nature biomedical engineering* 3.1 (2019), pp. 15–26.

[85] Katie Z Zhuang et al. "Shared human–robot proportional control of a dexterous myoelectric prosthesis". In: *Nature Machine Intelligence* 1.9 (2019), pp. 400–411.

[86] Matthew Zurek et al. "Situational confidence assistance for lifelong shared autonomy". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2783–2789.