

Late Breaking Results:

POSEiDON: Pose Estimation in Dynamic On-device Networks via Hyperdimensional Computing

Colin Dupuis, Emilien Meyer, Abu Kaisar Mohammad Masum, Sercan Aygun
 School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA, USA
 {c00478780, c00439243, c00591145, sercan.aygun}@louisiana.edu

Abstract—On-device pose estimation is widely available through models, yet efficiently classifying the resulting landmarks on resource-constrained mobile platforms remains challenging. We propose POSEiDON, a hyperdimensional computing (HDC)-based pose classification pipeline that combines landmark-symbol and joint-angle encodings with low-discrepancy (Sobol) sequences. On a custom 4-class mobile dataset and a 5-class YOGA benchmark, POSEiDON achieves up to 97.79% and 88.44% accuracy, respectively, using a single training pass, while iterative baselines require 10–200 passes or estimators to reach comparable accuracy. An Artix-7 FPGA prototype and an Android implementation further show that the HDC stage adds only marginal power, latency, and memory overhead, indicating that HDC is a promising primitive for lightweight on-device pose recognition.

I. INTRODUCTION

Computer vision has advanced to the point that it has become widely available, even on mobile platforms [1]. This has enabled the deployment of pose estimation models directly on consumer devices through frameworks like Google’s recent MediaPipe pose estimator [2]. As these models become increasingly lightweight, a new research question has emerged: *How should we interpret this newly available pose data for emerging applications while considering processing efficiency?* Human pose recognition can play an important role in applications ranging from fitness monitoring [3] to early detection of biomedical issues [4]–[6]. To achieve this, researchers have tried a variety of classical machine learning (ML) methods [7]–[9], ranging from Convolutional Neural Networks (CNNs) [10] to simple random forest models [11]. To better leverage the mobile-friendly nature of on-device pose estimators, this paper utilizes an alternative classification approach based on Hyperdimensional Computing (HDC) [12]. While previous HDC-based pose estimation works focus on object-level recognition [13], [14], this work, to the best of our knowledge, is the first to apply HDC directly to human pose classification from pose landmarks. HDC has gained popularity as a lightweight alternative to conventional ML methods, specifically for resource-constrained edge devices. HDC models rely on simple binary vector operations and offer few-shot training with lightweight processing. In this work, we propose a system, **POSEiDON: POSE Estimation in Dynamic On-device Networks**, that integrates pose-landmarks with a novel HDC classifier to distinguish between postures. The goal is to evaluate HDC’s practicality for real-time pose classification on mobile platforms.

II. PROPOSED METHOD

HDC is an emerging computing paradigm that is data-agnostic and embeds different information types into binary hypervectors before encoding and learning [15]. This work exploits multiple features of the pose landmarks in the classifier, replacing more expensive conventional classifiers with HDC using novel encodings. Fig. 1 shows the **POSEiDON** pipeline. Landmarks are first extracted from each frame and mapped to a grid. Joint angles and landmark positions are then encoded into hypervectors using angle-based and low-discrepancy (LD) sequence encodings [16]. These hypervectors are bundled into a single pose representation. A custom dataset is created to validate the performance and is collected by us using 360-degree walk-around videos of a single subject performing the activity. **POSEiDON** operates on real-time video sequences rather than single images. We follow Google’s MediaPipe pose model [2] to extract landmarks (hereafter referred to as *symbols*; Step-①).

POSEiDON, as seen in Fig. 1, has four steps of training. After the symbols, the proposed approach has a gridding phase (Step-①) to locate symbols coarsely-grained rather than pixel-wise. The encoding has two approaches to merge: Angle Encoding (Step-②) and Symbol Encoding (Step-③). Since HDC is neuro-symbolic,

representing symbols as vectors is computationally feasible. In what follows, we discuss these proposed encoding methods.

A. Angle Encoding

Beyond the symbols, we also exploit the geometric relations between landmarks by encoding joint angles. Eight joints (e.g., left shoulder) are selected that create the angles. Each joint angle $\theta \in [0, 180^\circ]$ is mapped to a continuous hypervector by interpolating between two discrete angle bins. We uniformly partition the range into B bins of width $\Delta = 180^\circ/B$ and assign each bin b a random bipolar hypervector. For a given angle θ , we compute its normalized bin position $p = \theta/\Delta$, the lower and upper bin indices $b_\ell = \lfloor p \rfloor$, $b_u = \min(b_\ell + 1, B - 1)$, and the fractional offset $\alpha = p - b_\ell$. The angle hypervector is then obtained by linear interpolation $\mathbf{A} = (1 - \alpha) \mathbf{b}_\ell + \alpha \mathbf{b}_u$. This interpolation can be viewed as a *soft 2-to-1 multiplexer* (MUX) between neighboring angle-bin hypervectors ($\mathbf{b}_\ell, \mathbf{b}_u$), where the mixing weights $(1 - \alpha)$ and α are determined by the normalized position of θ within the bin interval. Bins and α are generated using the random sequence encoding; D -dimensional random numbers are compared with the scalar bin value (see Fig. 1, Step-②-II), the vector gets the specific position 0 or 1 value (-1 and +1 in non-ASIC platforms, like mobile computing) whether b is greater or less than the current random number (R). For each joint J , we also encode the random vectors using a similar approach. However, the joints are symbolic, so the threshold ($th = 0.5$; middle point of the unit interval) is used for comparison of any R samples from the unit interval $[0, 1]$. This guarantees the no-bias orthogonal vectors. Then, both the angle (A) and the corresponding joint type (J) hypervectors are bound together with simple XOR (multiplication), yielding AE : angle encode (Fig. 1, Step-②-III).

B. Symbol Encoding

Each of the 33 symbols (e.g., nose, left eye inner, etc.) is also represented by the combination of vectors: symbol vector and its specific grid position. Symbolic identifier is unique to each keypoint and similar to joint vectors; symbols (S) are also created using the orthogonal approach (Fig. 1, Step-③-I). These encode the semantic meaning of the keypoint, and remain fixed during training and inference. The symbol position is also encoded into a vector. This step uses the LD sequence encoding to keep track of X and Y coordinates of the grid with deterministic encodings. We follow the Sobol LD sequence encoding [16], assigning each Sobol index to specific row–column positions [17] (Step-③-II). After obtaining S , X and Y vectors, in Step-③-III, the landmark and the grid positions are bound together, yielding Symbol Encode: $SE = S \oplus X \oplus Y$.

Now that we have AE and SE in hand, capturing a landmark’s joint angle and neuro-symbolic semantic & positional meaning, we can combine them into a single representation. Although AE and SE encode different types of information, HDC allows us to merge their D -dimensional binary data via bundling, i.e., a population count of 1s (or, equivalently, summing ± 1 on mobile-computing platforms). In Step-④, AE and SE are bundled via accumulation. This implements an incremental learning phase: specific poses, such as stand and sit, independently contribute to the corresponding pose hypervector P_{pose} as we iterate through the dataset. Once training is complete, each pose hypervector P (e.g., P_{stand}) is binarized via the *sign* function, yielding the final learned model. The inference stage in Step-⑤ follows the same encoding process, and classification is performed via a similarity check between test and learned pose hypervectors. In our experiments, this pipeline runs in real time on mobile phone platforms, deploying HDC in a realistic video-processing environment, the first of its kind.

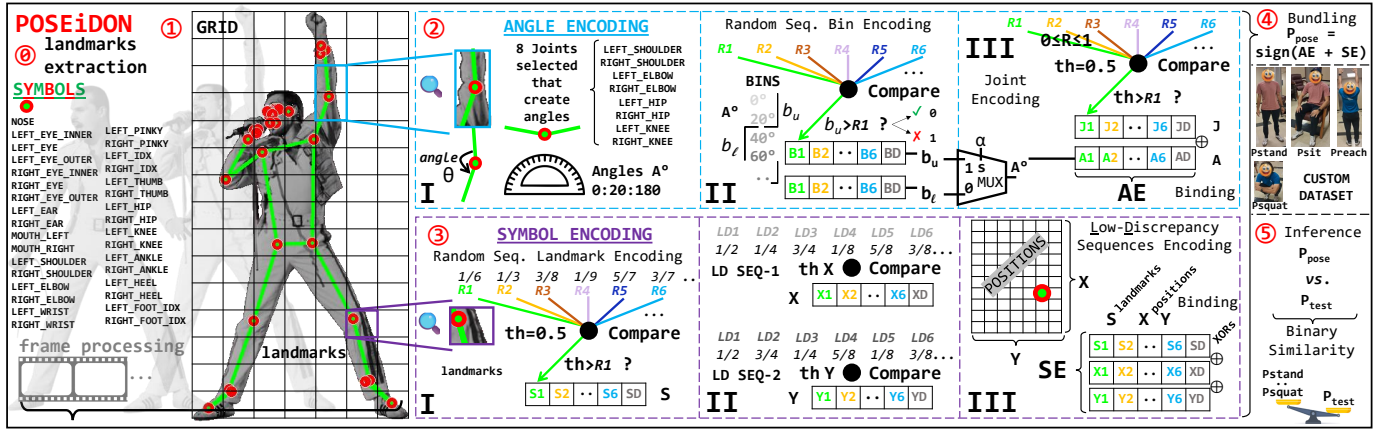


Fig. 1. The **POSEiDON** pipeline showing landmark extraction (grid: 9×16), angle and symbol encoding, and pose classification using HDC.

TABLE I
HARDWARE EFFICIENCY OF HDC POSE ESTIMATOR

D	LP	IOP	SP	TCP	CPD	LUT	FF	TA	ADP
1K	1.00	0.85	131.0	132.85	8.081	394	381	0.0040	0.033
2K	3.00	0.85	131.0	134.85	7.980	604	709	0.0067	0.054
4K	3.00	0.90	131.0	134.90	8.560	1011	1349	0.0120	0.102

LP = Logic Power (mW), IOP = I/O Power (mW), SP = Static Power (mW), TCP = Total On-Chip Power (mW), CPD = Critical Path Delay (ns), LUT = Look-Up Tables, FF = Flip-Flops, TA = Total Area (mm^2), ADP = Area-Delay Product ($\text{mm}^2 \cdot \text{ns}$).

TABLE II
HDC MODEL ACCURACY RESULTS: NO LD SEQUENCE

D	Precision	Recall	F1-Score	Accuracy
1K	83.93	83.82	83.64	83.32
2K	87.11	87.21	87.09	87.21
4K	93.65	93.53	93.50	93.53

III. EXPERIMENTAL RESULTS

A. Hardware Synthesis

We first test the sole hardware component of the HDC classifier to assess its efficiency. The architecture of **POSEiDON**, including the Steps ①-④ in Fig. 1 is implemented in the FPGA via Vivado using the Xilinx Artix-7 AC701 FPGA. Table I reports the results. Logic power increases slightly as more LUTs and FFs are required to process longer hypervectors ($D=1K\text{-bit} \rightarrow 4K\text{-bit}$) for better precision. Still, the total on-chip power remains dominated by static power, which is around 131 mW across all tested sizes. In contrast, area metrics such as LUT count, FF count, total area, and area-delay product (ADP) increase with dimension, reflecting the larger sequential and combinational structures required for higher D dimensions. The critical path delay (CPD) fluctuates slightly, but remains within a small range. All these results ensure computation-friendly usage and are suitable for battery-powered devices.

B. Accuracy

After confirming the hardware efficiency of **POSEiDON**, we evaluate its accuracy. Two experiment groups are tested first to show the impact of deterministic LD encoding: one using pseudo-random grid position encodings and one using the Sobol sequence encodings. As shown in Table II, random encoding reaches 93.53% accuracy only at the size of 4K. With Sobol encoding reported in Table III, the model achieves notably higher accuracy up to 97.79% with 4K dimensions. These results underscore that deterministic encoding for X - Y positions achieves strong performance even at lower vector dimensions. We further benchmark **POSEiDON** against conventional iterative ML models on both our 4-class mobile dataset and the 5-class YOGA benchmark [18]. Table IV reports a budget comparison between **POSEiDON** and four baselines: stochastic gradient-descent logistic regression (SGD-LR), a feedforward neural network (FNN), gradient-boosted decision trees (GBT), and a CNN. Here, the total budget denotes the number of full passes over the training set (epochs for neural networks and SGD-LR, and boosting iterations/trees for GBT). **POSEiDON** attains 97.79% and 88.44% accuracy on the two datasets, respectively, with a single training pass, whereas the baselines require between 10 and 200 passes or estimators to approach similar accuracy.

TABLE III
HDC MODEL ACCURACY RESULTS: WITH LD SEQUENCE

D	Precision	Recall	F1-Score	Accuracy
1K	93.18	93.09	93.07	93.09
2K	97.36	97.35	97.35	97.35
4K	97.81	97.89	97.79	97.79

TABLE IV
BUDGET COMPARISON OF **POSEiDON** AND ML BASELINES [3], [19], [20]

Method	Our Dataset		YOGA-5 [18]	
	Total Budget	Acc.	Total Budget	Acc.
SGD-LR	200 Passes	90.63	200 Passes	83.77
FNN	10 Passes	96.93	10 Passes	85.53
GBT	20 Estimators	97.77	20 Estimators	88.60
CNN	10 Passes	97.36	50 Passes	82.89
POSEiDON	1 Pass	97.79	1 Pass	88.44

C. Mobile Platform Computing Performance

In addition to the FPGA prototype of the stand-alone HDC architecture, this work also targets a real-life mobile deployment on an Android phone. We developed a testing application (APP) with two modes: a live mode that displays keypoints on the camera feed, and a benchmark mode that measures mobile hardware performance. Our evaluation uses the benchmark mode to test different hypervector dimensions. The benchmark processes every 5th frame of a test video, extracts keypoints using the MediaPipe pose model [2], and classifies each pose with the HDC model. The test runs until 1000 classifications are completed. During this time, the APP records battery drop, inference time, total processing time, and peak heap memory. Table V summarizes the breakdown for the landmark (MediaPipe) stage and the HDC pose classifier. Across all D , the additional HDC time, as well as the battery drop, total power, and peak heap memory, remain very close to the landmark-only budget, confirming that the overall **POSEiDON** pipeline adds comparably minimal computational overhead on the mobile platform.

TABLE V
MOBILE COMPUTING BUDGET BREAKDOWN

Breakdown	†	⊖	♣	★	☆	♣	⚡
$D=1K$ HDC	0.45	190.95	13.95	2.00	100.00	2252.73	151.13
$D=2K$ HDC	0.63	207.00	27.90	2.00	100.00	2285.49	159.92
$D=4K$ HDC	0.79	211.07	55.89	3.00	150.00	3347.25	163.98
Landmark	N/A	180.24	N/A	2.00	100.00	2516.44	152.66

Device: Samsung Galaxy A16, Android 14, 2.4 GHz, 128 GB, † Avg. HDC Time (s), ⊖ Avg. Total Inference Time (s), ♣ Model Size (kB), ★ Battery Drop (%), ☆ Current Hours (mAh), ♣ Total Power (mW), ⚡ Peak Heap Mem. (MB)

IV. CONCLUSION

POSEiDON is a lightweight HDC-based pose estimator that combines landmark symbols and joint-angle encoding to achieve accurate, low-cost on-device computation. Future work will scale **POSEiDON** to diverse pose datasets and explore using HDC as a drop-in classifier for SoTA deep learning models.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation grant No. 2019511, NASA grant 80NSSC25C0335, Vernon & Ruby Langlinais Non-Endowed Res. Fund, Lockheed Martin Corp. Endowed Professorship, award from NASA, and generous gifts from NVIDIA.

REFERENCES

- [1] T. Zhao, Y. Xie, Y. Wang, J. Cheng, X. Guo, B. Hu, and Y. Chen, "A survey of deep learning on mobile devices: Applications, optimizations, challenges, and research opportunities," *Proceedings of the IEEE*, vol. 110, no. 3, pp. 334–354, 2022.
- [2] Google AI Edge MediaPipe Team, "Pose landmark detection guide," https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker, 2025, last accessed: 30 November 2025.
- [3] M. Balaeva and A. Tikhomirov, "Yoga pose classification using machine learning," *International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, vol. 7th, pp. 01–05, 2025.
- [4] A. Raza, A. M. Qadri, I. Akhtar, N. A. Samee, and M. Alabdulhafith, "Logrf: An approach to human pose estimation using skeleton landmarks for physiotherapy fitness exercise correction," *IEEE Access*, vol. 11, pp. 107 930–107 939, 2023.
- [5] S. S. M. Salehi, S. Khan, D. Erdogmus, and A. Gholipour, "Real-time deep pose estimation with geodesic loss for image-to-template rigid registration," *IEEE Transactions on Medical Imaging*, vol. 38, no. 2, pp. 470–481, 2018.
- [6] A. Avogaro, F. Cunico, B. Rosenhahn, and F. Setti, "Markerless human pose estimation for biomedical applications: a survey," *Frontiers in Computer Science*, vol. 5, p. 1153160, 2023.
- [7] A. Gupta, K. Gupta, K. N. M. Gupta, and K. O. Gupta, "Human activity recognition using pose estimation and machine learning algorithm," in *International Symposium on Intelligent Control*, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:232401456>
- [8] C. Hong, J. Yu, J. Zhang, X. Jin, and K.-H. Lee, "Multimodal face-pose estimation with multitask manifold deep learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3952–3961, 2018.
- [9] G. Lan, Y. Wu, F. Hu, and Q. Hao, "Vision-based human pose estimation via deep learning: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 53, no. 1, pp. 253–268, 2022.
- [10] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.
- [11] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.
- [12] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, Jun. 2009. [Online]. Available: <https://doi.org/10.1007/s12559-009-9009-8>
- [13] P. Neubert and S. Schubert, "Hyperdimensional computing as a framework for systematic aggregation of image descriptors," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 933–16 942.
- [14] N. Anwar, C. Parameshwara, C. Fermüller, and Y. Aloimonos, "Towards an improved hyperdimensional classifier for event-based data," in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, 2023, pp. 1–4.
- [15] D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi, "A survey on hyperdimensional computing aka vector symbolic architectures, part i: Models and data transformations," *ACM Comput. Surv.*, vol. 55, no. 6, Dec. 2022. [Online]. Available: <https://doi.org/10.1145/3538531>
- [16] S. Liu and J. Han, "Energy efficient stochastic computing with sobol sequences," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 650–653.
- [17] S. Aygun and M. Hassan Najafi, "Sobol sequence optimization for hardware-efficient vector symbolic architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 3, pp. 937–950, 2025.
- [18] N. Pandit, "Yoga poses dataset," <https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset>, 2020, accessed: 2025-11-30.
- [19] M. Pavikars M and R. Jansi, "Yoga pose classification using cnn with prelu activation," *2024 International Conference on Advancements in Power, Communication and Intelligent Systems (APCI), KANNUR, India*, pp. 1–5, 2024.
- [20] T. Sabaichai, D. Tancharoen, and P. Limpasuthum, "Human action classification based on pose estimation and artificial neural network," in *2023 7th International Conference on Information Technology (InCIT)*, 2023, pp. 181–185.