

Accelerated Digital Twin Learning for Edge AI: A Comparison of FPGA and Mobile GPU

Bin Xu, Ayan Banerjee, Midhat Urooj, Sandeep K.S. Gupta
Impact Lab, School of Computing and Augmented Intelligence
Arizona State University, Tempe, AZ, USA
 {binxu4,abanerj3,murooj,sandeep.gupta}@asu.edu

Abstract—Digital twins (DTs) can enable precision healthcare by continually learning a mathematical representation of patient-specific dynamics. However, mission critical healthcare applications require fast, resource-efficient DT learning, which is often infeasible with existing model recovery (MR) techniques due to their reliance on iterative solvers and high compute/memory demands. In this paper, we present a general DT learning framework that is amenable to acceleration on reconfigurable hardware such as FPGAs, enabling substantial speedup and energy efficiency. We compare our FPGA-based implementation with a multi-processing implementation in mobile GPU, which is a popular choice for AI in edge devices. Further, we compare both edge AI implementations with cloud GPU baseline. Specifically, our FPGA implementation achieves an 8.8× improvement in performance-per-watt for the MR task, a 28.5× reduction in DRAM footprint, and a 1.67× runtime speedup compared to cloud GPU baselines. On the other hand, mobile GPU achieves 2x better performance per watts but has 2x increase in runtime and 10x more DRAM footprint than FPGA. We show the usage of this technique in DT guided synthetic data generation for Type 1 Diabetes and proactive coronary artery disease detection.

Index Terms—digital twin, hardware acceleration, precision healthcare, synthetic data generation

I. INTRODUCTION

A key technological innovation towards physical AI [1] is the concept of digital twin (DT). DTs are mathematical models of physical processes with two essential properties: a) the model structure is guided by first-principle satisfied by the physical process, and b) the model parameters are continuously calibrated with real world data in real-time. A major application of DT is in precision medicine, which brings a fundamental shift in disease management from decision making based on statistical inferences of individual variance of treatment efficacy to patient specific evaluations leading to just-in-time diagnosis, personalized treatment, and individualized recovery as shown in Figure 1.

A continuously calibrated DT can be used for simulating various potential treatment plans for their safety and efficacy on the specific patient [2]–[6], derive a personalized verified safe and effective plan [7], [8] or identify novel operational scenarios [9]. Given the mission critical nature of the application of precision medicine, the calibration, simulation and safety / efficacy feedback has to be performed within time constraints. These constraints are application specific and guided by hazard evolution dynamics [10] as shown in Table I.

A. Computational Challenges of DT Learning

Automated continuous learning of DT in real-time is a major scientific challenge in the age of physical AI.

Real-Time Challenge: The primary computational component of DT learning is physics-guided model recovery [11], where the model coefficients of a first-principle based differential dynamics is learned from real data under constraints of sampling, implicit or unmonitored dynamics, and human errors. The computational needs of physics-guided model recovery (MR) may prevent real-time operation even with parallelization with state-of-the-art (SOTA) multi-processing pipeline. Table I shows that the time to learn an application specific DT

exceeds the response time required to avoid medical hazards. One of the fundamental reason is that analytical operations with physics-guided models require solution of differential dynamics which are iterative in nature. Such iterative operations are not amenable for parallelization. As such the SOTA multi-processing pipeline is less effective in real-time DT learning.

TABLE I: Use of digital twins to provide personalized feedback in precision healthcare applications. The table summarizes application-specific response time requirements and digital twin (DT) learning times on a SOTA GPU (NVIDIA RTX 6000).

Domain	Hazard	Feedback	Response Time	DT model	GPU Time	Data transfer time
Diabetes [11]	Hypo-glycemia	Change insulin rate	900s	Bergman minimal metabolic model	1412s	27s
Cardiac disease [12], [13]	Ischemia	Alert first responders	re-100s	ECG generative model	452s	81s
Brain sensing [11], [14]	Attention deficit	Audio-visual cues	33ms	Resistance capacitance model	321s	125s

Edge AI [15] Challenge: Data driven inferencing in real-time suffers from data transfer bottleneck (Table I shows data transfer times in medical DT applications forms a significant percentage or even exceeds response time). Recent advancements in edge AI aim to bring DT learning computation closer to data source, potentially bypassing the data transfer time. However, edge AI devices such as mobile GPU are resource constrained and hence may not be capable of DT learning within real-time constraints.

Real World Challenge: Data obtained from real world deployments of healthcare systems are restricted in sampling rate, and often compromised signal quality with potentially poor signal-to-noise ratio (SNR) especially when collected from human participants in free living conditions [16], [17]. Moreover, privacy constraints may lead to unavailability of measurements of key dynamical parameters of the DTs. Hence, any DT learning mechanism in the real world require to learn implicit or unmeasured dynamics. Recently physics-guided sparse model recovery techniques such as Physics Informed Neural Networks (PINNs) [18] or Physics informed Neural ODE (PiNODE) [19] or Extracting sparse Model from ImpLicit dYnamics (EMILY) [16] have been proposed to tackle implicit dynamics under low sampling frequencies. These techniques calibrate DT with real world data by following the Koopman theory [20]. The techniques attempt to learn a Koopman operator [20] that models the first-principle based DT dynamics using an expanded sparse state space where the dynamics become linear. The techniques utilize the universal function modeling capability of neural networks to learn the implicit dynamics while maintaining robustness to sensor noise. However, apart from the significant computational requirements of solving an Ordinary Differential Equation (ODE) in each learning step, these techniques also suffer from high memory requirements to store the expanded state space during computation. Hence,

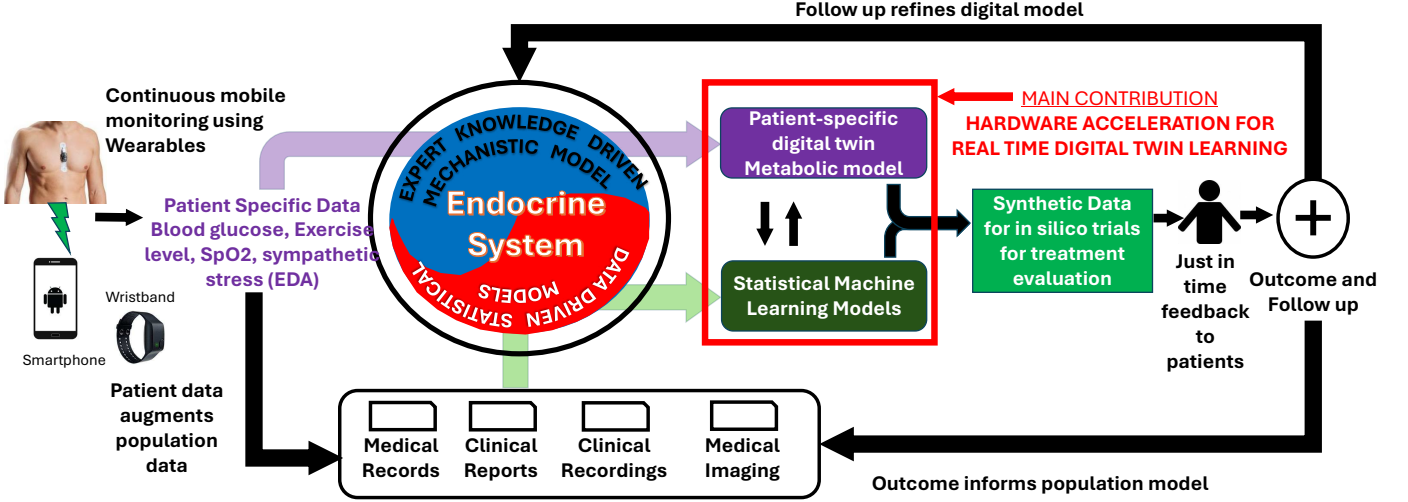


Fig. 1: Real-time digital twin learning for precision healthcare applications such as Diabetes Care.

although these techniques are capable to calibrate DTs with real world data, they may not meet the real-time requirements and resource constraints of edge devices to support edge AI applications.

B. Contribution of manuscript

The hardware acceleration of physics-guided model recovery techniques remains a relatively underexplored research area, particularly in terms of evaluating their feasibility for meeting the timing and resource constraints of edge AI applications. In this paper, we demonstrate a pathway towards hardware acceleration of physics-guided model recovery such as PINNs, PiNODE, and EMILY to enable real-time DT calibration for precision healthcare. We show the application of hardware acceleration of DT learning for two exemplary medical applications on insulin management for Type 1 diabetes and coronary artery disease detection using electrocardiogram (ECG) sensors and show a feasibility analysis of hardware acceleration in meeting timing and resource constraints of edge AI applications.

II. PHYSICS-GUIDED MODEL RECOVERY

The primary objective of MR is similar to an auto-encoder (Figure 2), where given a multivariate time series signal $X(t)$, the aim is to find a latent space representation that can be used to reconstruct an estimation $\hat{X}(t)$ with low error. It has the traditional encoder $\phi(t)$ and decoder $\Psi(t)$ of an auto-encoder architecture. MR represents the measurements X of dimension n and N samples as a set of nonlinear ordinary differential equation model in Eqn. 1.

$$\dot{X} = h(X, U, \theta), \quad (1)$$

where h is a parameterized nonlinear function, U is the m dimensional external input, and θ is the p dimensional coefficient set of the nonlinear ODE model.

Sparsity: An n -dimensional model with M^{th} order nonlinearity can utilize $\binom{M+n}{n}$ nonlinear terms. A sparse model only includes a few nonlinear terms $p \ll \binom{M+n}{n}$. Sparsity structure of a model is the set of nonlinear terms used by it.

Identifiable model: A model in Eqn. 1 is identifiable [22], if \exists time $t_I > 0$, such that $\forall \theta, \tilde{\theta} \in \mathcal{R}^p$:

$$\forall t \in [0, t_I], f(X(t), U(t), \theta) = f(X(t), U(t), \tilde{\theta}) \implies \theta = \tilde{\theta}. \quad (2)$$

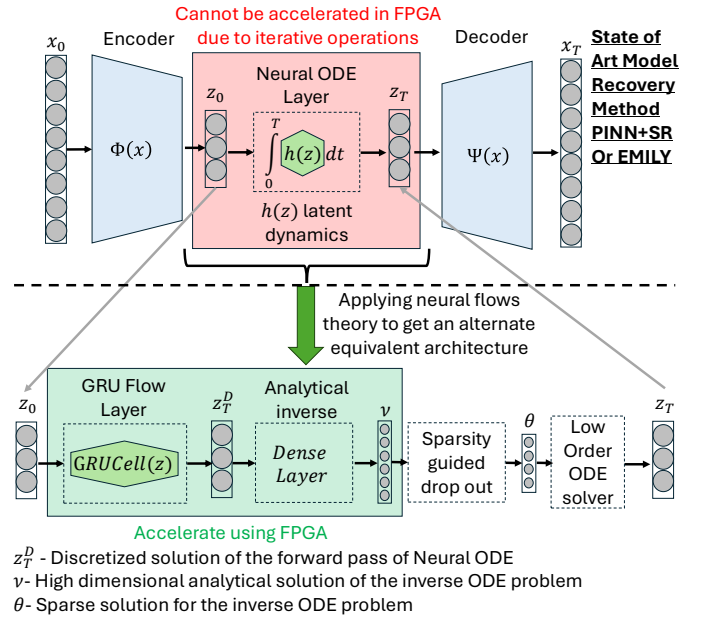


Fig. 2: FPGA acceleration strategy [21] using neural flow-based equivalent architecture to neural ODEs.

Eqn. 2 effectively means that a model is identifiable if two different model coefficients do not result in identical measurements X . In simpler terms, this means $\forall \theta_i \in \theta, \frac{dX}{d\theta_i} \neq 0$. In this paper, we assume that the underlying model is identifiable.

Problem 1 (Sparse Model Recovery). Given N samples of measurements X and inputs U , obtained from a sparse model in Eqn. 1 such that θ is identifiable, recover $\hat{\theta}$ such that for \hat{X} generated from $f(X, U, \hat{\theta})$, we have $\|X - \hat{X}\| \leq \epsilon$, where ϵ is the maximum tolerable error.

Role of NODE: Both EMILY [16] and PINN [18] utilize a layer of NODE cells in order to integrate the underlying nonlinear ODE dynamics. NODE cell's forward pass is by design the integration of the function h over time horizon T

with N samples (Fig. 2). This effectively requires an ODE solver in each cell of the NODE layer:

$$z(t) = \int_0^T h(z, u, \theta) dt, \quad (3)$$

where $z \in Z$ and $u \in U$ are each cells output and input. The results are then used further in the EMILY or PINN pipeline to extract the accurate underlying nonlinear ODE model.

III. HARDWARE ACCELERATION OF PHYSICS-GUIDED MR

A primary challenge in accelerating MR techniques lies in the iterative nature of ODE solvers, which are required for solving NODE cell operations during the forward pass. Recent works have explored the acceleration of standalone ODE solvers [23], [24], but these solutions assume fixed ODE model coefficients. Such fixed-coefficient approaches are not suitable for PiNODE, which requires solving a large number of ODEs with dynamically varying model coefficients, making traditional acceleration methods ineffective for generalizable MR architectures. We leverage the theory of neural flows [25] to develop an alternative neural structure that is mathematically equivalent to the NODE layers used in EMILY, PiNODE, and PINN while being more amenable to FPGA acceleration (Figure 2). Instead of using a conventional NODE layer, we apply a layer of invertible functions designed through a combination of Gated Recurrent Units (GRUs) and a dense layer of neurons with nonlinear activation functions. GRU [26] are a type of recurrent neural network (RNN) architecture that introduces gating mechanisms to control the flow of information over time. Compared to traditional RNNs or LSTMs, GRUs are computationally efficient and require fewer parameters [27], making them favorable for deployment on resource-constrained platforms such as FPGAs.

FPGA (Field-Programmable Gate Array) is a reconfigurable semiconductor device that enables developers to implement custom digital circuits directly in hardware [28]. Unlike fixed-function processors, FPGAs consist of an array of Configurable Logic Blocks (CLBs), Look-Up Tables (LUTs) for implementing combinational logic, Flip-Flops (FF) for sequential logic, and programmable interconnects [29]. FPGAs also incorporate on-chip memory resources, such as Block RAM (BRAM) and UltraRAM (URAM), as well as Digital Signal Processing (DSP) slices optimized for arithmetic-intensive operations.

One of the primary challenges in FPGA design lies in efficiently mapping high-level algorithms onto limited hardware resources while maximizing performance. Loop-carried dependencies—such as Read-After-Write (RAW), Write-After-Read (WAR)—can inhibit effective pipelining, and reduce throughput. In addition to control hazards, memory access patterns pose a significant design challenge. FPGAs feature a hierarchical memory system including block RAM (BRAM), Look-Up Tables (LUTs), and registers (FF), all of which must be judiciously partitioned and scheduled to avoid access bottlenecks and ensure data locality.

In our design, we address these challenges through two key techniques: *array partitioning* and *loop pipelining*, both guided by high-level synthesis (HLS) directives. The FPGA kernel interfaces with the processor using an AXI4-Lite protocol, after which input data is transferred to on-chip memory. We apply full array partitioning using the directive `#pragma HLS ARRAY_PARTITION complete`, which instructs the HLS compiler to map each element of the input array to an independent storage resource—such as a dedicated register or BRAM segment. This partitioning strategy eliminates inter-element memory conflicts and enables parallel access to the data elements.

We then construct a fully parallelized model recovery pipeline on the FPGA. All major computational

stages—including the forward pass, backpropagation, and loss computation—are pipelined using `#pragma HLS PIPELINE II=1`. Once the inputs are partitioned and loop-carried dependencies are removed, this setup achieves an initiation interval (II) of 1, allowing a new iteration to begin every clock cycle. This significantly boosts throughput and latency performance. There can be the violation of loop dependency in the simulation. In order to eliminate RAW and WAR hazards, we need to test `#pragma HLS PIPELINE II=2` or `#pragma HLS PIPELINE II=3`, which means a new iteration begins in every 2 cycles and 3 cycles. If there is no time violation in the simulation, it means there are no RAW and WAR hazards. However, more cycles mean more latency in the pipeline of computation.

IV. EDGE AI FEASIBILITY EVALUATION

We explore the feasibility of real-time DT calibration using real-world data in the edge for two exemplary applications of insulin management and electrocardiogram monitoring for coronary artery disease detection.

A. Applications

Automated insulin delivery (AID): For the insulin management system the digital twin took the form of Eqn: 4 - 6

$$\dot{i}(t) = -n\delta i(t) + p_4 u_1(t) \quad (4)$$

$$\dot{i}_s(t) = -p_1 \delta i_s(t) + p_2 (\delta i(t) - i_b) \quad (5)$$

$$\dot{G}(t) = -\delta i_s(t) G_b - p_3 (\delta G(t)) + u_2(t)/V_o I, \quad (6)$$

The input vector $U(t)$ consists of the overnight basal insulin level i_b and the glucose appearance rate in the body u_2 . The output vector $Y(t)$ comprises the blood insulin level i , the interstitial insulin level i_s , and the blood glucose level G . In AP, only the blood glucose level G is a measurable output. i_s and i are hidden states that are not measurable but contribute to the final glucose output. p_1, p_2, p_3, p_4, n , and $1/V_o I$ are all patient specific coefficients.

The DT was calibrated using the real-world OhioT1D dataset available in [30]. It is 14 time series data of glucose insulin dynamics. Each time series data had a duration of 16 hrs 40 mins which amounts to 200 samples of Continuous Glucose Monitor (CGM) and insulin data.

Cardiac Digital Twin: The cardiac digital twin is based on the ECGSYN model [31], the state variables (x, y, z) evolve as -

$$\dot{x} = (1 - \sqrt{x^2 + y^2}) x - \omega(t) y, \quad (7)$$

$$\dot{y} = (1 - \sqrt{x^2 + y^2}) y + \omega(t) x, \quad (8)$$

$$\dot{z} = -\sum_{i \in \{P, Q, R, S, T\}} a_i \Delta \theta_i \exp\left(-\frac{\Delta \theta_i^2}{2 b_i^2}\right) - (z - z_0(t)), \quad (9)$$

where $\Delta \theta_i = [\theta(t) - \theta_i] \bmod 2\pi$, $\theta(t) = \text{atan2}(y, x)$, $\omega(t) = \frac{2\pi}{r(t)}$, $z_0(t) = A_b \sin(2\pi f_{\text{resp}} t)$.

Variables are defined as: $[x, y]$ coordinates on the unit circle governing phase dynamics, $[\theta(t)]$ instantaneous phase $2(y, x)$, $[\omega(t)]$ angular velocity set by the instantaneous RR interval $r(t)$, $[z(t)]$ output signal, whose peaks form the ECG waveform, $[\theta_i]$ angles of the P, Q, R, S, T peaks on the circle, $[a_i]$ amplitude of the i th peak, $[b_i]$ width (standard deviation) of the i th peak, $[r(t)]$ time-varying RR interval with prescribed power spectrum, $[A_b, f_{\text{resp}}]$ baseline-wander amplitude and respiratory frequency.

B. Platforms Used

To evaluate the performance of our FPGA-based accelerator, we conducted a series of experiments across three hardware platforms: a cloud GPU, an edge-based mobile GPU, and a resource-constrained FPGA. The cloud GPU serves as the baseline for comparison, with a focus on evaluating power

efficiency, execution time, and inference accuracy across all platforms.

GPU Platform: Experiments were first conducted on a workstation equipped with an Intel Xeon w9-3475X CPU and an NVIDIA RTX 6000 GPU with 48 GB of memory. Models were implemented using TensorFlow 2.10 and Keras 2.10. Power consumption was monitored using `nvidia-smi`, while execution time and DRAM footprint were recorded using the `time` and `psutil` libraries.

Mobile GPU Platform: To assess edge-level performance, we deployed models on the NVIDIA Jetson Orin Nano Developer Kit. This platform features a 6-core Arm Cortex-A78AE CPU and 8 GB of LPDDR5 memory. Its integrated GPU, based on the NVIDIA Ampere architecture, includes 1024 CUDA cores and 32 Tensor Cores. Power consumption was measured using `tegrastats`.

FPGA Platform: For FPGA implementation, experiments were performed on the PYNQ-Z2 board, which includes a dual-core ARM Cortex-A9 processor and a 1.3M-configurable-gate FPGA. The GRU model was developed from scratch, with both forward pass and backpropagation logic implemented in C++ using High-Level Synthesis (HLS) in AMD’s Vitis toolchain. The forward-pass accelerator was integrated using Direct Memory Access (DMA) to interface with the processing system. Power consumption was evaluated through Vivado’s power analysis, while runtime and DRAM usage were recorded using the `time` and `psutil` libraries. All `#pragma` directives and hardware-specific constructs used are compliant with Vitis High-Level Synthesis (Vitis HLS) [32], and the design was compiled using the Vitis HLS compiler.

C. Results

As shown in Table II and Table III, the FPGA implementation achieves substantial efficiency gains compared to the GPU baseline. Specifically, it offers an 8.8 \times improvement in performance-per-watt for the MR task and achieves over 28.5 \times reduction in DRAM footprint. Additionally, the FPGA provides a 1.67 \times speedup in runtime for MR, despite operating at significantly lower frequencies. These results align with the comprehensive work by Cong et al. [33], which compare the performance of FPGAs and GPUs across a variety of application domains.

Figure 3 illustrates the roofline model [34] comparison among FPGA, mobile GPU, and cloud GPU platforms for the MR task. The FPGA demonstrates efficiency at low operational intensities (around 0.5 FLOPs/Byte), operating near its memory bandwidth limit. This aligns well with the demands of real-time, low-batch-size workloads typical in model recovery applications. Although the FPGA has a lower compute ceiling (1 GFLOPS) compared to the cloud GPU (10 GFLOPS), it delivers over 8 \times better performance-per-watt and achieves 1.6 \times faster runtime under the same task. Cloud GPUs, while powerful, are optimized for high operational intensities and suffer inefficiencies when handling memory-bound edge workloads. Mobile GPUs strike a balance between the two but still require significantly higher DRAM footprint than FPGAs to reach similar runtime performance.

From the Figure 3, we find that applications that benefit most from FPGA implementation under these constraints are those that are memory-bound, latency-sensitive, and have modest compute requirements. Examples include real-time physiological signal processing, streaming sensor fusion, and lightweight edge inference where throughput per watt and deterministic latency are more critical than peak FLOPs.

We derived the Pareto front from experimental results corresponding to the optimal hyperparameter settings of the hardware acceleration strategy. Figure 4 illustrates the Pareto front spanning Machine Learning (ML), Physics-Guided Machine

Learning (PG) [35], which integrates physical laws or domain knowledge into data-driven models, and Model Recovery (MR) tasks across FPGA, Mobile GPU (MGPU), and GPU platforms. The trend shows a clear separation between edge AI and cloud AI and shows the feasibility of an FPGA-based solution to achieve high speed and lower energy consumption with a modest DRAM footprint, making them ideal for DT learning in the edge. In contrast, GPU-based solutions require higher power but offer greater memory bandwidth, making them more suitable for compute-intensive workloads. The performance of MGPU falls between that of the FPGA and cloud GPU.

TABLE II: Performance comparison among FPGA, Mobile GPU, and GPU for **AID**.

Metric	FPGA	Mobile GPU	GPU
Average Error	4.60	3.07	2.90
Runtime (s)	253.84	562.75	423.28
Avg Power (W)	4.905	5.532	72.00
DRAM Footprint (MB)	214.23	2355.13	6118.36
Freq (MHz)	173	306	1410
Perf/Watt (s/W)	51.76	101.74	5.88

TABLE III: Performance comparison among FPGA, Mobile GPU, and GPU for **Cardiac**.

Metric	FPGA	Mobile GPU	GPU
Average Error	7.20	8.07	6.10
Runtime (s)	25.6	115.2	103.4
DRAM Footprint (MB)	121.3	1061.3	3101.6
Perf/Watt (s/W)	5.22	20.83	1.44

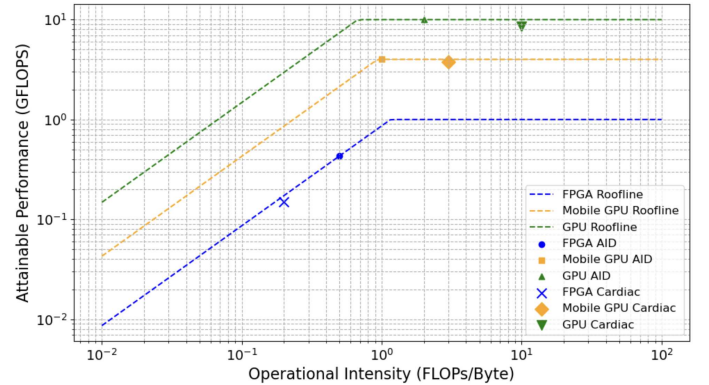


Fig. 3: Roofline model across FPGA, mobile GPU, and cloud GPU platforms, plotted on log-log scale.

V. DISCUSSION AND FUTURE DIRECTIONS

At its core, our work embodies the philosophy that algorithm and hardware must co-evolve: by reformulating physics-guided ODE solvers into neural-flow-inspired GRU+Dense blocks, we unlock a class of learning models whose data dependencies map naturally onto deep pipelines and fine-grained parallelism. This co-design approach transforms inherently iterative recovery methods into streamable computations, closing the gap between mathematical expressivity and real-time edge deployment. Beyond raw performance metrics, it reframes digital twins as living, on-device entities—no longer tethered to cloud resources but capable of continuous, patient-specific adaptation in situ.

Looking forward, this hardware-acceleration paradigm will extend in two key directions. First, heterogeneous integration of mixed-signal and analog compute elements promises even greater energy efficiency for sparse dynamics, enabling

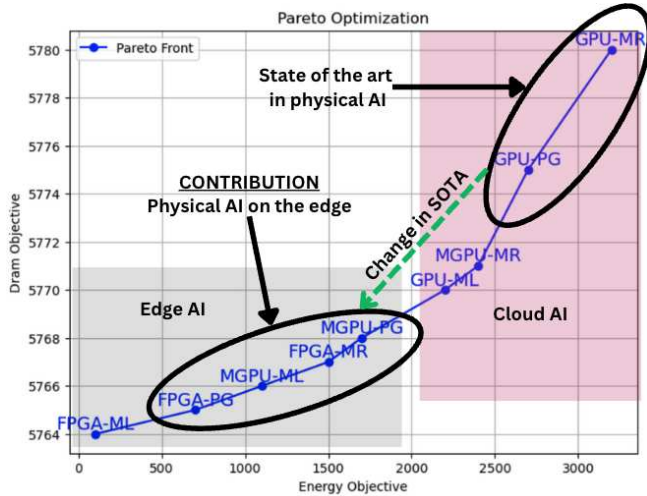


Fig. 4: Each blue dot represents a Pareto-optimal solution spanning Machine Learning (ML), Physics-Guided Machine Learning (PG), and Model Recovery (MR) tasks across FPGA, Mobile GPU (MGPU), and GPU platforms. State-of-the-art physical AI typically relies on cloud GPUs, represented in the red region. This work enables physical AI on edge platforms—illustrated in the grey region—achieving substantial reductions in both DRAM footprint and energy consumption.

miniaturized accelerators in wearables and implantables. Second, automated hardware synthesis from high-level model descriptions—leveraging domain-specific languages and compiler frameworks—will democratize custom accelerator design, so that new physiological models can be instantly mapped to optimized circuits. Together, these advances will drive a new generation of truly autonomous, safety-critical digital twins that learn, infer, and adapt at the edge.

ACKNOWLEDGMENTS

The work is partially funded by NSF FDT-Biotech grant 2436801.

REFERENCES

- [1] J. Huang, “GTC Keynote with NVIDIA CEO Jensen Huang,” 2025.
- [2] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta, “Ensuring safety, security, and sustainability of mission-critical cyber-physical systems,” *Proceedings of the IEEE*, vol. 100, no. 1, pp. 283–299, 2011.
- [3] A. Banerjee and S. K. S. Gupta, “Your mobility can be injurious to your health: Analyzing pervasive health monitoring systems under dynamic context changes,” in *2012 IEEE International Conference on Pervasive Computing and Communications*, pp. 39–47, 2012.
- [4] A. Banerjee and S. K. Gupta, “Analysis of smart mobile applications for healthcare under dynamic context changes,” *IEEE Transactions on Mobile Computing*, vol. 14, no. 5, pp. 904–919, 2015.
- [5] K. Sadeghi, A. Banerjee, and S. K. Gupta, “A system-driven taxonomy of attacks and defenses in adversarial machine learning,” *IEEE transactions on emerging topics in computational intelligence*, vol. 4, no. 4, pp. 450–467, 2020.
- [6] I. Lamrani, A. Banerjee, and S. K. S. Gupta, “Operational data-driven feedback for safety evaluation of agent-based cyber-physical systems,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3367–3378, 2021.
- [7] A. Banerjee, I. Lamrani, and S. K. S. Gupta, “Synthesizing operationally safe controllers for human-in-the-loop human-in-the-plant hybrid loop systems,” in *Pattern Recognition* (A. Antonacopoulos, S. Chaudhuri, R. Chellappa, C.-L. Liu, S. Bhattacharya, and U. Pal, eds.), (Cham), pp. 17–35, Springer Nature Switzerland, 2025.
- [8] A. Banerjee, A. Maity, I. Lamrani, and S. K. Gupta, “Towards certified safe personalization in learning enabled human-in-the-loop human-in-the-plant systems,” *ACM Journal on Emerging Technologies in Computing Systems*, 2025.

- [9] A. Maity, A. Banerjee, and S. K. Gupta, “Detection of unknown-unknowns in human-in-loop human-in-plant safety critical systems,” *IEEE Transactions on Artificial Intelligence*, pp. 1–16, 2025.
- [10] N. Leveson, M. Daouk, N. Dulac, and K. Marais, “Applying STAMP in accident analysis,” in *NASA Conference Publication*, pp. 177–198, NASA, 1998, 2003.
- [11] A. Banerjee and S. K. Gupta, “Recovering implicit physics model under real-world constraints,” in *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2024)*, vol. 392 of *Frontiers in Artificial Intelligence and Applications*, (Hague, Netherlands), pp. 737–744, IOS Press, 2024.
- [12] R. Salian, A. Banerjee, P. Kamboj, S. Gupta, A. K. Sriramou, and K. Srivathsan, “Expert-guided transformers: Advancing cad diagnosis,” in *2024 58th Asilomar Conference on Signals, Systems, and Computers*, pp. 455–459, 2024.
- [13] S. Nabar, A. Banerjee, S. K. Gupta, and R. Poovendran, “GeM-REM: Generative model-driven resource efficient eeg monitoring in body sensor networks,” in *2011 International Conference on Body Sensor Networks*, pp. 1–6, 2011.
- [14] K. Sadeghi, A. Banerjee, J. Sohankar, and S. K. S. Gupta, “Toward parametric security analysis of machine learning based cyber forensic biometric systems,” in *ICMLA*, pp. 626–631, IEEE, 2016.
- [15] R. Singh and S. S. Gill, “Edge AI: a survey,” *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023.
- [16] A. Banerjee and S. Gupta, “EMILY: Extracting sparse model from implicit dynamics,” in *Proceedings of the 1st ECAI Workshop on Machine Learning Meets Differential Equations: From Theory to Applications* (C. Coelho, B. Zimmering, M. F. P. Costa, L. L. Ferrás, and O. Niggemann, eds.), vol. 255 of *Proceedings of Machine Learning Research*, pp. 1–11, PMLR, 20 Oct 2024.
- [17] A. Banerjee and S. K. Gupta, “Generalized model recovery from sampled data,” in *2024 58th Asilomar Conference on Signals, Systems, and Computers*, pp. 921–925, 2024.
- [18] Z. Chen, Y. Liu, and H. Sun, “Physics-informed learning of governing equations from scarce data,” *Nature communications*, vol. 12, no. 1, p. 6136, 2021.
- [19] A. Sholokhov, Y. Liu, H. Mansour, and S. Nabi, “Physics-informed neural ODE (PINODE): embedding physics into models using collocation points,” *Scientific Reports*, vol. 13, no. 1, p. 10166, 2023.
- [20] A. Mauroy, Y. Susuki, and I. Mezic, *Koopman operator in systems and control*, vol. 484, Springer, 2020.
- [21] B. Xu, A. Banerjee, and S. K. Gupta, “MERINDA: Model Recovery in FPGA based Dynamic Architecture,” *62nd DAC Work-in-Progress Poster*, 2025.
- [22] N. Verdière and S. Orange, “A systematic approach for doing an a priori identifiability study of dynamical nonlinear models,” *Mathematical biosciences*, vol. 308, pp. 105–113, 2019.
- [23] I. Stamoulias, M. Möller, R. Miedema, C. Strydis, C. Kachris, and D. Soudris, “High-performance hardware accelerators for solving ordinary differential equations,” in *Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, pp. 1–6, 2017.
- [24] A. Ebrahimi and M. Zandsalimy, “Evaluation of FPGA hardware as a new approach for accelerating the numerical solution of CFD problems,” *IEEE Access*, vol. 5, pp. 9717–9727, 2017.
- [25] M. Bilos, J. Sommer, S. S. Rangapuram, T. Januschowski, and S. Günnemann, “Neural flows: Efficient alternative to neural ODEs,” *Advances in neural information processing systems*, vol. 34, pp. 21325–21337, 2021.
- [26] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, “A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU,” *arXiv preprint arXiv:2305.17473*, 2023.
- [27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [28] C. Maxfield, *The design warrior’s guide to FPGAs: devices, tools and flows*. Elsevier, 2004.
- [29] I. Kuon, *Measuring and navigating the gap between FPGAs and ASICs*. PhD thesis, 2007.
- [30] C. Marling and R. Bunesco, “The OhioT1DM dataset for blood glucose level prediction: Update 2020,” in *CEUR workshop proceedings*, vol. 2675, p. 71, NIH Public Access, 2020.
- [31] S. Gupta, A. Banerjee, S. Nabar, and R. Poovendran, “Generative model-driven resource-efficient monitoring in body sensor networks,” Nov. 28 2013. US Patent App. 13/901,442.
- [32] A. D. AMD, “AMD Vitis™ High-Level Synthesis (HLS) Tool: Principles and Evolution,”
- [33] J. Cong, Z. Fang, M. Lo, H. Wang, J. Xu, and S. Zhang, “Understanding performance differences of FPGAs and GPUs,” in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 93–96, IEEE, 2018.
- [34] G. Ofenbeck, R. Steinmann, V. Caparros, D. G. Spampinato, and M. Püschel, “Applying the rooftop model,” in *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 76–85, IEEE, 2014.
- [35] S. Pawar, O. San, B. Aksoylu, A. Rasheed, and T. Kvamsdal, “Physics guided machine learning using simplified theories,” *Physics of Fluids*, vol. 33, no. 1, 2021.