PPGWeaver: Diffusion-Augmented Models for Real-Time Heart Rate Estimation on Microcontrollers

Vinayak Narasimhan
Samsung System LSI
Samsung Semiconductor
Pasadena, CA, USA
v.narasim@samsung.com

Raimi Shah
Synthefy
Austin, TX, USA
raimi@synthefy.com

Shubhankar Agarwal
Synthefy
Austin, TX, USA
somi@synthefy.com

Sai Shankar Narasimhan
Synthefy
Austin, TX, USA
sai@synthefy.com

Shailabh Kumar
Samsung System LSI
Samsung Semiconductor
Pasadena, CA, USA
shailabh.k@samsung.com

Sang Kyu Kim
Samsung System LSI
Samsung Electronics
Hwaseong, South Korea
sangq.kim@samsung.com

Sandeep Chinchali
Synthefy
Austin, TX, USA
sandeep@synthefy.com

Radwanul Hasan Siddique
Samsung System LSI
Samsung Semiconductor
Pasadena, CA, USA
r.siddique@samsung.com

***Abstract*—We present a compact, deployable heart rate (HR) estimation system using photoplethysmography (PPG) and inertial measurement unit (IMU) data, combining *TimeWeaver*, a conditional diffusion model for metadata-aware synthetic augmentation, with progressive structured pruning of Temporal Convolutional Networks (TCNs). Our smallest model, with 1.56k parameters, achieves a mean absolute error (MAE) of 4.92 BPM on the PPG-DaLiA dataset and supports real-time inference (<40 ms latency) on a 64 MHz ARM Cortex-M4F microcontroller (MCU) without requiring quantization. Synthetic data conditioned on subject metadata, HR, and activity type significantly enhances model generalization, enabling pruned models to match or exceed the accuracy of larger baselines, achieving over a 23% improvement compared to training on real data alone. Our work establishes a new Pareto frontier for real-time, on-device HR monitoring using diffusion-augmented training and sub-2k parameter models.**

Keywords—Heart Rate Estimation, Photoplethysmography, PPG, Synthetic Data, Diffusion Models, Model Pruning, Edge AI, Temporal Convolutional Networks, Microcontrollers, Wearables

I. Introduction

PPG sensors, often combined with IMU data, are widely used in wrist-worn HR devices but suffer from motion artifacts in real-world use. While deep learning models have outperformed traditional signal-processing approaches on benchmark datasets like PPG-DaLiA, the de facto benchmark for HR estimation used in nearly all state-of-the-art (SOTA) studies, challenges persist when deploying these models on constrained MCUs. Most SOTA methods compromise either accuracy, latency, or model size, and performance often degrades sharply when compressed below 10k parameters.

This work uses *TimeWeaver* [1], a conditional diffusion model that introduces a novel approach to PPG data synthesis by leveraging subject metadata, such as age, gender, body type, activity type and skin tone to enhance training data diversity. Our hypothesis is that targeted synthetic injection helps compensate for performance degradation at higher pruning levels by populating underrepresented HR ranges and activity classes, particularly for subjects and transitions poorly represented in the real training distribution. We integrate these synthetic signals with real PPG-DaLiA data to train a structurally pruned TCN-based model with only 1.56 k parameters, achieving real-time inference on an ARM Cortex-M4F in under 40 ms. Our system delivers SOTA accuracy (4.92 BPM MAE) among ultra-lightweight models without requiring quantization or hardware-specific tuning establishing a new Pareto frontier for real-time HR estimation (Fig. 1).

II. Background & Related Work

The PPG-DaLiA dataset (~36 hours of PPG and accelerometer data from 15 subjects) has emerged as the benchmark for HR estimation under real-world conditions [2]. Traditional digital-signal processing (DSP) pipelines were effective in lab settings but lack generalizability on unconstrained datasets like PPG-DaLiA.

Deep learning methods now dominate [2-10]. Early efforts like DeepPPG and NAS-PPG improved accuracy but were impractical for deployment due to size [2, 3]. Q-PPG introduced quantized TCNs with variants running on STM32 MCUs, showing that sub-2 kB models could achieve 7.73 BPM MAE with real-time performance [4]. EnhancePPG used self-supervised learning and classical augmentation to achieve 3.54 BPM MAE, albeit at the cost of higher latency and larger model size [5]. KID-PPG demonstrated further gains using domain knowledge but omitted deployability metrics [6]. Recent works like AugmentPPG introduced synthetic augmentation via sensor fusion and demonstrated efficient deployment on GAP8 [7]. However, these methods either rely on handcrafted transformations or fail to meet all constraints of low parameter count, high accuracy, and real-time performance simultaneously.

Recent notable studies have predominantly employed DSP-based augmentation techniques, which offer limited signal diversity and yield only modest performance gains (~5%) [5–7]. These insights motivate our approach of using *TimeWeaver* [1], a state-of-the-art diffusion model, to generate realistic, metadata-conditioned synthetic data. By coupling this with structured pruning, we address the dual challenge of improving generalization and accuracy while reducing model size and latency. This method yields a test performance improvement of

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

over 23% while producing ultra-compact models (<2K parameters) suitable for deployment on resource-constrained MCUs, without compromising full-precision accuracy.

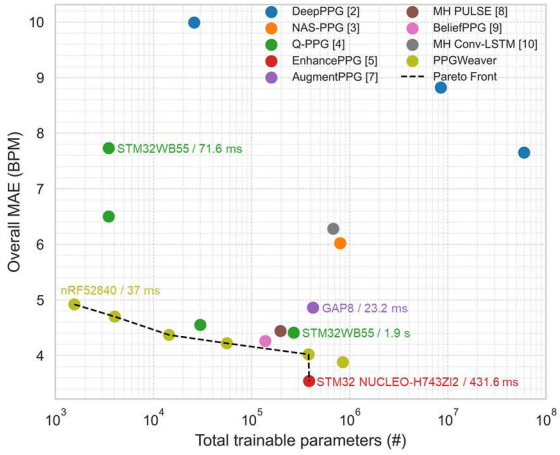

Fig. 1. PPGWeaver establishes a new Pareto frontier for PPG-based HR estimation, achieving lower error at significantly smaller model sizes. Models deployed on MCUs are annotated with reported runtime latencies. Deployability metrics are shown only for works with reported MCU results.

III. Methods

A. Seed Architecture

Our starting model adopts a lightweight TCN inspired by the seed TEMPONet [11] used in Q-PPG, which itself underpins a design space exploration framework producing Pareto-optimal tradeoffs between complexity and accuracy on the PPG-DaLiA dataset [4]. Our modified 512k-parameter seed network, called PPGNet-512k (Fig. 2a), consumes 256 sample windows from four channels (PPG + 3-axis IMU), and applies three stacked convolutional blocks featuring increasing dilation rates (1→2→4), interleaved with pooling and SE attention modules to capture temporal context while suppressing motion artifact noise. Feature maps expand from 32 to 128 channels across blocks, after which the representation is flattened and passed through fully connected layers (256→128 units with BatchNorm and ReLU), culminating in a scalar regression head for HR output. This over-parameterized architecture provides capacity for pruning and augmentation. We trained the seed on real PPG-DaLiA samples using Log Cosh regression loss for smooth convergence, with BatchNorm + ReLU ensuring stability during pruning and later quantization.

B. Structured Pruning With Synthetic Injection

To reduce model size while maintaining predictive accuracy, we applied structured pruning iteratively to the seed network described in Section 3.1, generating a progressively smaller set of architectures (PPGNet-512k→PPGNet-1.56k) by reducing convolutional channel widths and dense layer sizes in a controlled manner. Each pruned architecture was first trained and evaluated using only the real PPG-DaLiA dataset following a robust 4×Leave-One-Group-Out cross-validation protocol [2, 4]. In this setup, outer folds were created by grouping subjects, with inner folds assigning individual subjects as held-out test sets. Each model was trained on real windows and validated on unseen subjects. This real-data-only evaluation defined the baseline accuracy of each pruned model. To improve performance, we then incrementally introduced synthetic training windows generated by *TimeWeaver*, into the same training folds, beginning with 5% of available synthetic data and progressing up to 100%. Synthetic windows were never included in validation or test sets. For a given test subject, synthetic time-series data was generated using the metadata of that subject. At each increment, models were warm-started from previous weights, using the same architecture, optimizer, and stopping strategy. The result is a series of Pareto-efficient models whose size-performance trade-off improves significantly with synthetic augmentation.

C. Synthetic Data Generation Via Conditional Diffusion

To generate high-fidelity synthetic PPG signals for data augmentation, we utilized *TimeWeaver*, a conditional diffusion model trained to synthesize PPG + IMU + ECG signals conditioned on rich metadata. Importantly, ECG signals are used only to generate ground-truth HR labels during training and are never inputs to the deployed predictor model. Unlike adversarial generative models, *TimeWeaver* follows a denoising score-matching framework that iteratively learns to reverse a noise process applied to real time-series waveforms. By leveraging heterogeneous, time-varying metadata, *TimeWeaver* achieves up to 40× better performance than GANs and conventional methods on real-world datasets [1]. Each synthetic window is 512 samples long, corresponding to an 8-second segment at 64 Hz. These windows are later downsampled to 32 Hz during model training. Metadata including subject ID, activity type, target HR, skin tone (binned via Fitzpatrick scale), and session time are embedded and injected into both the conditioning and denoising paths of the model using a Conditional Score-based Diffusion Imputation (CSDI) architecture. Categorical variables are encoded through learned embeddings, while continuous metadata is projected via dense layers and fused with attention mechanisms. The model is trained end-to-end on PPG-DaLiA data using a linear noise schedule over 200 timesteps and was selected based on minimum validation loss.

D. MCU Deployment

To enable MCU deployment, each trained TensorFlow model was converted to TFLite using four quantization strategies: FP32, FP16, INT8, and dynamic-range mixed precision. INT8 required a representative dataset from PPG-DaLiA, while others used default optimizations. All variants were benchmarked on a CPU (13th Gen Intel i7-1360P, 2.20 GHz) for latency, memory, and HR accuracy. Despite testing quantized models, the final deployment used FP32 TFLite due to its small size and full-precision fidelity. Models were serialized into C arrays via xxd for ARM Cortex M4F toolchain compatibility. Note that *TimeWeaver* is used only offline during training to generate synthetic data; the deployed MCU runs solely the compact 1.56k-parameter predictor.

The final pruned FP32 TensorFlow Lite model was deployed to an Arduino Nano 33 BLE Rev 2, which features a Nordic nRF52840 MCU with a 32 bit ARM Cortex M4F core running at 64 MHz, along with 256 kB SRAM and 1 MB flash. Model inference is implemented in an Arduino sketch, which receives one 8-second (256 samples × 4 channels) window over serial, runs inference using TFLite Micro, and sends back a predicted

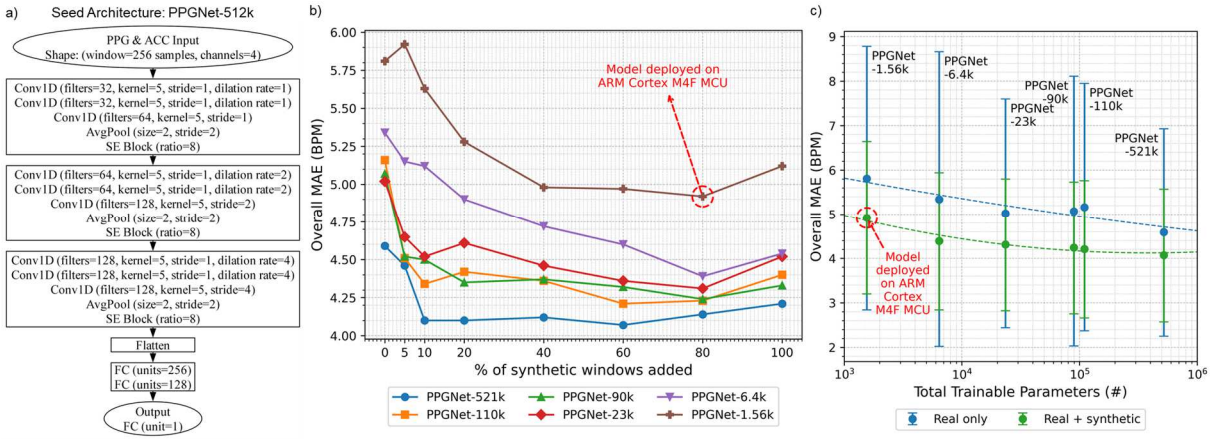

Fig. 2. a) PPGNet-512. b), c) Synthetic augmentation to improve test MAE maintaining pareto efficiency

HR and measured latency. The code allocates a 20 kB tensor arena, resolves all necessary ops, and invokes the TFLite interpreter on the input window stored in the flat float tensor.

IV. RESULTS

A. Traversing The Pareto Frontier With TimeWeaver

To explore the trade-off between model compactness and predictive accuracy, we evaluated eight structurally pruned architectures under progressive synthetic data augmentation using the *TimeWeaver* generator. For each model, we performed end-to-end training on real-only data (129,369 windows), then repeated training with synthetic windows added in increasing proportions up to 100% (an additional 107,238 windows), maintaining the same cross-validation protocol.

Across all model variants, adding even a small fraction (5–10%) of synthetic data significantly improved performance, with the strongest gains (> 23%) observed in mid-size models. For instance, PPGNet-436k (not shown) improved from 5.29 BPM MAE to 4.06 BPM with just 5% synthetic augmentation. Similarly, PPGNet-110k improved from 5.16 to 4.21 BPM MAE at 60% augmentation. Notably, compact models such as PPGNet-1.56k-Dilated saw MAE drop from 5.82 to 4.80 BPM with 80% synthetic data, a substantial gain despite limited capacity (Fig. 2b, c).

The benefit of synthetic augmentation plateaued between 60%–80% for most architectures, beyond which improvements plateaued or reversed slightly. These results suggest diminishing marginal returns at high augmentation ratios. Overall, synthetic injection helped recover or even surpass baseline performance levels for models that had been heavily pruned, effectively shifting the accuracy–efficiency Pareto frontier upward.

B. Synthetic Data Generated With TimeWeaver

To evaluate the fidelity and utility of synthetic data generated by *TimeWeaver*, we conducted a series of statistical and downstream analyses. First, we assessed signal realism by comparing the distributions of amplitude and spectral energy between real PPG and synthetic data. As shown in Fig. 3a, b, *TimeWeaver*-generated samples align closely with ground-truth distributions in both time and frequency domains, while GAN baselines fail to capture multimodal or skewed properties under high-motion or high-HR conditions.

Next, to assess whether *TimeWeaver* can generate physiologically plausible PPG signals, we performed qualitative analysis on held-out metadata conditions. In one set of experiments, we aimed to simulate subjects with high resting HRs. For example, Subject S5 in PPG-DaLiA naturally exhibits a baseline HR of approximately 125 BPM. As shown in Fig. 3c, the synthesized waveform from Subject S11, conditioned on a target HR of 125 BPM, closely resembled the ground-truth waveform of Subject S5 in both periodic structure and amplitude morphology.

C. MCU Deployment

We evaluated the full inference pipeline from TensorFlow to TFLite conversion and final deployment on a resource-constrained MCU. The smallest pruned model, PPGNet-1.56k, trained with 80% synthetic data, was converted to TFLite using four quantization configurations: FP32, FP16, INT8, and mixed precision. Notably, all TFLite variants of PPGNet-1.56k achieved sub-0.025 ms average latency per 256-sample window on a 13th Gen Intel i7 CPU, with FP32 executing in just 0.019 ms. Due to its extremely small size, PPGNet-1.56k exhibited minimal memory and latency overhead even at full-precision (FP32), enabling real-time inference (~37 ms latency) without requiring INT8 quantization on resource-constrained MCUs like the ARM Cortex-M4. This suggests that, in certain cases, model compression and data-driven regularization can eliminate the need for aggressive quantization, even for deployment on low-power devices.

Given these findings, the FP32 variant of PPGNet-1.56k was selected for deployment on a 64 MHz Arm Cortex-M4F MCU with 256 KB SRAM. Inference results across all 15 PPG-DaLiA test subjects revealed identical performance to the parent TensorFlow model, with an overall MAE of 4.92 BPM (Figs. 4a-c). Importantly, this confirms that the MCU deployment retained full numerical fidelity without requiring quantization, made possible by an extremely

compact Pareto-efficient model enabled by *TimeWeaver*-generated synthetic data. This establishes the feasibility of real-time HR estimation directly on MCU-class wearables without compromising on accuracy or throughput.

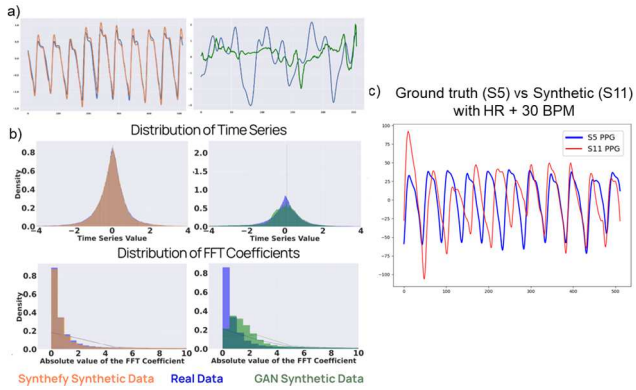

Fig. 3. a), b) *TimeWeaver*-generated PPG signals (orange) closely match unseen ground truth (blue), outperforming GANs (green) in both time and frequency domains. c) Comparison of real PPG from Subject S5 (blue) and TimeWeaver-generated data by increasing Subject S11's HR.

V. CONCLUSION

This work presents a complete pipeline for real-time, MCU-based HR estimation using PPG and IMU data, addressing the longstanding trade-off between accuracy, model size, and deployability. Through structured pruning and novel synthetic augmentation via *TimeWeaver*, we achieve over 23% performance gains and show that ultra-lightweight sub-2 k parameter models can match or surpass SOTA accuracy. Our smallest model, with only 1.56k parameters, achieved 4.92 BPM MAE and maintained identical performance when deployed on a 64 MHz ARM Cortex M4F MCU without requiring quantization. This underscores the strength of combining generative augmentation with hardware-aware compression.

Future work will focus on applying *TimeWeaver* to proprietary datasets with richer sensing modalities and more diverse activities and user populations. Our goal is to extend this approach toward commercial-grade wearable algorithms capable of robust HR estimation on edge platforms, paving the way for scalable, low-power, on-device health monitoring.

REFERENCES

[1] S. S. Narasimhan, *et al.*, "Time Weaver: A Conditional Time Series Generation Model," in *Proc. 41st International Conference on Machine Learning (ICML)*, Vienna, Austria, Jul. 2024, vol. 235, pp. 37293–37320.

[2] A. Reiss *et al.*, "Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks," Sensors, vol. 19, no. 14, p. 3079, 2019

[3] S. B. Song, J. W. Nam, and J. H. Kim, "NAS-PPG: PPG-Based Heart Rate Estimation Using Neural Architecture Search," IEEE Sensors Journal, vol. 21, no. 13, pp. 14941–14949, Jul. 1, 2021.

[4] A. Burrello *et al.*, "Q-PPG: Energy-Efficient PPG-Based Heart Rate Monitoring on Wearable Devices," IEEE Transactions on Biomedical Circuits and Systems, vol. 15, no. 6, pp. 1196–1209, Dec. 2021.

[5] L. Benfenati *et al.*, "EnhancePPG: Improving PPG based Heart Rate Estimation with Self Supervision and Augmentation," arXiv, Dec. 2024.

[6] C. Kechris *et al.*, "KID-PPG: Knowledge Informed Deep Learning for Extracting Heart Rate From a Smartwatch," IEEE Transactions on Biomedical Engineering, vol. 72, no. 3, pp. 870–877, Mar. 2025.

[7] A. Burrello *et al.*, "Improving PPG-based Heart-Rate Monitoring with Synthetically Generated Data," in Proc. 2022 IEEE Biomedical Circuits and Systems Conference (BioCAS), Taipei, Taiwan, 2022, pp. 153–157.

[8] P. Kasnesis *et al.*, "Multi Head Cross Attentional PPG and Motion Signal Fusion for Heart Rate Estimation," arXiv preprint arXiv:2210.11415, Oct. 2022.

[9] V. Bieri *et al.*, "BeliefPPG: Uncertainty-aware Heart Rate Estimation from PPG Signals via Belief Propagation," arXiv, June 2023.

[10] M. Wilkosz and A. Szczęsna, "Multi-Headed Conv-LSTM Network for Heart Rate Estimation during Daily Living Activities," *Sensors*, vol. 21, no. 15, art. 5212, Jul. 31, 2021.

[11] M. Zanghieri *et al.*, "Robust Real-Time Embedded EMG Recognition Framework Using Temporal Convolutional Networks on a Multicore IoT Processor," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 2, pp. 244–256, Apr. 2020.

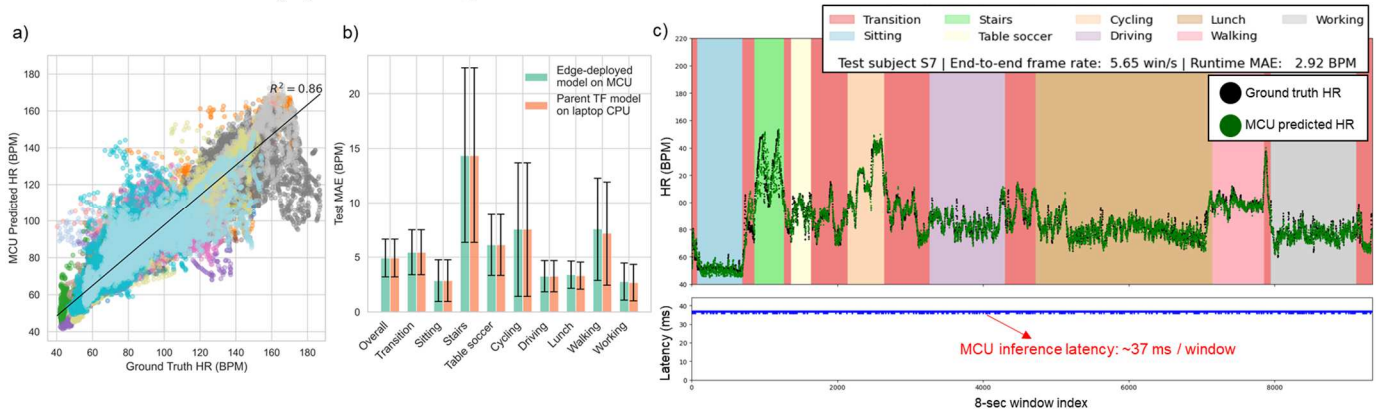

Fig. 4. a) Correlation ($R^2 = 0.86$) between MCU-predicted HR from PPGNet-1.56k and ground-truth HR (color-coded by subject). b) MCU accuracy matches parent TensorFlow model via full-precision deployment. c) Real-time inference on Subject S7 with <40 ms per-window latency.