REAL-TIME CAMERA-BASED HEART RATE ESTIMATION SYSTEM USING DEEP LEARNING ON EDGE DEVICE

Chiquita Ahsanunnisa[⋆]*, Fariska Z. Ruskanda*[⋆†]*, Robithoh Annur*[⋆]*, Rinaldi Munir*[⋆]*, Nur Ahmadi*[⋆†‡]

[⋆]School of Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia
[†]Center for Artificial Intelligence (U-CoE AI-VLB), Bandung Institute of Technology, Indonesia
[‡]Center for Microelectronics, Bandung Institute of Technology, Indonesia

2026 IEEE 23rd International Symposium on Biomedical Imaging (ISBI) | 979-8-3315-7763-6/26/$31.00 ©2026 IEEE | DOI: 10.1109/ISBI61048.2026.11515871

ABSTRACT

Remote photoplethysmography (rPPG) provides a non-contact solution for heart rate (HR) monitoring, but deploying deep learning-based rPPG algorithms on resource-limited edge devices remains challenging. This work presents a real-time camera-based HR estimation system running on the Raspberry Pi AI Kit equipped with a Hailo-8 AI accelerator. The proposed pipeline integrates face detection, tracking, rPPG signal extraction, and HR estimation in a fully end-to-end framework. Through benchmarking on the WIDERFACE and UBFC-rPPG datasets, SCRFD-2.5G was selected for face detection and DeepPhys for rPPG extraction based on accuracy–latency trade-offs. Quantization of DeepPhys, including a mixed-precision 20% UINT4 and 80% UINT8 scheme with equalization, reduced latency from 163,493.80 ms to 28.94 ms while preserving accuracy (MAE 3.01 bpm). Additional pipeline optimizations—window chunking, batch inference, and incremental statistics—further improved real-time performance. Results demonstrate that deep learning-based rPPG systems can be efficiently deployed on edge device while meeting both accuracy and real-time requirements for practical use.

Index Terms— Real-time, heart rate, remote photoplethysmography (rPPG), deep learning, AI accelerator.

1. INTRODUCTION

Heart rate (HR) is a vital physiological parameter commonly measured using electrocardiography (ECG) or photoplethysmography (PPG). Although accurate, these contact-based methods may cause discomfort for individuals with sensitive skin and increase the risk of disease transmission. To address these limitations, remote photoplethysmography (rPPG) has emerged as a non-contact alternative that estimates HR by detecting subtle color variations in facial skin caused by blood volume changes [1].

rPPG techniques are categorized into two groups [2]: (1) conventional methods (e.g., POS [3], ICA [4]) and (2) deep learning–based approaches (e.g., DeepPhys [5], PhysFormer [6]). Conventional methods are computationally lightweight and suitable for real-time deployment, but they generally lack robustness in unconstrained environments. Deep learning methods extract richer spatiotemporal features and offer superior accuracy and stability, yet their computational demands typically prevent real-time operation on embedded hardware.

With increasing interest in portable, non-contact physiological monitoring, there is a growing need for rPPG systems that can operate in real time on resource-constrained edge devices. Real-time performance is essential for applications including telemedicine, fitness tracking, and public health monitoring. However, deploying deep learning–based rPPG algorithms on embedded systems presents a fundamental challenge: achieving low latency while maintaining estimation accuracy.

Previous works [7, 8] have demonstrated real-time rPPG pipelines on the Raspberry Pi, but these systems have relied primarily on conventional algorithms, which as noted in [2], often degrade under motion or lighting variations. Although deep learning methods offer improved robustness, they have rarely been realized in embedded real-time settings due to their high computational cost.

To bridge this gap, this study develops a real-time deep learning–based rPPG system on the Raspberry Pi using the Hailo-8 AI accelerator. This hardware accelerator enables efficient inference, allowing advanced rPPG models to achieve both high accuracy and real-time performance on edge devices.

2. METHODOLOGY

2.1. Data Collection

Three publicly available datasets were used in this study: WIDERFACE [9], PURE [10], and UBFC-rPPG [11]. WIDERFACE, a large-scale face detection benchmark with 32,203 images and 393,703 annotated faces, was used for evaluating the face detection module using its publicly available validation split. PURE contains 59 video–pulse oximeter recordings from 10 subjects across six controlled movement scenarios and served as the primary dataset for developing

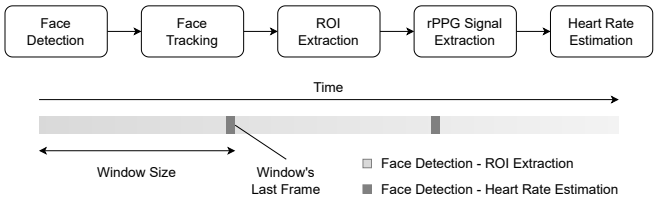

Fig. 1. Flowchart of the proposed method.

and refining the deep learning models, with processing conducted using the rPPG-Toolbox [12]. UBFC-rPPG consists of 42 video–pulse oximeter recordings from 42 subjects under natural movement and was used for benchmarking and final evaluation. PURE was chosen for model development based on prior evidence that models trained on PURE achieve better performance than those trained on UBFC-rPPG [12].

2.2. Proposed Method

Fig. 1 illustrates the proposed method which processes prerecorded video input. All processing is performed on the Raspberry Pi AI Kit and the final output (HR) is displayed on a monitor. The main modules are as follows:

2.2.1 Face Detection. This module identifies facial regions of interest (ROI) for rPPG processing. Since the system runs on the Hailo-8 NPU, it can directly utilize face detection models optimized for this hardware. Degirum AI Hub provides several compatible options, including YOLO-based single-stage detectors, RetinaFace with five-point landmark regression using ResNet or MobileNet backbones, and SCRFD [13], a lightweight detector with variants such as 0.5G, 2.5G, and 10G designed for edge deployment. Because no prior benchmarking exists across these models on the NPU, a comparative evaluation was conducted using the WIDERFACE validation split to determine the most suitable model for this system.

2.2.2 Face Tracking. This module tracks the position of the face in each frame. For real-time requirements, a simple Centroid Tracker [14] was employed. Centroid Tracker is an object tracking method that relies on calculating the Euclidean distance between the centroids of existing objects and the centroids of newly detected objects in consecutive video frames.

2.2.3 ROI Extraction. This module extracts the ROI used as input for the signal extraction stage. For deep learning–based methods, the ROI is a face-bounding box resized to the model's input requirements. For conventional methods, the ROI can be more flexible, such as the full face, cheeks, or forehead.

2.2.4 rPPG Signal Extraction. This module extracts physiological signals from detected faces using either conventional or deep learning–based methods. Conventional approaches include GREEN, ICA, CHROM, PBV, POS, LGI, and OMIT, each relying on color channel transformations or statistical separation to isolate pulse-related variations.

Fig. 2. Flow of the proposed pipeline modules.

Deep learning methods aim to improve robustness under challenging conditions. Representative models include DeepPhys [5], PhysNet, TS-CAN, EfficientPhys, PhysFormer, FactorizePhys, iBVPNet, PhysMamba, and RhythmFormer [15]. These models leverage spatiotemporal learning and attention mechanisms, and can be evaluated using the rPPG-Toolbox [12], which provides pretrained models and standardized metrics.

Selecting a deep learning model introduces challenges for real-time deployment due to higher computational demand. Although the Hailo-8 NPU can mitigate this, no rPPG models are currently adapted or quantized for this accelerator, necessitating additional optimization for real-time performance.

2.2.5 HR Estimation. This module extracts the dominant frequency from the rPPG signal, which fundamentally represents the HR. To obtain the dominant frequency, Fast Fourier Transform (FFT) was applied to the signal. Prior to frequency extraction, bandpass filtering was used to isolate the relevant HR frequencies, with a low-pass cutoff of 0.75 Hz and a high-pass cutoff of 2.5 Hz, following the rPPG-Toolbox study [12].

2.3. Real-Time Deployment

The deployment hardware consists of a Raspberry Pi 4 with an Arm Cortex-A76 quad-core processor, 8 GB RAM, a Hailo-8L NPU, 128 GB storage, and Debian 12 (Bookworm). During deployment, an HDMI-connected monitor displays the real-time HR and FPS results. The overall processing pipeline is illustrated in Fig. 2. The proposed method was evaluated using the UBFC-rPPG dataset. During evaluation, each video was processed frame by frame through the pipeline on the Raspberry Pi AI Kit. The estimated HR and FPS were displayed on the monitor, while all results were simultaneously logged to a CSV file for further analysis.

2.4. Evaluation Metrics

Average Precision (AP) was used to assess face detection accuracy, while Mean Absolute Error (MAE) was used to quantify HR estimation accuracy by comparing predicted values with ground truth measurements. Real-time performance is evaluated using latency, defined as the delay between processing initiation and output, and Frames Per Second (FPS), which indicates the number of frames the system can process or display per second.

Table 1. Face Detection Benchmark on WIDERFACE

Model	AP (%) ↑			Latency ↓
	Easy	Medium	Hard	(ms)
YOLOv8n-ReLU6-Face	87.30	78.86	44.66	12.37
SCRFD-0.5G	89.06	85.72	63.66	20.44
SCRFD-2.5G	93.18	90.87	74.08	18.91
SCRFD-10G	95.09	93.38	80.89	26.57
RetinaFace-MobileNet	87.27	82.63	61.30	69.63

3. RESULTS AND DISCUSSION

3.1. Pipeline Component Selection

3.1.1. Face Detection

Face detection is a critical component in the real-time pipeline, as it is invoked continuously for every frame. Benchmarking was performed directly on the Raspberry Pi AI Kit using the WIDERFACE validation set, with results summarized in Table 1. Latency is a key factor: based on our own experiments, face detection must achieve less than 20 ms per frame to avoid choppy performance, since computational resources are shared with other pipeline modules. Among the models with latency below 20 ms, SCRFD-2.5G demonstrated the highest accuracy and was therefore chosen as the final model.

3.1.2. rPPG Signal Extraction

Accuracy benchmarking was conducted on the UBFC-rPPG dataset using the rPPG-Toolbox with default configurations. All models were pretrained on the PURE dataset. The results are summarized in Table 2. It is evident that deep learning-based methods (MAE range: 0.837–2.992 bpm) significantly outperform conventional methods (MAE range: 3.735–19.775 bpm). Therefore, a deep learning-based approach was clearly chosen. Among all deep learning models, DeepPhys was selected as the final model since its layer structure is fully supported by the Hailo accelerator. Other models were not supported due to incompatible layers such as tensor 5D reshape operations, expand functions, and 3D CNN layers that are not supported by the Hailo platform.

To leverage the Hailo-8 NPU capabilities, the selected DeepPhys model required quantization and optimization. The quantization process utilized a calibration dataset consisting of 128 randomly sampled images from the PURE training set (0-80%), while validation was performed on the PURE validation set (80-100%). Several experiments were conducted to optimize the DeepPhys model while maintaining acceptable accuracy as summarized in Table 3. Full UINT8 quantization reduced latency from 163,493.80 ms (Raspberry Pi 4, FP32) to 33.55 ms on the Hailo-8, with a small MAE increase from 3.01 bpm to 3.28 bpm. The proposed model—quantized with 20% UINT4 and 80% UINT8 precision and optimized through equalization—achieved a substantial reduction in average latency from 163,493.80 ms to 28.94 ms when deployed on the Raspberry Pi 4 with the Hailo-8 NPU. Despite the drastic improvement in computational efficiency, the MAE remained unchanged at 3.01 bpm, indicating that the quantization and hardware acceleration introduced no significant loss in accuracy. The proposed implementation also achieved an average performance of 29.972 FPS, which is nearly identical to the standard 30 FPS camera input rate, demonstrating that the system can operate effectively under real-time conditions. This result highlights the effectiveness of the proposed optimization strategy in achieving real-time performance while preserving model reliability.

Table 2. MAE Comparison Between Conventional and Deep Learning Methods

Conventional Methods		Deep Learning Methods	
Method	MAE (bpm) ↓	Method	MAE (bpm) ↓
ICA	15.203	DeepPhys	1.109
POS	3.735	EfficientPhys	1.381
CHROM	3.934	FactorizePhys	1.130
GREEN	19.775	iBVPNet	2.992
LGI	15.663	PhysFormer	1.465
PBV	15.140	PhysMamba	1.528
OMIT	15.663	PhysNet	2.658
		RhythmFormer	0.837
		TS-CAN	1.172

3.2. Final Pipeline

Based on the component selection process, the final system integrates SCRFD-2.5G for face detection and a proposed quantized DeepPhys model for rPPG extraction. To achieve stable real-time performance, three optimizations were applied. (1) Window chunking: the original 180-frame window was split into 30-frame chunks to avoid computational spikes by distributing the workload across time. (2) Batch inference: chunk-level batch processing was introduced to leverage the parallelism of the Hailo-8 NPU and reduce per-frame overhead. (3) Incremental statistics: DeepPhys preprocessing statistics were updated continuously as frames arrived, removing expensive on-demand computation and enabling smooth end-to-end operation.

4. CONCLUSION AND FUTURE WORK

This study demonstrated the feasibility of deploying deep learning-based rPPG algorithms on resource-constrained edge devices by developing a real-time, camera-based heart rate estimation system using the Raspberry Pi AI Kit. The proposed system achieved an effective balance between accuracy and real-time performance through systematic component selection, model quantization, and pipeline optimization. While DeepPhys was fully supported by the Hailo-8 accelerator, other high-performing models such as RhythmFormer

Table 3. Performance of DeepPhys Model Modifications

Model	Runtime Environment	Bit Precision	Optimization	Latency (ms) ↓	MAE (bpm) ↓
Baseline	Raspberry Pi 4	100% FP32	–	163,493.80	3.01
Experiment	Raspberry Pi 4 + Hailo-8	100% UINT8	–	33.55	3.28
Proposed	Raspberry Pi 4 + Hailo-8	20% UINT4, 80% UINT8	Equalization	**28.94**	**3.01**

and FactorizePhys were incompatible due to unsupported layer operations. Future work may explore architectural modifications of these models and advanced quantization methods such as adaptive rounding and quantization-aware fine-tuning to improve the balance between accuracy and real-time performance.

5. COMPLIANCE WITH ETHICAL STANDARDS

This research study was conducted retrospectively using human subject data made available by Bobbia *et al.* [11].

6. ACKNOWLEDGMENTS

This work was supported by the PPMI ITB Research Grant 2025 (No. 1148/IT1.C12/TA/2025) and the Indonesian Endowment Fund for Education (LPDP) on behalf of the Indonesian Ministry of Higher Education, Science and Technology and managed under the EQUITY Program (Contract No. 4298/B3/DT.03.08/2025).

7. REFERENCES

[1] Wim Verkruysse, Lars O Svaasand, and J Stuart Nelson, "Remote plethysmographic imaging using ambient light.," *Optics Express*, vol. 16, no. 26, pp. 21434–21445, 2008.

[2] Daniel McDuff, "Camera measurement of physiological vital signs," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–40, 2023.

[3] Wenjin Wang, Albertus C Den Brinker, Sander Stuijk, and Gerard De Haan, "Algorithmic principles of remote ppg," *IEEE Trans. Biomed. Eng.*, vol. 64, no. 7, pp. 1479–1491, 2016.

[4] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard, "Advancements in noncontact, multiparameter physiological measurements using a webcam," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 1, pp. 7–11, 2010.

[5] Weixuan Chen and Daniel McDuff, "Deepphys: Video-based physiological measurement using convolutional attention networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 349–365.

[6] Zitong Yu, Yuming Shen, Jingang Shi, Hengshuang Zhao, Philip HS Torr, and Guoying Zhao, "Physformer: Facial video-based physiological measurement with temporal difference transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 4186–4196.

[7] Nur Ahmadi et al., "Development and evaluation of a contactless heart rate measurement device based on rppg," in *IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*. IEEE, 2022, pp. 1–4.

[8] Muhammad H Hyanda, Nur Ahmadi, Peter H Charlton, Timothy G Constandinou, Ayu Purwarianti, and Trio Adiono, "A comparative evaluation of video codecs for rppg-based heart rate estimation," in *Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*. IEEE, 2023, pp. 243–247.

[9] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang, "Wider face: A face detection benchmark," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5525–5533.

[10] Ronny Stricker, Steffen Müller, and Horst-Michael Gross, "Non-contact video-based pulse rate measurement on a mobile service robot," in *IEEE Int. Symp. Robot Human Interact. Commun.* IEEE, 2014, pp. 1056–1062.

[11] Serge Bobbia, Richard Macwan, Yannick Benezeth, Alamin Mansouri, and Julien Dubois, "Unsupervised skin tissue segmentation for remote photoplethysmography," *Pattern Recognit. Lett.*, vol. 124, pp. 82–90, 2019.

[12] Xin Liu et al., "rppg-toolbox: Deep remote ppg toolbox," in *Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 68485–68510.

[13] Jia Guo, Jiankang Deng, Alexandros Lattas, and Stefanos Zafeiriou, "Sample and computation redistribution for efficient face detection," in *Int. Conf. Learn. Represent. (ICLR)*. IEEE, 2022, pp. 1–17.

[14] A Bakliwal et al., "Crowd counter: An application of centroid tracking algorithm," *Int. Res. J. Modern. Eng. Technol. Sci.*, vol. 2, no. 4, pp. 1138–1141, 2020.

[15] Bochao Zou, Zizheng Guo, Jiansheng Chen, Junbao Zhuo, Weiran Huang, and Huimin Ma, "Rhythmformer: Extracting patterned rppg signals based on periodic sparse attention," *Pattern Recognition*, vol. 164, pp. 111511, 2025.