Template Matching Based Early Exit CNN for Energy-efficient Myocardial Infarction Detection on Low-power Wearable Devices

NAFIUL RASHID, University of California, Irvine, USA
BERKEN UTKU DEMIREL, University of California, Irvine, USA
MOHANAD ODEMA, University of California, Irvine, USA
MOHAMMAD ABDULLAH AL FARUQUE, University of California, Irvine, USA

Myocardial Infarction (MI), also known as heart attack, is a life-threatening form of heart disease that is a leading cause of death worldwide. Its recurrent and silent nature emphasizes the need for continuous monitoring through wearable devices. The wearable device solutions should provide adequate performance while being resource-constrained in terms of power and memory. This paper proposes an MI detection methodology using a Convolutional Neural Network (CNN) that outperforms the state-of-the-art works on wearable devices for two datasets – PTB and PTB-XL, while being energy and memory-efficient. Moreover, we also propose a novel Template Matching based Early Exit (TMEX) CNN architecture that further increases the energy efficiency compared to baseline architecture while maintaining similar performance. Our baseline and TMEX architecture achieve 99.33% and 99.24% accuracy on PTB dataset, whereas on PTB-XL dataset they achieve 84.36% and 84.24% accuracy, respectively. Both architectures are suitable for wearable devices requiring only 20 KB of RAM. Evaluation of real hardware shows that our baseline architecture is 0.6× to 53× more energy-efficient than the state-of-the-art works on wearable devices. Moreover, our TMEX architecture further improves the energy efficiency by 8.12% (PTB) and 6.36% (PTB-XL) while maintaining similar performance as the baseline architecture.

CCS Concepts: • **Computer systems organization** → *Embedded systems*; • **Computing methodologies** → *Artificial intelligence*; • **Hardware** → *Digital signal processing*; *Biology-related information processing*.

Additional Key Words and Phrases: myocardial infarction, early exit CNN, energy-efficiency, wearable devices

ACM Reference Format:
Nafiul Rashid, Berken Utku Demirel, Mohanad Odema, and Mohammad Abdullah Al Faruque. 2022. Template Matching Based Early Exit CNN for Energy-efficient Myocardial Infarction Detection on Low-power Wearable Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 2, Article 68 (June 2022), 22 pages. https://doi.org/10.1145/3534580

1 INTRODUCTION

Myocardial Infarction (MI) is one of the most fatal forms of heart disease being the leading cause of death worldwide. MI occurs when one of the coronary arteries responsible for supplying oxygenated blood to the heart muscle gets blocked. This blockage happens due to the deposition of plaques on the inner wall of the coronary arteries. Eventually, the heart muscle suffers from the shortage of oxygen and essential nutrients leading to a heart attack. About 805,000 people in the USA suffer from it every year. About 75% of them encounter a heart attack for the first time and the rest 25% experience recurrent heart attacks [1] which has a 6 times higher mortality rate

Authors' addresses: Nafiul Rashid, University of California, Irvine, Irvine, USA, nafiulr@uci.edu; Berken Utku De mirel, University of California, Irvine, Irvine, USA, bdemirel@uci.edu; Mohanad Odema, University of California, Irvine, Irvine, USA, modema@uci.edu; Mohammad Abdullah Al Faruque, University of California, Irvine, Irvine, USA, alfaruqu@uci.edu.

2474-9567/2022/6-ART68
https://doi.org/10.1145/3534580

than the first ones [45]. The higher risk associated with recurrent heart attacks requires continuous monitoring of those patients. Moreover, 1 out of 5 heart attacks is silent where victims are unaware of the damage [1]. And the mortality rate significantly increases by 41-62% if the treatment is delayed for more than 2 hours from MI initiation [18]. The aforementioned facts necessitate the real-time monitoring and detection of MI. Currently, most of the monitoring takes place in a clinical environment with bulky medical equipment which lacks portability. Therefore, wearable devices represent a more convenient solution for continuous monitoring on a daily basis.

Besides, a real-time monitoring solution using wearable devices enables physicians to keep track of their patients' health remotely. Currently, wearable device solutions for mobile health follow a cloud computing architecture where the raw data from wearable devices is offloaded to fog (mobile phones) or cloud (remote servers) where all the processing takes place [34]. This offloading consumes a huge amount of communication energy which reduces the battery life of wearable devices, as well as mobile phones [26]. In addition to that, it also introduces latency which hinders the real-time monitoring and detection in healthcare applications [40]. Moreover, offloading the raw data to a mobile phone or cloud makes the users' data vulnerable to privacy breaches. To overcome these aforementioned limitations, the 'Edge Computing' [23] paradigm has gained momentum in recent years where all the processing is done on the wearable devices and only the analyzed results are sent to the cloud for remote monitoring. Thus, it addresses the aforementioned issues related to energy consumption, latency, and vulnerability of privacy breaches.

The designed algorithms for wearable devices should be energy-efficient, memory-efficient, and provide acceptable performance within the previous two constraints. State-of-the-art works on MI detection are not wearable device compatible as they prioritize performance and do not consider the other two constraints. They use complex machine learning [4, 37, 38] and deep learning algorithms [3, 27, 30, 33, 39] to achieve high performance. Machine learning algorithms perform classification based on the extracted features from the data. However, the feature extraction processes are often time-consuming and require a huge amount of energy. Deep learning algorithms like Convolutional Neural Networks (CNN) do not require manual feature engineering and extraction as they automatically extract features through convolution. Moreover, the layered architecture of CNN provides flexibility to design a network by adding or removing layers as necessary in the training phase. Later, this architecture may be used to classify data during the inference phase. However, the full architecture from the training phase may not be needed at the inference phase as many of the data can be correctly classified using only the first few layers of the architecture. This early exit capability of CNN should help to avoid redundant operations during the inference phase leading to energy efficiency for wearable device solutions while maintaining the performance.

1.1 Motivational Example

We conducted a small experiment to demonstrate the advantage of the early exit CNN architecture. We have created a multi-output CNN architecture with 2 convolution blocks and 2 output blocks. One output block is used after each of the convolution blocks to allow the early exit after any convolution block at the inference phase. Each convolution block consists of one convolution layer, one pooling layer, and one batch normalization layer. The details of the multi-output CNN architecture are provided in Section 3.2.1. Throughout the rest of the paper, the first output block (FOB) is used to represent the CNN architecture that exits after the first convolution block. The second output block is considered as the baseline[1] architecture that exits after the second convolution block. 5-fold cross-validation of the multi-output CNN architecture is performed with 62306 heartbeat segments extracted from 200 patients of the PTB diagnostic ECG database [7]. Figure 1 shows the blockwise statistics of the first and second output block. As demonstrated in Figure 1, the FLOP counts, execution time, and energy increases as performance increases from the first to second output block. We choose the second output block

[1]Baseline architecture uses full architecture. No dynamic model compression techniques or early exit strategies are applied to it

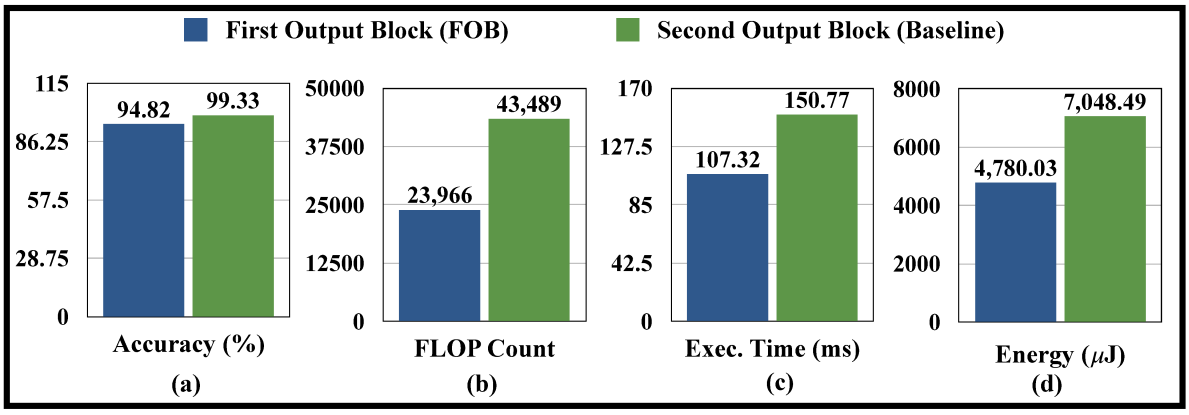

Fig. 1. Blockwise Statistics of Multi-output CNN Architecture

Table 1. Difference between Baseline and Early Exit Architecture

Architecture Used	Output Block Used	% of Total Segments	# of Total Segments	Total FLOP Count	Total Exec. Time (S)	Total Energy (J)
Baseline	**Second**	**100**	**62306**	**2709.62x10^6**	**9393.87**	**439.16**
Early Exit	First	94	58567	1403.61x10^6	6285.41	279.95
	Second	6	3739	162.60x10^6	563.72	26.35
	Overall	**100**	**62306**	**1566.21x10^6**	**6849.13**	**306.30**
Theoretical Saving due to Early Exit				**1143.41x10^6**	**2544.74**	**132.86**
Average Saving per Segment due to Early Exit				**18351.52**	**40.84 (ms)**	**2132.37 (μJ)**

as the baseline architecture as it shows better performance of the two. Figure 1a shows that around 94% of the segments can be correctly classified by the FOB. Therefore, further convolution of those segments would be redundant. If we can avoid the redundant convolution operations, we can easily save some inference time and energy for the wearable devices. Table 1 shows the theoretical breakdown of the performance, FLOP counts, execution time, and energy of the early exit architecture to classify 62306 segments. Table 1 demonstrates that using early exit architecture, it is possible to save a total of 1143.41x10^6 FLOPs, 2544.74 seconds of execution time, and 132.86 J of energy for 62306 segments. On average for each segment, we can save 18351.52 FLOPs, 40.84 ms of execution time, and 2132.37 μJ of energy using an early exit architecture compared to the baseline one. In summary, such early exit architecture would provide a much more energy-efficient solution than a baseline architecture while providing better or similar performance that is suitable for low-power wearable devices.

1.2 Problem Statement

Many state-of-the-art works [6, 19, 30, 32, 36] have explored this early exit strategy for various applications. Usually, an extra output layer is added after each convolutional layer and a decision function is used to make the exit decisions for such early exit architectures. Some works [19, 30, 32, 36] use a simple decision function based on classification confidence[2]. If the classification confidence at the output layer is above a certain threshold,

[2]The classification confidence refers to the probability distribution of the classes at the output layer

the decision function makes the exit decision. Generally, this kind of decision function is extremely lightweight, energy-efficient, and suitable for wearable devices. However, a substantial number of misclassified segments can cause a significant performance degradation compared to the baseline, given how the classification confidence at the earlier layers can be misleading, as will be demonstrated later in Section 5.2. Authors in [6] proposed to overcome this problem by using another machine learning model as the decision function. The use of the machine learning models as the decision function is suitable for deep architectures (with tens to hundreds of layers) for the projected computational gains. However, it is not suitable for wearable devices as the decision function's computational overhead may be comparable to the compact baseline classifier of the wearable devices. Moreover, state-of-the-art early exit strategies follow a brute force approach in which the decision function is invoked for every potential exit point that succeeds each convolution layer. Such needless invocations may lead to extra computational overheads if the majority of the input data segments eventually require to be processed by the baseline model. The aforementioned limitations prevent the state-of-the-art early exit strategies to be adopted for wearable device solutions for MI detection. Therefore, in this paper, we introduce an output block selector (OBS) as our decision function to address these limitations, where it performs template matching using the simple Pearson correlation of a heartbeat segment against a template heartbeat. The template matching coefficient is used to select the output block, thus avoiding the brute force approach. Additionally, the same template matching coefficient is used for the exit decision-making criteria alongside the classification confidence to minimize the performance loss. What's more, the OBS is also lightweight enough to be adopted for wearable devices. To the best of our knowledge, we are the first to consider such a template matching based early exit (TMEX) architecture for MI detection that provides energy efficiency while ensuring similar performance as the baseline.

1.3 Novel Contributions

The novel contributions of this paper are as follows:

- We propose an MI detection methodology to implement a baseline CNN architecture that outperforms the state-of-the-art works while being energy and memory-efficient.
- We introduce a novel Template Matching based Early Exit (TMEX) CNN architecture that further increases the energy efficiency of the baseline architecture while maintaining similar performance.
- Evaluation of our methodology on two well known datasets - PTB [7] and PTB-XL [43] from PhysioNet [17]. It shows that both the baseline and TMEX architectures outperform related works for both datasets.
- Evaluations on real hardware show that our baseline architecture achieves from 0.6× to 53× energy efficiency compared to the state-of-the-art work on wearable devices. Moreover, our TMEX architecture further improves the energy efficiency over the baseline by 8.12% (PTB) and 6.36% (PTB-XL) while maintaining similar performance.
- Hardware evaluations also demonstrate that our baseline and TMEX architectures are compatible with the low-memory requirement of wearable devices requiring only 20 KB of RAM.

2 RELATED WORKS

2.1 MI Detection Using Single Lead ECG

As our proposed solution is targeted for wearable devices using only one lead ECG data, we will discuss and compare against the works that used only one lead for MI detection. Authors in [4, 37, 38] used machine learning algorithms K-Nearest Neighbor (k-NN), Support Vector Machine (SVM), Random Forest (RF) for MI detection. All of these works applied 4-level Discrete Wavelet Transform (DWT) using Daubechies 6 (db6) basis function on the heartbeat segments which results in four detail and four approximation coefficients. Then for each of those 8 coefficients, they extracted normalized energy, Higuchi's fractal dimension, along with the following entropies: approximate, fuzzy, permutation, wavelet, Shannon, Renyi, and Tsalis which results in a total of 72

features. Then they applied an infinite latent feature selection algorithm [35] to sort those 72 features based on their relevance. Authors in [4] used first 47 of those features and achieved a very high performance (Accuracy = 98.80%, Sensitivity = 99.45%, Specificity = 96.27%) using k-NN. However, their work is intended for a clinical setup. Moreover, k-NN is not suitable for wearable device solutions as it requires all the training data to be stored locally. Authors in [37] used an event-driven 2-level SVM classifier to achieve energy efficiency on wearable devices. For the first level SVM, they only used the first 5 of those 72 features and for the second (full) level they used the first 47 features similar to [4]. Their event-driven 2-level SVM and full SVM achieved an accuracy of 90% and 95%, respectively. Authors in [38] also proposed an event-driven technique using a 5-level RF classifier where the first, second, third, and fourth levels use the first 5, 10, 15, and 20 features out of those 72 features, respectively. And the fifth/full level uses all of those 72 features. Their event-driven technique achieved relatively poor performance (accuracy = 80.32%, sensitivity = 81.02%, specificity = 79.63%) whereas the full RF achieved a slightly better performance (accuracy = 83.26%, sensitivity = 87.95%, specificity = 78.82%).

The works in [37, 38] applied an event-driven technique to reduce the classifier complexity for energy efficiency. However, they along with [4], used extensive feature engineering to find optimal features for their classifiers which is a very difficult [42] task. Moreover, the feature extraction process is very expensive in terms of time and energy, making these methods unsuitable for real-time MI detection on wearable devices. In this scenario, deep learning algorithms like Convolutional Neural Networks (CNNs) [22] are a better alternative as they perform classification by automatically extracting features through convolution. Authors in [3] used 1-D deep CNN architecture with 4 convolutional layers, 4 max-pooling layers, and 3 fully connected layers. As their solution is intended for clinical setup, they only focused on performance without any resource constraints. Their method achieved an accuracy of 95.22%, a sensitivity of 95.49%, and specificity of 94.19%. Such a network is not suitable for wearable devices as it does not consider the energy and memory constraints. On the other hand, the authors in [33] developed a wearable device solution prioritizing energy and memory efficiency while sacrificing performance. Hence, they used a Binary Convolutional Neural Network (BCNN) with only 3 layers which achieved an accuracy of 90.29%, the sensitivity of 90.41%, and specificity of 90.16%. Another work in [30] used a CNN with an early exit strategy to develop a wearable device solution for MI detection. They applied neural architecture search to find an optimal network suitable for the wearable device while considering performance, energy efficiency, and memory efficiency as design objectives. Their baseline architecture achieved better performance (accuracy = 98.03%, sensitivity = 97.26%, specificity = 98.82%) than other wearable device solutions. Interestingly, their architecture with early exit strategy achieved a slightly better performance (accuracy = 98.54%, sensitivity = 97.66%, specificity = 99.44%) than the baseline, which may be attributed to the fact that the earlier exit's accuracy had a similar performance to the baseline one. Unfortunately, the details of their implemented architectures are not mentioned. However, their exit decision-making is based on the classification confidence only, potentially leading to a deterioration in the overall performance when there is a considerable performance mismatch between the earlier layer's exit and the baseline's one, as will be detailed later in Section 5.1. In another work, the authors in [27] applied a deep LSTM architecture to detect MI using single lead (II) ECG data from the recently introduced PTB-XL dataset [43]. They achieved an accuracy, sensitivity and specificity of 84.17% ,78.37%, and 87.55%, respectively. However, their sizable architecture is not suitable for resource constrained wearable devices. In this paper, we consider three metrics (performance, energy, memory) when designing a solution for wearable devices, and accordingly, we propose a methodology for implementing a CNN architecture that is energy- and memory-efficient while providing the maximum performance possible.

2.2 Energy-efficient CNN Design Approaches

Originally designed for computer vision applications, CNN has been widely adopted in other applications such as natural language processing, biomedical applications. Sometimes the deep learning nature of CNN allows

the architecture size to grow extremely large having 100s of convolutional, pooling, and fully connected layers. However, such an architecture is very computationally expensive and not suitable for energy and memory constraint applications of mobile health such as wearable devices. Therefore, many researchers have been working on different approaches to reduce the architecture size to make it energy and memory-efficient while maintaining similar or competitive performance. Such approaches can be broadly classified into two categories - 1) Software-based approach, 2) Hardware-based approach.

Software-based approaches mainly focus on minimizing the network size or developing techniques to satisfy the energy or memory constraints while maintaining performance. The software-based approaches can be further classified into 2 phases - 1) Offline or training phase, 2) Online or inference phase. The software-based approaches in the training phase can be broadly divided into 3 types - a) Neural Architecture Search (NAS), b) Network Pruning, c) Model Compression. NAS involves automatically finding the optimum network parameters from a search space using reinforcement learning [25, 47] or gradient-based methods [24] or multi-objective bayesian optimization [9, 15]. Network pruning involves random pruning of a portion of the big network, retraining/fine-tuning it, and repeat the process until it achieves the desired performance [8, 13, 16]. Finally, model compression involves binarization [12] or quantization [14, 44] of network weights to reduce the model size to make it memory-efficient. Whatever the methods are applied, the final model is considered as the baseline classifier to be used in the inference phase.

The software-based approach in the inference phase includes dynamic network pruning [29], slimmable neural network [46], dynamic quantization [41], and early exit strategies [6, 19, 32, 36]. The dynamic network pruning [29] approach prunes the baseline network (weights/neurons) during the inference phase. Unlike pruning during the training phase, dynamic pruning does not perform retraining/fine-tuning after the pruning step and the network may suffer from performance loss. The slimmable neural network [46] on the other hand uses the same network but with a reduced number of filter kernels or active channels during the inference phase. This is similar to the dynamic pruning of neurons. In practice, Both dynamic pruning and slimmable neural network are suitable for large network architectures with redundant weights/neurons that can be pruned without a substantial performance degradation. However, due to energy and memory constraints, such large architectures (with redundant weights/neurons) are not suitable for wearable devices in the first place, that is, wearable devices require efficient and compact architectures with minimal/no redundant weights/neurons to whom the application of dynamic pruning or slimming may degrade performance considerably. This could have been tackled by fine-tuning/ retraining the network in the inference phase, however, this will incur significant computation overhead for the wearable device which may overshadow the benefits achieved from it. On the other hand, dynamic quantization [41] involves the quantization of the network (weights and activations) in the inference phase. Similarly, the weights are not retrained after quantization, potentially exacerbating performance losses due to quantization errors. The early exit strategies [6, 19, 32, 36] leverage the layered architecture of the neural network and introduce multiple exit (output) layers in the network. However, as mentioned in Section 1.2, the state-of-the-art early exit strategies have inherent limitations which prevent them from being adopted for wearable device solutions. Our work addresses those limitations by introducing template matching based early exit (TMEX) architecture. It implements an output block selector as the decision function that addresses the limitations of state-of-the-art early exit strategies by using simple and yet effective template matching by Pearson correlation coefficient.

It is to note that, the software-based approaches for the inference phase are independent of each other and may also be applied together. For example, a model can be dynamically quantized and pruned and then an early exit strategy can be adopted at the same time. As our proposed baseline architecture is designed for wearable devices and already compact, further pruning and quantization during inference phase leads to performance loss as will be shown later in 5.3. Therefore, in this paper we implement our TMEX architecture directly on the baseline architecture.

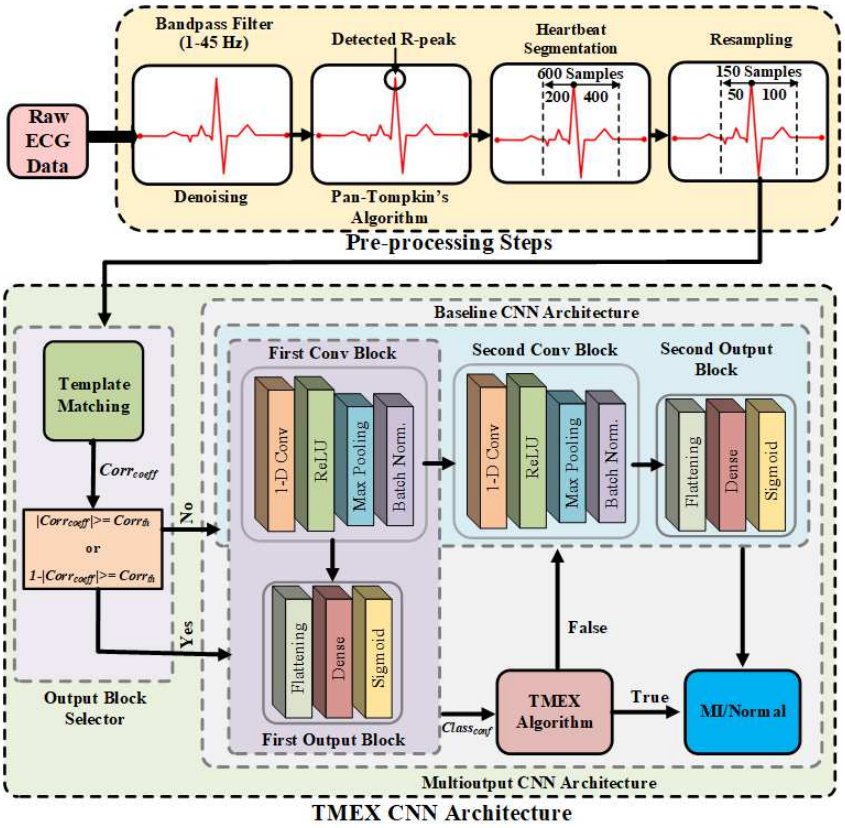

Fig. 2. Overview of Our Proposed Methodology. Baseline CNN architecture refers to the full architecture with no dynamic model compression or early exit techniques applied to it.

On the other hand, hardware-based approaches focus on the design of custom hardware such as accelerators which are specifically designed for CNNs [10, 11]. These accelerators facilitate speeding up the inference process and thereby making it more energy-efficient. Both Software-based and Hardware-based approaches are independent of each other and can be either applied separately or combined in a HW/SW co-design-based approach. However, Software-based approaches are more commonly adopted as they can be applied to all commercially available computing platforms (CPUs, GPUs, MCUs) and do not require any customized hardware.

3 PROPOSED METHODOLOGY

The following sections provide the details of our proposed methodology. The overview of our proposed methodology is demonstrated in Figure 2.

3.1 Pre-processing Steps

3.1.1 Filtering. As shown in Figure 2, the pre-processing step starts with the denoising of raw ECG data using a tenth-order Butterworth bandpass filter with cut-off frequencies f_1=1Hz and f_2=45Hz.

3.1.2 R-peak Detection. Once denoised, Pan-Tompkin's algorithm [31] is used to detect the R-peaks from the filtered ECG data.

3.1.3 Segmentation. Given a frequency value f, we take $f/5$ samples before and $2f/5$ samples after the R-peak. Thus, PTB dataset (f=1KHz) each segment consists of 600 samples and for PTB-XL dataset (f=500Hz) each segment consists of 300 samples representing a heartbeat.

3.1.4 Resampling. This step contributes a lot to the energy efficiency of our solution by reducing the number of samples to be processed by the CNN architecture for each segment. Therefore, it is important to carefully determine how much reduction is possible without compromising the performance. We determine the important frequencies from the power spectrum of the heartbeat segment. As shown in Figure 3, all the necessary frequency components are present within 125 Hz of the signal which means a sampling frequency of 250 Hz is good enough based on Nyquist theorem [21]. This is one-fourth of the sampling frequency used in the PTB dataset (1 kHz) and half of the sampling frequency used in the PTB-XL dataset (500 Hz). Therefore, we can resample each PTB heartbeat segments by one-fourth and PTB-XL segments by half. Thus, the resampled segments contain 150 samples. The resampling is done by applying an anti-aliasing low pass filter to each segment using Kaiser window method. Then the segments are downsampled by 4 times. We use the resample function available in MATLAB to perform this operation.

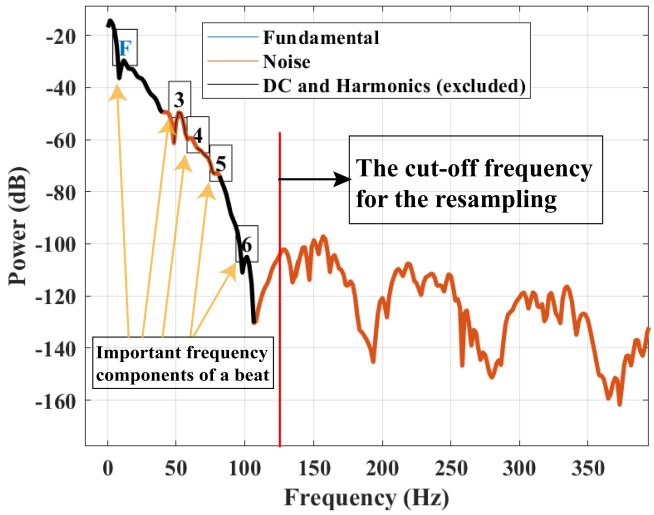

Fig. 3. The Power Spectrum of Filtered ECG Signal

3.2 TMEX CNN Architecture

As shown in Figure 2, our TMEX CNN architecture consists of three parts - 1) Multi-output CNN architecture that classifies the heartbeat segments, 2) Output block selector that decides which output block to start with based on the correlation of each heartbeat segment against the template beat, 3) TMEX algorithm that uses the correlation coefficient from the output block selector as an additional exit condition along with the classification confidence of the FOB.

Table 2. Multi-output CNN Architecture Details

Layer Name	Kernel Size	Stride Size	Activation Function	Output Shape	# of Param.
Input	-	-	-	150x1	0
Conv 1	31	1	ReLU	120x3	96
Pooling 1	2	2	-	60x3	0
Batch Norm.	-	-	-	60x3	12
Flatten 1	-	-	-	180	0
Dense 1	-	-	Sigmoid	1	181
Conv 2	7	1	ReLU	54x8	176
Pooling 2	2	2	-	27x8	0
Batch Norm	-	-	-	27x8	32
Flatten 2	-	-	-	216	0
Dense 2	-	-	Sigmoid	1	217
Total Number of Parameters after First Output Block					**289**
Total Number of Parameters after Second Output Block					**533**

3.2.1 Multi-output CNN Architecture. The multi-output CNN architecture is designed considering the resource constraints of the wearable devices. It consists of 2 convolution blocks and 2 output blocks where each convolution block is followed by one output block. Figure 2 shows the architecture layout of our multi-output CNN architecture. Each convolution block consists of one convolution layer which is passed through ReLU activation, one max-pooling layer, and one batch normalization layer. Each output block consists of one flattening layer, and one dense layer which is passed through sigmoid activation. The details of the architecture parameters for each of the layers are given in Table 2. As shown in Table 2, the total number of parameters required to classify a heartbeat segment after the first, and second output block is 289, and 533, respectively.The two convolution blocks along with the second output block are considered as the baseline architecture in this paper.

3.2.2 Output Block Selector. The Output Block Selector (OBS) is used to avoid the brute force method followed by traditional early exit CNN architecture. It implements the template matching mechanism using the Pearson correlation coefficient to determine whether we should try to exit FOB or not. We create a template beat from the average of 10 random MI heartbeat segments. It is to be noted that one can use a different number of heartbeat segments to create the template beat. However, using very few heartbeats (like 2 or 3) may not generalize the MI heartbeat template and may get biased to specific patients. Similarly, using too many heartbeat segments (100 or 200) might lose the MI pattern in the template beat. We also tried with 20 and 30 random beats to calculate the template beat. However, found no significant change in the performance of our TMEX architecture. Therefore, we decided to use the 10 random beats as it generalizes well and at the same time maintains the MI pattern on the template beat. If the absolute value of the Pearson correlation coefficient [5] between a segment and the template beat is greater than the correlation threshold, $Corr_{th}$, that means the segment is less complex and have a higher chance to be correctly classified by the FOB. Therefore, it selects the FOB to classify the segment. Otherwise, it directly uses the baseline architecture to classify a particular segment. It is to note that, even if the OBS decides to use the FOB, it does not guarantee early exit. The early exit decision is made by the TMEX algorithm as discussed in Section 3.2.3.

Algorithm 1: TMEX Algorithm

Input: $Corr_{coeff}$: Pearson Correlation Coefficient from the OBS
Input: $Class_{conf}$: Classification confidence of the FOB
Output: $Exit_{flag}$: Early exit decision
Constant Variables:
$Corr_{th}$: Correlation Threshold used by OBS
$Conf_{th}$: Confidence Threshold for FOB
`// For MI segments`
if $|Corr_{coeff}| \geq Corr_{th}$ *and* $Class_{conf} \geq Conf_{th}$ **then**
  $Exit_{flag}$=True
end
`// For Normal segments`
else if $1 - |Corr_{coeff}| \geq Corr_{th}$ *and* $1 - Class_{conf} \geq Conf_{th}$ **then**
  $Exit_{flag}$=True
end
`// To move to next convolution block`
else
  $Exit_{flag}$=False
end
return $Exit_{flag}$

3.2.3 TMEX Algorithm. TMEX algorithm is used to make the final decision of early exit after FOB. Algorithm 1 shows the stepwise procedure followed to make the exit decision. As the template beat is created from MI heartbeats, $1 - |Corr_{coeff}|$ represents the correlation coefficient for the normal heartbeats. Similarly, as we are using the sigmoid function at the output block, $Class_{conf}$ represents the classification confidence for MI segments. Therefore, $1 - Class_{conf}$ represents the classification confidence for normal segments. If both correlation co-efficient and classification confidence for either MI or normal segments are greater than the corresponding thresholds, the algorithm decides to exit early. Otherwise, it proceeds to the next convolution block. The use of both the correlation coefficient and the confidence thresholds allows us to overcome the limitations of traditional early exit architecture when the exit decision is solely based on classification confidence. For example, if FOB misclassifies a segment with high confidence it will exit after the FOB.

4 EXPERIMENTAL SETUP

4.1 Database Used

We use two well known datasets – PTB diagnostic ECG database [7] and PTB-XL datastet [43] from PhysioNet [17]. The PTB database contains MI data from 148 subjects and normal healthy control data from 52 subjects whereas the PTB-XL contains MI data from 5486 patients and healthy data from 9528 normal subjects. Each record includes 15 simultaneously measured signals: the conventional 12 leads (i, ii, iii, avr, avl, avf, v1, v2, v3, v4, v5, v6) together with the 3 Frank lead ECGs (vx, vy, vz). As our work is intended for the wearable device we use the single lead ECG data. We use the 11^{th} lead (v5) from PTB dataset and 2^{nd} lead (II) from PTB-XL dataset, to ensure fair comparison with the related works on the corresponding datasets. The signal in PTB and PTB-XL dataset is digitized at 1000 and 500 samples per second respectively. Table 6 shows the summary of data distribution and the specific lead used in each of the related works.

4.2 Performance Metric

As the number of segments for different classes in both the dataset is highly imbalanced, only classification accuracy is not appropriate to measure performance. We use both the sensitivity and specificity metric to ensure a fair comparison with our related works as shown below:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3}$$

Where TP, TN, FP, FN represents True Positives, True Negatives, False Positives, and False Negatives respectively. Sensitivity represents the true positive rate that measures the portion of the positive class (MI segments) that is correctly classified. Similarly, specificity represents the true negative rate that measures the portion of the negative class (Normal segments) that is classified correctly.

4.3 Training Multi-output CNN Classifier

To validate the performance of our Multi-output CNN classifier, we use data of 200 subjects (52 Normal, 148 MI) from PTB [7]. A total of 62306 (51880 MI, 10426 Normal) heartbeat segments are obtained after pre-processing steps from Lead 11 ECG data. The PTB dataset is highly imbalanced since the number of MI segments is approximately 5 times the normal segments. For the PTB-XL dataset data from 15014 subjects (9528 Normal, 5486 MI). Unlike PTB, PTB-XL provides a specific distribution of train, validation, test subjects to ensure a fair comparison of different related works. Thus, training and test data comes from completely different subjects. After pre-processing steps, we obtained 136136 (83765 Normal, 52371 MI) train segments, 17186 (10536 Normal, 6650 MI) validation segments, and 17319 (10575 Normal, 6744 MI) test segments. The PTB-XL dataset is also imbalanced but towards normal segments. To ensure proper training on the imbalance dataset, we assign class weights to each class during training using the following formula in Eq. 4.

$$w_i = \frac{1}{N_i} * \frac{N}{n_c} \tag{4}$$

Here, w_i, and N_i represent the class weight and the number of segments belonging to class i, respectively. N is the total number of segments from all classes and n_c is the number of output classes which is 2 in our case.

We train the classifier with a batch size of 500. The models are trained for 150 epochs and select the model with minimum validation loss as the best one from those epochs. We use *Binary Crossentropy* as the loss function for each output block. *Adam* optimizer is used to train the models with a learning rate of .001. Similar to the related works in PTB, we also perform stratified 5-fold cross-validation on the total data where 80% data (4 folds) is used for training and 20% is used for testing. Moreover, 20% of the training data is used validation during training. Stratified 5-fold cross-validation ensures each fold contains a similar distribution of data from each class. For the PTB-XL dataset, the model was trained, validated, and tested using the provided distribution of train, validation, and test subjects data.

4.4 Target Wearable Device

Our work is designed for low-power, low-memory wearable devices like SmartCardia [2]. For example, SmartCardia INYU [28] device is equipped with an ultra-low-power 32-bit microcontroller STM32L151 containing an ARM Cortex–M3 with a maximum clock rate of 32 MHz. It has 48 KB RAM, 384 KB Flash, and a standard 710

Table 3. Correlation and Confidence Threshold Analysis of TMEX Architecture on PTB Dataset

Work	OBS	FOB	Performance (%)			Output Block		Time	Pwr.	Energy	Energy
	$Corr_{th}$	$Conf_{th}$	Acc.	Sen.	Spec.	First	Second	(ms)	(mW)	(μJ)	Saving (%)
Baseline	–	–	99.33	99.25	99.74	0	62306	150.77	46.75	7048.49	0
Early Exit	–	0.5	94.82	94.22	97.83	62306	0	110.50	44.76	4945.98	29.83
		0.6	95.09	94.51	98.01	61860	446	110.81	44.77	4960.78	29.62
		0.7	95.37	94.80	98.23	61379	927	111.15	44.78	4976.74	29.39
		0.8	95.68	95.15	98.33	60691	1615	111.63	44.79	4999.59	29.07
		0.9	96.12	95.65	98.47	59434	2872	112.50	44.81	5041.36	28.48
Template Matching Based Early Exit (TMEX)	0.5	0.5	98.85	99.06	97.78	39740	22566	126.24	45.16	5700.88	19.12
		0.6	98.88	99.06	97.99	39494	22812	126.41	45.16	5709.18	19.00
		0.7	98.92	99.07	98.21	39238	23068	126.59	45.17	5717.81	18.88
		0.8	98.96	99.09	98.31	38855	23451	126.85	45.18	5730.74	18.70
		0.9	99.00	99.11	98.45	38139	24167	127.35	45.19	5754.91	18.35
	0.6	0.5	98.89	99.11	97.78	36097	26209	128.78	45.22	5823.93	17.37
		0.6	98.92	99.11	97.99	35866	26440	128.94	45.23	5831.74	17.26
		0.7	98.96	99.11	98.21	35618	26688	129.11	45.23	5840.13	17.14
		0.8	98.99	99.12	98.33	35264	27042	129.36	45.24	5852.11	16.97
		0.9	99.03	99.14	98.47	34598	27708	129.82	45.25	5874.65	16.65
	0.7	0.5	99.06	99.15	98.60	30006	32300	133.02	45.33	6030.40	14.44
		0.6	99.08	99.14	98.77	29814	32492	133.16	45.34	6036.92	14.35
		0.7	99.11	99.15	98.94	29607	32699	133.30	45.34	6043.95	14.25
		0.8	99.14	99.15	99.05	29304	33002	133.51	45.35	6054.25	14.11
		0.9	99.16	99.16	99.16	28726	33580	133.92	45.36	6073.90	13.83
	0.8	0.5	99.20	99.17	99.33	17723	44583	141.59	45.55	6449.55	8.50
		0.6	99.20	99.16	99.40	17621	44685	141.66	45.55	6453.05	8.45
		0.7	99.21	99.16	99.44	17509	44797	141.74	45.55	6456.88	8.39
		0.8	99.23	99.17	99.52	17326	44980	141.87	45.56	6463.16	8.30
		0.9	**99.24**	**99.18**	**99.55**	**16952**	**45354**	**142.13**	**45.56**	**6475.98**	**8.12**
	0.9	0.5	99.22	99.13	99.71	1260	61046	153.07	45.84	7017.20	0.44
		0.6	99.22	99.13	99.71	1256	61050	153.07	45.84	7017.33	0.44
		0.7	99.23	99.13	99.72	1251	61055	153.08	45.84	7017.51	0.44
		0.8	99.25	99.15	99.74	1241	61065	153.08	45.84	7017.85	0.43
		0.9	99.26	99.16	99.74	1229	61077	153.09	45.84	7018.27	0.43

mAh battery. The device captures ECG signals using a single lead ECG sensor [28]. Our solution applies to any wearable device having the above or similar specifications.

5 EXPERIMENTAL RESULTS AND ANALYSIS

5.1 Correlation and Confidence Threshold Analysis of TMEX Architecture

The correlation threshold, $Corr_{th}$ of the output block selector (OBS), and the confidence threshold, $Conf_{th}$ of the first output block (FOB) play the most important role in our TMEX architecture. Therefore, we conduct a detailed analysis of the different combinations of these two thresholds and their impact on the early exit decision,

Table 4. Correlation and Confidence Threshold Analysis of TMEX Architecture on PTB-XL Test Data

Work	OBS	FOB	Performance (%)			Output Block		Time	Pwr.	Energy	Energy
	$Corr_{th}$	$Conf_{th}$	Acc.	Sen.	Spec.	First	Second	(ms)	(mW)	(μJ)	Saving (%)
Baseline	–	–	84.36	78.60	88.03	0	17319	150.77	46.75	7048.49	0
Early Exit	–	0.5	84.12	74.02	90.55	17319	0	110.50	44.76	4945.98	29.83
		0.6	84.15	74.32	90.42	17108	211	111.03	44.77	4971.17	29.47
		0.7	84.18	74.57	90.32	16868	451	111.63	44.79	4999.84	29.07
		0.8	84.16	74.96	90.03	16557	762	112.41	44.81	5037.01	28.54
		0.9	84.05	75.37	89.59	16147	1172	113.44	44.83	5086.07	27.84
Template Matching Based Early Exit (TMEX)	0.5	0.5	84.08	74.87	89.95	12002	5317	123.84	45.10	5585.06	20.76
		0.6	84.05	74.97	89.84	11904	5415	124.09	45.11	5596.92	20.59
		0.7	84.09	75.22	89.75	11788	5531	124.38	45.11	5610.97	20.39
		0.8	84.10	75.61	89.52	11617	5702	124.81	45.12	5631.69	20.10
		0.9	84.05	76.11	89.11	11380	5939	125.40	45.14	5660.41	19.69
	0.6	0.5	84.16	75.43	89.72	9854	7465	129.23	45.24	5845.81	17.06
		0.6	84.11	75.53	89.58	9776	7543	129.42	45.24	5855.30	16.93
		0.7	84.11	75.77	89.43	9685	7634	129.65	45.25	5866.38	16.77
		0.8	84.06	76.11	89.13	9552	7767	129.99	45.26	5882.58	16.54
		0.9	84.02	76.62	88.75	9355	7964	130.48	45.27	5906.59	16.20
	0.7	0.5	84.21	75.98	89.47	6930	10389	136.56	45.42	6203.12	11.99
		0.6	84.17	76.08	89.33	6881	10438	136.69	45.43	6209.14	11.91
		0.7	84.16	76.29	89.18	6811	10508	136.86	45.43	6217.72	11.79
		0.8	84.07	76.53	88.88	6710	10609	137.12	45.44	6230.12	11.61
		0.9	84.05	77.00	88.55	6570	10749	137.47	45.45	6247.31	11.37
	0.8	0.5	84.50	76.81	89.41	3948	13371	144.05	45.61	6570.35	6.78
		0.6	84.47	76.93	89.28	3916	13403	144.13	45.62	6574.31	6.73
		0.7	84.42	77.05	89.13	3869	13450	144.24	45.62	6580.12	6.64
		0.8	84.31	77.19	88.84	3802	13517	144.41	45.62	6588.40	6.53
		0.9	**84.24**	**77.46**	**88.56**	**3707**	**13612**	**144.65**	**45.63**	**6600.15**	**6.36**
	0.9	0.5	84.55	77.65	88.95	1430	15889	150.36	45.77	6882.65	2.35
		0.6	84.54	77.73	88.88	1414	15905	150.40	45.77	6884.64	2.32
		0.7	84.51	77.77	88.80	1395	15924	150.45	45.78	6887.01	2.29
		0.8	84.40	77.85	88.58	1354	15965	150.55	45.78	6892.11	2.22
		0.9	84.30	77.91	88.38	1308	16011	150.67	45.78	6897.84	2.14

performance and energy efficiency. For the second output block, we always use the confidence threshold of 0.5 as it is the last output block in our architecture. Tables 3 and 4 show the performance and the different output blocks used to classify the segments of PTB and PTB-XL test dataset, respectively. We have also analysed the execution time, power, energy, and associated energy saving achieved by the TMEX architecture for different $Corr_{th}$ and $Conf_{th}$. The details of the energy calculation is provided later in Section 5.6. As shown in Tables 3 and 4, the increasing value of $Corr_{th}$ increase the performance while increasing the energy consumption which in turn reduces the amount of energy saved by the TMEX architecture. This is because the OBS selects more segments to be classified by the second output block which has better performance but requires more energy. For

each $Corr_{th}$ value, the same happens with the increase of the $Conf_{th}$ value. The performance increases at the cost of energy. This happens as the TMEX algorithm does not allow early exit after FOB as it requires higher classification confidence. Rather they are sent to the second output block which has better performance and higher energy requirement. This is why the number of segments classified by FOB decreases which increases the count for second output blocks. Tables 3 and 4 show that, for $Corr_{th}$=0.5 and $Conf_{th}$=0.5, our TMEX archiecture achieves the lowest performance with the highest energy saving of 19.12% and 20.76% energy efficiency for the PTB and PTB-XL test dataset, respectively. Conversely, for $Corr_{th}$=0.9 and $Conf_{th}$=0.9, it achieves the highest performance with lowest energy efficiency of 0.43% and 2.14% on the respective datasets. This proves that there is a trade off between performance and energy efficiency for different values of correlation and confidence thresholds. Therefore, our goal is to find out the best combination of these two thresholds that maintains a similar performance of the TMEX architecture as the baseline one while providing descent energy efficiency. As shown in Tables 3 and 4, $Corr_{th}$ = 0.8 and $Conf_{th}$ = 0.9 achieves the best performance considering all three metrics while maintaining a decent energy efficiency of 8.12% and 6.36% for the respective datasets. Therefore, we consider these two values for the correlation threshold of OBS and the confidence threshold of FOB in our TMEX architecture. However, TMEX architecture provides flexibility to achieve more energy efficiency at the cost of performance by choosing lower threshold values.

5.2 Performance Analysis of the Traditional Early Exit

To demonstrate the importance of the template matching based early exit architecture, we analyze the performance of the traditional early exit where exit decision is made solely based on the FOB confidence threshold. Tables 3 and 4 show the performance of traditional early exit architecture for different confidence thresholds ($Conf_{th}$ = 0.5 to 0.9). As the tables show, the performance of traditional early exit increases with the increasing value confidence thresholds from 0.5 to 0.9. For the PTB dataset in Table 3, the best performance of traditional early exit architecture (for $Conf_{th}$ = 0.9) is still lower than that of the lowest performance of our TMEX architecture (for $Corr_{th}$ = 0.5, $Conf_{th}$ = 0.5). And for the PTB-XL test dataset in Table 4, our TMEX architecture starting from $Corr_{th}$ = 0.5 and $Conf_{th}$ = 0.8 or any combination upto 0.9, outperforms traditional early exit architecture. However, traditional early exit architecture achieves higher energy efficiency while sacrificing the performance. They achieve upto 29.83% of energy saving compared to the 19.12% and 20.76% for our TMEX architecture. On the other hand, TMEX architecture ensures similar performance as the baseline while being as energy-efficient as possible.

5.3 Performance Analysis of the Other Dynamic Model Compression Techniques

This section evaluates the performance of other dynamic model compression techniques on our baseline architecture. As shown in Table 5, for all the dynamic compression techniques the performance of the baseline architecture decreases to some extent. For slimmable neural network and dynamic pruning we have applied compression on the weights/neurons of the two convolution layers and the dense layer was kept intact. This is because, compressing dense layer causes even more performance degradation. For slimmable neural network, we have considered two configurations. In the first configuration we have slimmed the lowest magnitude filter kernel from the first convolution layer thus using only 2 of the 3 filter kernels. Similarly, in the second configuration we have slimmed the lowest magnitude filter kernel from the second layer thus using 7 of the 8 filter kernels. For dynamic weight pruning, we have pruned 20% of the weights with lowest magnitude. And for dynamic neuron pruning we have pruned 20% of the neurons with lowest magnitude. As shown in the Table 5, the baseline performance drops significantly for both the slimming and pruning options. This justifies the fact that our baseline architecture is already compact and does not have much redundant weights/neurons to prune or slim. On the other hand, dynamic quantization shows a better performance compared to pruning or slimming, as it does not

Table 5. Performance Comparison with Dynamic Model Compression Techniques

Technique	Configuration	PTB			PTB-XL Test		
		Acc.	Sen.	Spec.	Acc.	Sen.	Spec.
Baseline [Ours]	–	**99.33**	**99.25**	**99.74**	**84.36**	**78.60**	**88.03**
TMEX [Ours]	$Corr_{th}$=0.8, $Conf_{th}$=0.9	**99.24**	**99.18**	**99.54**	**84.24**	**77.46**	**88.56**
Slimmable NN [46]	Filter: Conv1=2; Conv2=8	87.21	98.50	31.03	69.92	88.67	57.96
	Filter: Conv1=3; Conv2=7	95.47	99.25	76.61	66.44	93.29	49.31
Dynamic Pruning [29]	Weight Pruning (20%)	95.77	98.97	79.84	84.20	76.52	89.10
	Neuron Pruning (20%)	82.98	99.42	1.16	74.06	89.39	64.29
Dynamic Quantization[41]	int8	97.33	97.05	98.72	83.53	76.01	89.69
Trad. Early Exit [19, 32]	$Conf_{th}$=0.9	96.12	95.64	98.46	84.05	75.37	89.59

change the baseline architecture during inference. Rather, it just quantizes the network weights while keeping the architecture intact. Similarly, traditional early exit architecture also suffers from performance loss compared to baseline one. And TMEX architecture outperforms other techniques while maintaining similar performance as the baseline one. It is to note that, the dynamic model compression techniques are independent of each other and can be used simultaneously. Our TMEX architecture can be implemented on a quantized and pruned version of the baseline architecture. However, we chose to implement the TMEX architecture on the baseline directly as other compression techniques reduce the baseline performance significantly.

5.4 Performance Evaluation against Related Works on MI Detection

As shown in Tables 6 and 7, our baseline architecture outperforms the related works on MI detection for both PTB and PTB-XL datasets. For PTB dataset (Table 6), our baseline architecture achieves an accuracy of **99.33%**, sensitivity of **99.25%**, and specificity of **99.74%**. Our TMEX architecture shows a similar performance as the baseline one with an accuracy, sensitivity, and specificity of 99.24%, 94.18%, 99.54%, respectively. Both baseline

Table 6. Performance Comparison of Related Works on PTB Dataset

Work	PTB Patient			Classifier Type	Performance (%)		
	Normal	MI	Lead		Accuracy	Sensitivity	Specificity
[4]	52	148	11	k-NN	98.80	99.45	96.27
[37]	52	52	11	Full SVM	95	–	–
				2-level SVM	90	–	–
[38]	52	52	11	Full RF	83.26	87.95	78.82
				5-level RF	80.32	81.02	79.63
[3]	52	148	2	CNN	95.22	95.49	94.19
[33]	52	148	11	BCNN	90.29	90.41	90.16
[30]	52	148	11	Baseline CNN	98.03	97.26	98.82
				Early Exit ($Conf_{th}$=0.99)	98.54	97.66	99.44
Ours	52	148	11	FOB CNN	94.82	94.22	97.83
				Baseline CNN	**99.33**	**99.25**	**99.74**
				TMEX ($Corr_{th}$=0.8, $Conf_{th}$=0.9)	**99.24**	**99.18**	**99.54**

Table 7. Performance Comparison of Related Works on PTB-XL Dataset

Work	PTB-XL Patient			Classifier Type	Performance (%)		
	Normal	MI	Lead		Accuracy	Sensitivity	Specificity
[27]	9528	5486	2	Deep LSTM	84.17	78.37	87.55
Ours	9528	5486	2	FOB CNN	84.12	74.02	90.55
				Baseline CNN	**84.36**	**78.60**	**88.03**
				TMEX ($Corr_{th}$=0.8, $Conf_{th}$=0.9)	**84.24**	**77.46**	**88.56**

and TMEX architectures significantly outperform the other state-of-the-art works [3, 4, 33, 37, 38] in almost all three metrics. Moreover, SVM and RF models, in general, are not suitable for wearable devices in terms of memory footprint which we will discuss in the next Section 5.5. The work [3] using deep CNN achieves comparatively better performance compared to other wearable device solutions [33, 37, 38]. However, our work still outperforms [3] which is designed for clinical setups. The work [4] achieves the highest performance among the related works with accuracy, sensitivity, and specificity of 98.80% and 99.45%, and 96.27% respectively. However, the k-NN classifier is only suitable for clinical set up as all the training data should be loaded in the memory during the inference phase. The solution in [30] is implemented for wearable devices and achieves a close performance to ours. However, our architecture is much more energy-efficient than [30] which is detailed later in Section 5.6. As shown in Table 7, our baseline architecture also outperforms [27] on PTB-XL dataset achieving an accuracy, sensitivity, and specificity of 84.36%, 78.60.36%, and 88.03%, respectively. Our TMEX architecture shows a similar performance as the baseline one with an accuracy, sensitivity, and specificity of 84.24%, 77.46%, 88.56%, respectively. Moreover, the work [27] uses deep LSTM network which is not suitable for wearable devices.

5.5 Memory Footprint Evaluation on Real Hardware

We evaluate the memory footprint of all works mentioned in Table 6 except for the work [4] that uses k-NN classifier. The k-NN classifier is not suitable for wearable devices as it requires all the training data to be loaded into the memory. For the machine learning approaches in [37, 38] the reported memory footprint is for the feature extraction and classification process. For the deep learning approaches using CNN [3, 33], we evaluate the memory footprint of the classification as it automatically extracts features during classification.

The work [37] and [38] uses 2-level SVM and a 5-level RF classifier respectively. Both approaches have three major drawbacks in terms of memory. ***First***, the model size of the SVM and RF increases with the increasing amount of training data as well as the number of features. ***Second***, both of them need to have multiple models (2 for SVM and 5 for RF) loaded into memory. ***Third***, the feature extraction process for the full level SVM and RF classifier requires a huge amount of memory.

To demonstrate an example of the *first* drawback, we train the SVM and RF models with varying numbers of training samples. For SVM models, we train with both 5 (first level) and 47 (second/full) features. For RF models we train with 5 and 72 features corresponding to the first and fifth level classifiers in [38]. As shown in Figure 4, the size of the SVM model increases linearly with the increasing number of training samples. The model size also increases as the feature number increases from 5 to 47 for SVM. On the other hand, the RF model is trained with 100 weak learners as used in [38]. Moreover, they also used a variable-sized split for the RF where the tree grows until no further split is possible which also increases the model size exponentially. In Figure 4, we demonstrate the change of the RF model size for both variable split and a fixed split of 20. As shown in Figure 4, the size of the RF model for both variable and fixed split increases with the increasing number of training samples as well as the increase of features from 5 to 72. Figure 4 shows that the SVM, fixed split RF, and variable split RF model trained with 8000 samples will have a model size of 185, 281, 896 KBs, respectively for only 5 features. For the same 8000

samples, the SVM model size increases from 185 KB to 567 KB when the number of features increases from 5 to 47. The same holds for the RF model with fixed and variable split as the number of features increases from 5 to 72, the model size changes from 281 KB and 896 KB to 299 KB and 1290 KB, respectively. Thus, both of these approaches are not suitable for low-memory wearable devices.

Moreover, the work [37] requires 2 of the SVM models and the work [38] requires 5 of the RF models to be loaded into the memory which gives us the perspective of the *second* drawback. Besides, both the approaches in [37, 38] require a huge amount of memory for the feature extraction process of their full level classifiers which brings us to the *third* drawback. We demonstrate the *third* drawback using the RAM footprint for each level of the SVM and RF classifier on the EFM32 Giant Gecko microcontroller which has 128 KB of RAM and 1 MB of flash memory. As our goal is to evaluate the memory requirement for the feature extraction of different levels of SVM and RF, we train the SVM with only 20 training samples and RF with only 100 training samples so that they can fit within the 128 KB of RAM. Also, for the RF model, we use only 10 weak learners with a fixed split of 10. As shown in Table 8, both the full level classifier in [37] and [38] requires almost 83 MB of RAM making them incompatible for wearable devices with lower memory.

One of the advantages of deep learning approaches in [3, 27, 30, 33] over the machine learning ones in [37, 38] is that the classifier model size does not change with the number of training samples. However, the memory requirement of the deep learning models still changes with architecture size, parameters, and input segment size. For example, the 11 layers deep CNN architecture used in [3] requires 114 KB of RAM. Whereas the work in [33] focuses on low memory wearable devices and requires only 3.5 KB of RAM. The approach in [33] focused on memory and energy efficiency while sacrificing performance. Authors in [30] developed wearable device solution

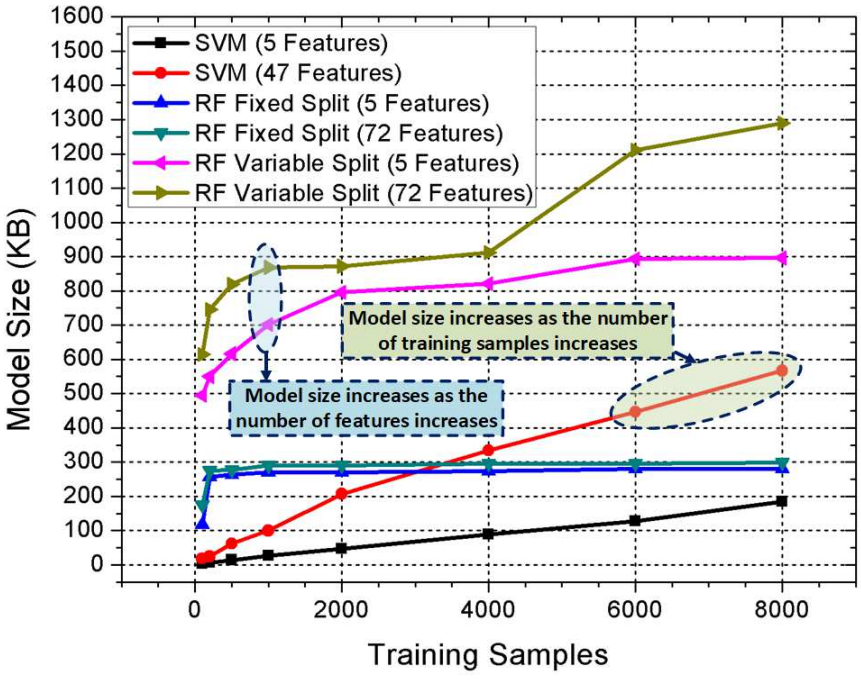

Fig. 4. Variation of Model Size with Increasing Training Samples and Features

Table 8. Memory Footprint and Energy Consumption Evaluation on EFM32 Giant Gecko Development Board

Works	Classifier Level	RAM Footprint (B)	Exe. Time (ms)	Avg. Power (mW)	Energy (μJ)
SVM[37]	First (5 Features)	78844	347.03	46.57	16161.18
	Full (47 Features)	Not Compatible: RAM Overflowed			
RF[38]	First (5 Features)	78844	345.35	46.57	16082.94
	Second (10 Features)	85948	3556.37	46.58	165655.71
	Third (15 Features)	87316	7669.7	46.81	359018.65
	Fourth (20 Features)	88132	8188.21	46.48	380588.00
	Full (72 Features)	Not Compatible: RAM Overflowed			
CNN[3]	-	114176	2036.82	46.97	95669.43
BCNN[33]	-	3568	253.73	44.45	11278.29
CNN[30]	Baseline	15972	–	–	28320
	Early Exit ($Conf_{th}$=0.99)	15972	–	–	28189
Deep LSTM[27]	–	Not Compatible: RAM Overflowed			
CNN[Ours]	OBS	2612	3.18	44.98	143.04
	FOB	11868	107.32	44.54	4780.03
	Baseline	**20160**	**150.77**	**46.75**	**7048.49**
	TMEX* (PTB)	**20160**	**142.13**	**45.56**	**6475.98**
	TMEX* (PTB-XL Test)	**20160**	**144.65**	**45.63**	**6600.15**

* *TMEX energy measurements for $Corr_{th}$=0.8, $Conf_{th}$=0.9. Measurements for other thresholds are presented in Tables 3 and 4*

with a RAM footprint of 15.97 KB. The deep LSTM architecture used in [27] has more than fourteen thousand parameters and encounters RAM overflow.

Table 8 shows the RAM footprint our baseline architecture is 20 KB. The OBS requires only 2.61 KB of RAM (which is less the 20 KB) without adding any extra memory overhead. The RAM footprint of our TMEX architecture is also 20 KB which is the maximum of RAM footprints of the output block selector and each of the output blocks. Thus, our proposed baseline and TMEX CNN architecture are compatible with any device with a minimum RAM of 32 KB.

5.6 Energy Consumption Evaluation on Real Hardware

Evaluation of energy consumption is conducted on the same EFM32 Giant Gecko microcontroller as used for memory evaluation. The energy measurement is done using the integrated energy profiler [20] of the Simplicity Studio IDE for the EFM32 boards. While measuring the energy, the board is set to active/run mode (EM0-Energy Mode 0) with a clock speed of 48 MHz. To ensure a fair comparison, the same configuration is used for measuring the energy of all the state-of-the-art work implementations. For the machine learning approaches [37, 38], the energy for the feature extraction and classification are calculated. For deep learning ones using CNN, we evaluate the classification energy only as they automatically extract features during classification. The energy for the TMEX architecture is calculated through the summation of the products between the ratios segments classified by each output block and the corresponding energy consumption for that block. For the energy evaluation of work [37], we train the SVM classifier with only 20 training samples to just fit the trained model in the memory. Similarly, the RF classifier in [38] is trained with only 100 training samples, 10 weak learners with a fixed split

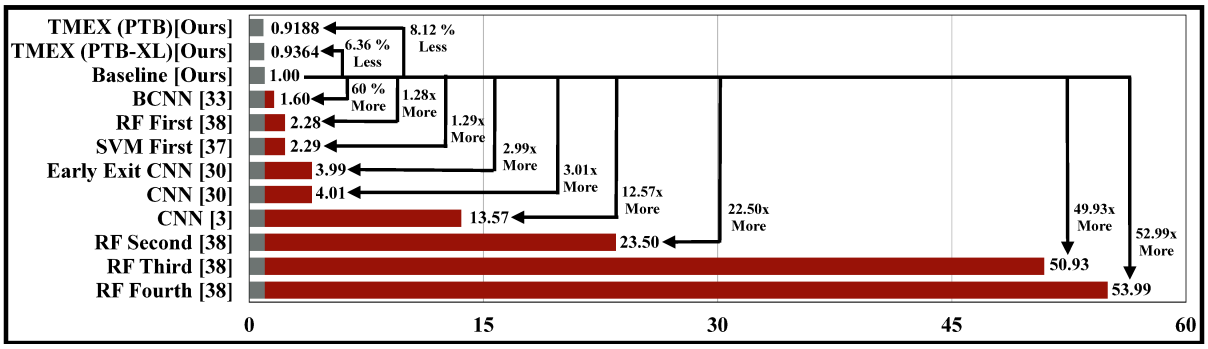

Fig. 5. Comparison of Energy Consumption w.r.t Baseline Classifier

of 10 just to fit the model into the RAM. The detailed analysis of execution time, power, and energy for one heartbeat segment are shown in Table 8.

Figure 5 shows a summary of the energy consumption of various works reported in Table 8. The results are normalized with respect to the energy consumption of our baseline CNN classifier. As shown in Figure 5, RF first, second, third, and fourth level classifiers in [38] consumes 1.28×, 22.50×, 49.93×, and 52.99× more energy compared to our baseline classifier. All the 20 features calculated in RF fourth level are the same as the first 20 features out of the 47 features in full SVM classifier of [37]. This indicates that the full SVM and RF classifiers will consume much more energy compared to our baseline classifier. Even the first level SVM classifier in [37] consumes 1.29× more energy compared to our baseline classifier. It is to be noted that we used a small number of training samples to train the SVM and RF models to keep the model size small. The energy consumption of SVM and RF classifiers, like the model size, will increase with the increase of training samples. The BCNN [33] which has the lowest energy consumption among the related works also consumes 60% more energy than our baseline classifier. The work [3] using CNN classifier consumes 12.57× more energy than the baseline one. The work [30] is designed for wearable devices and still consumes 3.01× more energy than ours. The early exit version of [30] also consumes 2.99× more energy than ours. It is to note that the early exit version of [30] does not provide any significant energy saving from its baseline. This is because they used a confidence threshold of 0.99 which probably causes most of the segments to be classified by the baseline architecture. That is why the performance of the early exit is also similar to the baseline (as shown in Table 6) as it hardly uses the early exit option. On the other hand, our TMEX architecture (for $Corr_{th}$=0.8, $Conf_{th}$=0.9) for PTB-XL and PTB dataset consumes 6.36% and 8.12% less energy compared to our baseline architecture as shown in Figure 5. Moreover, for lower correlation and confidence threshold values our TMEX architecture can achieve upto 19.12% (PTB) and 20.76% (PTB-XL) energy efficiency while sacrificing some performance (but still outperforming the traditional early exit strategy).

5.7 Ablation Study of the Segment Resampling

We perform an ablation study without resampling the segment for our baseline architecture to demonstrate the efficacy of the segment resampling in our proposed methodology. As mentioned in Section 3.1.4, before resampling there are 600 and 300 samples in a heartbeat segment for PTB and PTB-XL dataset, respectively. Therefore, we trained our baseline architecture as discussed in Section 3.2.1 with an input size of 600 and 300 samples instead of 150. As shown in Table 9 for PTB dataset, the accuracy, sensitivity, and specificity of our baseline architecture with resampling is very similar to that without resampling. This shows that resampling a segment from 600 samples to 150 does not cause any significant performance loss. Rather, resampling reduces the parameter and FLOP count of the baseline architecture by 2.68× and 4.89×, respectively as shown in Table 9.

Table 9. Performance and Resource Consumption Analysis of the Baseline Architecture W/O Resampling

Dataset	Baseline	Performance (%)			Param Count	FLOP Count	Time (ms)	Energy (μJ)	RAM (Bytes)
		Acc.	Sen.	Spec.					
PTB	W/O Resampling	**99.35**	**99.43**	**98.92**	1429	212665	446.82	21188.2	101760
	With Resampling	99.33	99.25	99.74	**533**	**43489**	**150.77**	**7048.49**	**20160**
PTB-XL	W/O Resampling	83.42	77.12	87.44	829	99865	224.40	10598.41	44880
	With Resampling	**84.36**	**78.60**	**88.03**	**533**	**43489**	**150.77**	**7048.49**	**20160**

This in turn makes the inference time 2.96× faster requiring 3.01× less energy. Moreover, resampling reduces the memory footprint by 5.05× thus making the baseline architecture memory-efficient. For PTB-XL dataset, our baseline architecture with resampling outperforms the one without resampling as shown in Table 9. Also, it reduces the – parameter count, FLOP count, inference time, energy consumption, and memory footprint of the baseline architecture by 1.56×, 2.30×, 1.49×, 1.50×, and 2.23×, respectively. The ablation study shows that resampling in our methodology helps to achieve energy and memory efficiency without significant performance loss making it suitable for resource constraint wearable devices.

6 CONCLUSION

This paper proposes an energy-efficient methodology for real-time MI detection on wearable devices using a Convolutional Neural Network (CNN). It involves novel pre-processing of the heartbeat segments to reduce the sample size that allows the baseline CNN to outperform the state-of-the-art works for two different datasets - PTB and PTB-XL, while being energy and memory-efficient. Moreover, we also propose a Template Matching based Early Exit (TMEX) CNN architecture that further increases the energy efficiency compared to baseline architecture while maintaining similar performance. On PTB dataset, our baseline and TMEX architecture achieve 99.33% and 99.24% accuracy, whereas on PTB-XL dataset they achieve 84.36% and 84.24% accuracy, respectively. Evaluation on real hardware shows that our baseline architecture achieves from **0.6x** to **53x** more energy efficiency while outperforming state-of-the-art works on wearable devices. Moreover, our TMEX architecture further achieves 8.12% (PTB) and 6.36% (PTB-XL) more energy efficiency compared to the baseline architecture while maintaining similar performance. To the best of our knowledge, the baseline and TMEX architecture of our methodology achieve the best performance on wearable devices while being energy-efficient with a RAM footprint of only 20 KB.

ACKNOWLEDGMENTS

Authors would like to thank the anonymous reviewers for their constructive suggestions which significantly improved the credibility and readability of the manuscript. This work was partially supported by the National Science Foundation (NSF) under awards CMMI-1739503 and CCF-2140154. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] 2020. *Heart Disease Facts.* Retrieved May 3, 2020 from https://www.cdc.gov/heartdisease/facts.htm

[2] 2020. *SmartCardia.* Retrieved Aug 10, 2020 from https://smartcardia.com/

[3] U Rajendra Acharya, Hamido Fujita, Shu Lih Oh, Yuki Hagiwara, Jen Hong Tan, and Muhammad Adam. 2017. Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. *Information Sciences* 415 (2017), 190–198.

[4] U Rajendra Acharya, Hamido Fujita, Vidya K Sudarshan, Shu Lih Oh, Muhammad Adam, Joel EW Koh, Jen Hong Tan, Dhanjoo N Ghista, Roshan Joy Martis, Chua K Chua, et al. 2016. Automated detection and localization of myocardial infarction using electrocardiogram: a comparative study of different leads. *Knowledge-Based Systems* 99 (2016), 146–156.

[5] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. Springer, 1–4.

[6] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. 2017. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*. PMLR, 527–536.

[7] R Bousseljot, D Kreiseler, and A Schnabel. 1995. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. *Biomedizinische Technik/Biomedical Engineering* 40, s1 (1995), 317–318.

[8] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2020. Once for All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations*. https://arxiv.org/pdf/1908.09791.pdf

[9] Lile Cai, Anne-Maelle Barneche, Arthur Herbout, Chuan Sheng Foo, Jie Lin, Vijay Ramaseshan Chandrasekhar, and Mohamed M Sabry Aly. 2019. TEA-DNN: the Quest for Time-Energy-Accuracy Co-optimized Deep Neural Networks. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.

[10] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. 2016. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits* 52, 1 (2016), 127–138.

[11] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 2 (2019), 292–308.

[12] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

[13] Xuanyi Dong and Yi Yang. 2019. Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems*. 759–770.

[14] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Remi Gribonval, Herve Jegou, and Armand Joulin. 2020. Training with Quantization Noise for Extreme Fixed-Point Compression. *arXiv preprint arXiv:2004.07320* (2020).

[15] Igor Fedorov, Ryan P Adams, Matthew Mattina, and Paul Whatmough. 2019. Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers. In *Advances in Neural Information Processing Systems*. 4978–4990.

[16] Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[17] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation* 101, 23 (2000), e215–e220.

[18] GUSTO Angiographic Investigators. 1993. The effects of tissue plasminogen activator, streptokinase, or both on coronary-artery patency, ventricular function, and survival after acute myocardial infarction. *New England Journal of Medicine* 329, 22 (1993), 1615–1622.

[19] Nitthilan Kanappan Jayakodi, Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. 2020. Design and Optimization of Energy-Accuracy Tradeoff Networks for Mobile Platforms via Pretrained Deep Models. *ACM Transactions on Embedded Computing Systems (TECS)* 19, 1 (2020), 1–24.

[20] Silicon Labs. 2022. *Silicon Labs Energy Profiler*. Retrieved Jan 27, 2022 from https://docs.silabs.com/simplicity-studio-5-users-guide/1.0/using-the-tools/energy-profiler/#:~:text=Simplicity%20Studio%C2%AE%205%20(SSv5,power%20performance%20of%20these%20systems.

[21] HJ Landau. 1967. Sampling, data transmission, and the Nyquist rate. *Proc. IEEE* 55, 10 (1967), 1701–1706.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[23] He Li, Kaoru Ota, and Mianxiong Dong. 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE network* 32, 1 (2018), 96–101.

[24] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[25] Qing Lu, Weiwen Jiang, Xiaowei Xu, Yiyu Shi, and Jingtong Hu. 2019. On Neural Architecture Search for Resource-Constrained Hardware Platforms. *arXiv preprint arXiv:1911.00105* (2019).

[26] Hossein Mamaghanian, Nadia Khaled, David Atienza, and Pierre Vandergheynst. 2011. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. *IEEE Transactions on Biomedical Engineering* 58, 9 (2011), 2456–2466.

[27] Harold Martin, Ulyana Morar, Walter Izquierdo, Mercedes Cabrerizo, Anastasio Cabrera, and Malek Adjouadi. 2021. Real-Time frequency-independent single-lead and single-beat myocardial infarction detection. *Artificial Intelligence in Medicine* 121 (2021), 102179.

[28] Srinivasan Murali, Francisco Rincon, and David Atienza. 2015. A wearable device for physical and emotional health monitoring. In *2015 Computing in Cardiology Conference (CinC)*. IEEE, 121–124.

[29] Fragoulis Nikolaos, Ilias Theodorakopoulos, Vasileios Pothos, and Evangelos Vassalos. 2019. Dynamic Pruning of CNN networks. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*. 1–5. https://doi.org/10.1109/IISA.2019.8900711

[30] Mohanad Odema, Nafiul Rashid, and Mohammad Abdullah Al Faruque. 2021. EExNAS: Early-Exit Neural Architecture Search Solutions for Low-Power Wearable Devices. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.
[31] Jiapu Pan and Willis J Tompkins. 1985. A real-time QRS detection algorithm. *IEEE transactions on biomedical engineering* 3 (1985), 230–236.
[32] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. 2016. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 475–480.
[33] Nafiul Rashid and Mohammad Abdullah Al Faruque. 2020. Energy-efficient Real-time Myocardial Infarction Detection on Wearable Devices. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*. 4648–4651. https://doi.org/10.1109/EMBC44109.2020.9175232
[34] Nafiul Rashid, Manik Dautta, Peter Tseng, and Mohammad Abdullah Al Faruque. 2021. HEAR: Fog-Enabled Energy-Aware Online Human Eating Activity Recognition. *IEEE Internet of Things Journal* 8, 2 (2021), 860–868. https://doi.org/10.1109/JIOT.2020.3008842
[35] Giorgio Roffo, Simone Melzi, Umberto Castellani, and Alessandro Vinciarelli. 2017. Infinite latent feature selection: A probabilistic latent graph-based ranking approach. In *Proceedings of the IEEE International Conference on Computer Vision*. 1398–1406.
[36] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. 2020. Why should we add early exits to neural networks? *arXiv preprint arXiv:2004.12814* (2020).
[37] Dionisije Sopic, Amin Aminifar, Amir Aminifar, and David Atienza. 2017. Real-time classification technique for early detection and prevention of myocardial infarction on wearable devices. In *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 1–4.
[38] Dionisije Sopic, Amin Aminifar, Amir Aminifar, and David Atienza. 2018. Real-time event-driven classification technique for early detection and prevention of myocardial infarction on wearable systems. *IEEE transactions on biomedical circuits and systems* 12, 5 (2018), 982–992.
[39] Nils Strodthoff, Patrick Wagner, Tobias Schaeffter, and Wojciech Samek. 2021. Deep Learning for ECG Analysis: Benchmarks and Insights from PTB-XL. *IEEE Journal of Biomedical and Health Informatics* 25, 5 (2021), 1519–1528. https://doi.org/10.1109/JBHI.2020.3022989
[40] Kalle Tammemäe, Axel Jantsch, Alar Kuusik, Jürgo-Sören Preden, and Enn Õunapuu. 2018. Self-aware fog computing in private and secure spheres. In *Fog Computing in the Internet of Things*. Springer, 71–99.
[41] TensorFlow. 2022. *Post-training dynamic range quantization.* Retrieved Jan 27, 2022 from https://www.tensorflow.org/lite/performance/post_training_quant
[42] Bram van Ginneken. 2017. Fifty years of computer analysis in chest imaging: rule-based, machine learning, deep learning. *Radiological physics and technology* 10, 1 (2017), 23–32.
[43] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter. 2020. PTB-XL, a large publicly available electrocardiography dataset. *Scientific data* 7, 1 (2020), 1–15.
[44] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8612–8620.
[45] WHO. 2020. *WHO — Prevention of Recurrences of Myocardial Infarction and Stroke Study.* Retrieved May 3, 2020 from https://www.who.int/cardiovascular_diseases/priorities/secondary_prevention/country/en/index1.html
[46] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).
[47] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).