Layer-Wise Learning Framework for Efficient DNN Deployment in Biomedical Wearable Systems

Saleh Baghersalimi*, Alireza Amirshahi*, Tomas Teijeiro†, Amir Aminifar‡, David Atienza*
**École Polytechnique Fédérale de Lausanne (EPFL)*, Lausanne, Switzerland
†*Basque Center for Applied Mathematics (BCAM)*, Bilbao, Spain
‡*Lund University (LU)*, Lund, Sweden
*{saleh.baghersalimi, alireza.amirshahi, david.atienza}@epfl.ch, †tteijeiro@bcamath.org, ‡amir.aminifar@eit.lth.se

Abstract—**The development of low-power wearable systems requires specialized techniques to accommodate their unique requirements and constraints. While significant advancements have been made in the inference phase of artificial intelligence, the training phase remains a challenge, particularly for biomedical wearable systems. Traditional training algorithms might not be suitable for these applications due to the substantial memory requirements and high computational costs associated with processing the large number of bits involved in neural network operations. In this paper, we introduce a novel learning procedure specifically designed for low-power wearable systems, dubbed Bio-BPfree (deep neural network training without backpropagation for low-power wearable systems). Using a two-class classification task, Bio-BPfree replaces conventional forward and backward backpropagation passes with four forward passes, two for data of the positive class and two for data of the negative class. Each layer is equipped with a unique objective function aimed at minimizing the distance between data points within the same class while maximizing the distance between data points from different classes. Our experimental results, which were obtained by conducting rigorous evaluations on the MIT-BIH dataset that features electrocardiogram (ECG) signals, effectively demonstrate the superior performance and suitability of Bio-BPfree for two-class classification tasks, particularly within the challenging environment of low-power wearable systems designed for continuous health monitoring and assessment.**

Index Terms—**Low-power wearable systems, Training algorithms, Memory requirements, Deep neural networks.**

I. Introduction

Neural network training on low-power wearable devices boasts numerous benefits, such as continuous on-device training that allows learning without a cloud or external server connection. This results in reduced latency and improved responsiveness and is vital for real-time applications. On-device training also mitigates privacy concerns by limiting sensitive data transmission and leads to energy savings by avoiding data transfer and processing on external servers. These advantages are crucial for devices with limited battery life and contribute to enhanced performance in personalized neural networks.

Training neural networks for edge devices is an active research area due to various challenges. During training, neural networks require more bits for data representation, increasing resource needs. Implementing backpropagation with optimization methods like SGD [1] or Adam [2] is problematic. Computing gradients for a layer requires gradients from the next layer, which delay weight updates. This process is time-consuming and presents significant challenges for edge devices with limited processing capabilities and memory.

Powerful algorithms such as SGD [1] and Adam [2], along with large datasets, facilitate neural network training; however, challenges arise when applied to low-power wearable devices. State-of-the-art deep learning frameworks typically rely on high-power GPUs, essential for frameworks like Squeeze and Excitation networks [3], CycleGAN [4], and Parallel WaveNet [5]. These frameworks' exceptional results are due to GPUs' computational power, but incorporating GPUs into edge devices is problematic due to high power consumption.

Human decision-making involves considering current observations and past experiences, which current neural networks and deep learning algorithms do not emulate. Humans use their senses to make decisions during everyday tasks and efficiently transfer knowledge between tasks. Current deep learning models lack this efficiency, so there is a need to develop models that simulate human-like learning processes for increased adaptability and effectiveness across scenarios.

In this research article, we introduce an innovative approach to train deep neural networks without relying on backpropagation. We propose a layer-wise training strategy that takes advantage of locally generated errors, allowing independent training of each layer and updating hidden-layer weights during the forward pass. By employing local loss functions, we negate the need for gradient backpropagation to preceding layers. Our approach aims to maximize the distance between distinct categories while minimizing intra-category distances in feature spaces, fostering valuable representations within hidden units for precise binary classification.

II. Layer-Wise Learning Framework for DNN Deployment to Low-Power Wearable Systems

We introduce *Bio-BPfree*, an innovative approach for training Deep Neural Networks (DNNs) for binary class classification, eliminating the need for backpropagation. Rather than propagating errors globally, each weight layer is trained using a local learning signal that is not back-propagated throughout

This work has been supported by the FVLLMONTI EC project (grant no. 101016776) and the RESoRT project of Botnar Foundation, the grant RYC2021-032853-I funded by MCIN/AEI/ 10.13039/501100011033, the European Union NextGenerationEU/PRTR, and the WASP Program funded by the Knut and Alice Wallenberg Foundation.

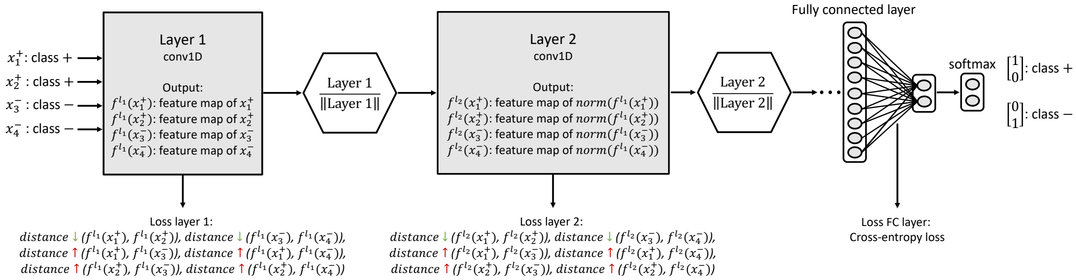

x_1^+: class +
x_2^+ : class +
x_3^- : class −
x_4^- : class −
Layer 1
conv1D
Output:
$f^{l_1}(x_1^+)$: feature map of x_1^+
$f^{l_1}(x_2^+)$: feature map of x_2^+
$f^{l_1}(x_3^-)$: feature map of x_3^-
$f^{l_1}(x_4^-)$: feature map of x_4^-
$\frac{\text{Layer 1}}{\|\text{Layer 1}\|}$
Layer 2
conv1D
Output:
$f^{l_2}(x_1^+)$: feature map of $norm(f^{l_1}(x_1^+))$
$f^{l_2}(x_2^+)$: feature map of $norm(f^{l_1}(x_2^+))$
$f^{l_2}(x_3^-)$: feature map of $norm(f^{l_1}(x_3^-))$
$f^{l_2}(x_4^-)$: feature map of $norm(f^{l_1}(x_4^-))$
$\frac{\text{Layer 2}}{\|\text{Layer 2}\|}$
Fully connected layer
softmax
$\begin{bmatrix}1\\0\end{bmatrix}$: class +
$\begin{bmatrix}0\\1\end{bmatrix}$: class −
Loss layer 1:
distance ↓ ($f^{l_1}(x_1^+)$, $f^{l_1}(x_2^+)$), distance ↓ ($f^{l_1}(x_3^-)$, $f^{l_1}(x_4^-)$),
distance ↑ ($f^{l_1}(x_1^+)$, $f^{l_1}(x_3^-)$), distance ↑ ($f^{l_1}(x_1^+)$, $f^{l_1}(x_4^-)$),
distance ↑ ($f^{l_1}(x_2^+)$, $f^{l_1}(x_3^-)$), distance ↑ ($f^{l_1}(x_2^+)$, $f^{l_1}(x_4^-)$)
Loss layer 2:
distance ↓ ($f^{l_2}(x_1^+)$, $f^{l_2}(x_2^+)$), distance ↓ ($f^{l_2}(x_3^-)$, $f^{l_2}(x_4^-)$),
distance ↑ ($f^{l_2}(x_1^+)$, $f^{l_2}(x_3^-)$), distance ↑ ($f^{l_2}(x_1^+)$, $f^{l_2}(x_4^-)$),
distance ↑ ($f^{l_2}(x_2^+)$, $f^{l_2}(x_3^-)$), distance ↑ ($f^{l_2}(x_2^+)$, $f^{l_2}(x_4^-)$)
Loss FC layer:
Cross-entropy loss

Fig. 1: Example of the proposed learning process of layers in a neural network. Each layer has its loss/objective function, which is to minimize the distance between samples from the same category and maximize the distance between samples from different categories.

the network. Let $\mathcal{D} : \{x_1, .., x_n\}$, where $\mathcal{D}$ is the training set and $x_i \in \mathbb{R}^L$ are the samples with length L. The goal of the main detection task is to predict the output of $y_i \in \mathbb{R}^2$ where y_i shows the class of the corresponding input x_i. In Bio-BPfree, we modify the task as follows. In each training iteration of Bio-BPfree, we take a random subset $\tilde{\mathcal{D}} : \{x_1^+, x_2^+, x_3^-, x_4^-\} \subset \mathcal{D}$. The samples x_1^+ and x_2^+ are sampled from the distribution $p(x_i|y_i = 1)$ and similarly, the samples x_3^- and x_4^- are from the distribution $p(x_i|y_i = 0)$. Each sample of the subset $\tilde{\mathcal{D}}$ is applied to the model one, by one and the intermediate outputs $f^{l_k}(x_i^{\pm})$ are extracted for every layer, where l_k represents the k-th layer.

We employ a distance-based loss function to train each DNN layer, using the subset of $\tilde{\mathcal{D}}$ in each iteration, as illustrated in Fig. 1. The loss function is defined as follows:

$$\begin{aligned}\mathcal{L}_{\tilde{\mathcal{D}}}^{l_k} &= d(f^{l_k}(x_1^+),\ f^{l_k}(x_2^+)) + d(f^{l_k}(x_3^-),\ f^{l_k}(x_4^-)) \\ &\quad + \sum_{i=1,2}\sum_{j=3,4} 1/d(f^{l_k}(x_i^+),\ f^{l_k}(x_j^-)),\end{aligned}$$

where $d(.,.)$ denotes the distance function. This loss function aims to minimize the distance between samples of the same class while maximizing the distance between samples of different classes.

In this study, we assess the performance of *Bio-BPfree* in the context of an end-to-end Deep Neural Network model called Res1DCNN [6]. Res1DCNN consists of 13 convolutional layers and a fully connected layer for binary classification. To facilitate DNN training on low power wearable systems, we implemented a layer-wise training strategy in *Bio-BPfree*. We train the layers sequentially, starting with layer #1 and proceeding to layer #2 and so on. For each layer, the loss function assesses the similarity matching of the feature maps employing the distance L1 for all possible combinations of the four samples from the two classes. The objective is to minimize the L1 distance for combinations containing samples from the same class while maximizing the distance for combinations with samples from different classes, achieved by minimizing the inverse L1 distance. In the final layer, which comprises a fully connected layer, the loss function quantifies the cross-entropy between the prediction generated by a local classifier and the corresponding target. A potential challenge arises when the activities of the first hidden layer contain all the necessary information for classification, rendering it redundant for subsequent layers to learn new features. To address this issue, we introduce a normalization step that removes this information, encouraging subsequent layers to rely on the relative activities of the neurons in the first hidden layer [7], [8].

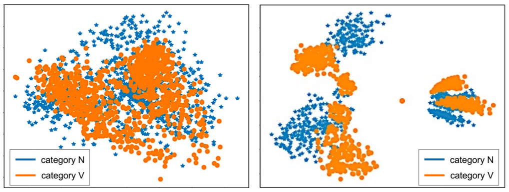
Our approach replaces backpropagation, reducing computational demands and enabling DNN training on low-power wearables. We aimed to assess *Bio-BPfree* as an alternative for low-power DNN development, finding performance comparable to traditional backpropagation on the MIT-BIH dataset. This highlights the potential of distance-based learning as a backpropagation substitute. Although not a primary focus, we also mention *Bio-BPfree*'s applicability in distributed learning scenarios. Here, the model is divided among multiple devices, each handling specific layers, and they communicate to exchange intermediate outputs, ensuring synchronized and uniform training across all devices.

III. Experimental Setup

We tested our method on the PhysioNet MIT-BIH Arrhythmia database [9], with ECG signals from 48 subjects. Using ECG lead II, classes N and V, and leave-one-out cross-validation, we evaluated binary classification in 21 patients with over 40 beats in class V. Class N includes NORMAL, LBBB, RBBB, AESC, and NESC, while Class V covers PVC and VESC. Preprocessing involved extracting and normalizing ECG waves, splitting into heartbeats without filtering or noise removal. Models were trained on these segments, initializing weights from a normal distribution and biases at zero, optimizing inter-class correlation and sample distances. Training utilized the Adam optimizer with a learning rate of 10^{-4}.

IV. Evaluation

The evaluation of the proposed Bio-BPfree method encompasses two main aspects: an examination of how the approach can effectively learn without backpropagation, and a comparison with state-of-the-art algorithms, such as those described in [10], [11], on the MIT-BIH database. The evaluation focuses on understanding learning curves, visualizing output space through PCA, and comparing classification performance and computational costs.



(a) Epoch 1 **(b)** Epoch 20

Fig. 2: Visualization of DNN feature clusters in the last layer using PCA for epoch 1 and 20.

A. *Effectiveness in Learning without Backpropagation*

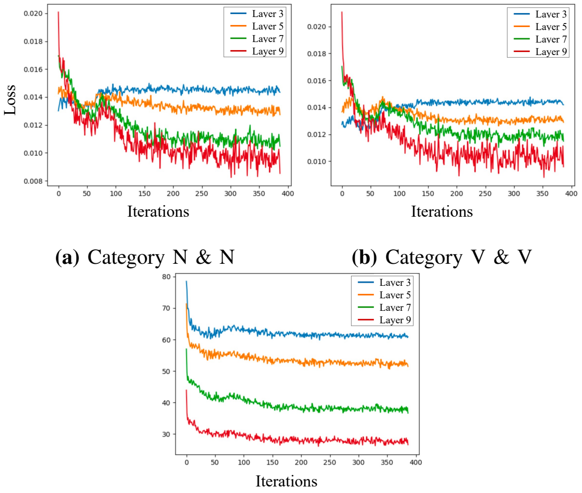
1) Feature Distribution Analysis Through Principal Component Analysis (PCA): Investigating DNN feature visualization through semantic clustering is vital in deep learning research. Semantic clustering groups features based on meaningful relationships rather than numerical similarity alone. In this study, we use PCA to reduce the dimensionality of DNN-learned features and visualize them in a lower-dimensional space. PCA identifies the most significant feature variation directions and projects features accordingly, facilitating interpretable structure analysis.

Figure 2 shows the DNN's last-layer feature visualization using PCA at epochs 1 and 20. Neuronal activations are extracted for input data points, and PCA identifies and projects the principal components into a lower-dimensional space. Scatter plots reveal the feature distribution in this space. Figure 2a shows random patterns of last-layer features in epoch 1, while Figure 2b in epoch 20 reveals distinguishable clusters according to image content. These clusters represent the two data classes, indicating that high-level DNN representations contain information for accurate signal classification.

By comparing the features of Res1DCNN trained with Bio-BPfree and backpropagation, this visualization reveals how the proposed method allows for effective learning without backpropagation.

2) Analyzing Layer-Wise Learning through Loss Function Examination: Examining the loss function in neural network training is key to understanding performance and identifying issues. Often visualized through a loss curve, this study looks at loss function variation across layers to grasp how they learn differences between categories and similarities within them. This analysis helps identify layers needing refinement to improve overall model performance. By minimizing each layer's loss through weight and bias adjustments during training, the examination of loss per layer offers insights into the learning process and potential areas for enhancement.

In this study, the deep neural network (DNN) processes four samples per iteration, including two from Class I (A, B) and two from Class II (C, D). Each layer's loss function has six elements, with two reflecting same-class sample distances ([A1, B1] and [C1, D1]) and four for different-class samples ([A1, C1], [A1, D1], [B1, C1], and [B1, D1]). Figures 3a and 3b show the loss variation for same-class samples, revealing only latter layers learn similarities within the class. Figure 3c illustrates different-class sample loss variation, indicating all layers learn differences between classes. These insights guide the DNN's layer-wise learning process, assisting in optimizing architecture and training for various applications.



(a) Category N & N **(b)** Category V & V

(c) Category V & N

Fig. 3: Variation of loss function across Res1DCNN layers for Intra-Class and Inter-Class sample comparisons.

B. *Comparison with Res1DCNN Trained with Backpropagation*

1) Computational and Memory Costs: In this study, we compare the Res1DCNN model trained using the Bio-BPfree method (without backpropagation) to the Res1DCNN model trained with backpropagation, focusing on computational and memory costs. The comparison begins with assessing the number of trainable parameters in each model, as this metric offers a rough approximation of the memory required for model storage. Generally, a network with fewer parameters demands less memory. As shown in Table I, the Res1DCNN model trained with backpropagation requires 8.72 megabytes to store parameters in 14 layers. Conversely, the Res1DCNN model trained with Bio-BPfree shows variable memory requirements for parameter storage, ranging from 0.002 megabytes for the least demanding layer to 3 megabytes for the most demanding layer, as each layer is trained individually.

The memory assessment goes beyond considering model parameters alone, including the storage of intermediate gradients, activations, and feature maps. As illustrated in Table I, the total memory usage for the Res1DCNN model trained using backpropagation reaches 17.81 megabytes. In contrast, the total memory consumption with Bio-BPfree varies between 0.095 megabytes and 3.0137 megabytes. This substantial decrease in memory requirements enhances the feasibility of training deep neural networks, making them well-suited for biomedical wearable systems.

Computational costs of both backpropagation and the Bio-BPfree method were compared using FLOPS, as outlined in Table I. Calculations were performed with TensorFlow

TABLE I: Comparative analysis of Res1DCNN trained with backpropagation and Res1DCNN trained using Bio-BPfree: Computational and Memory Costs

		Memory for training					FLOP			
Training method	#Layers	Parameters	Gradients	Feature map	Sum	Reduction	Forward	Backward	Sum	Reduction
Backpropagation	14	8.72 Mbytes	8.72 Mbytes	0.3737 Mbytes	17.81 Mbytes	-	4.57 M	4.57 M	9.14 M	-
Bio-BPfree	1 (lightest layer)	0.002 Mbytes	0	0.0930 Mbytes	0.095 Mbytes	99.47%↓	897	0	897	99.99%↓
Bio-BPfree	1 (heaviest layer)	3 Mbytes	0	0.0137 Mbytes	3.0137 Mbytes	83.09% ↓	1.57 M	0	1.57 M	82.8%↓

1.14, profiling both forward and backward operations. For one iteration, the Bio-BPfree method needed significantly fewer FLOPS than traditional backpropagation, indicating lower computational demands. It also required fewer FLOPs than state-of-the-art networks like EfficientNet and MobileNet, emphasizing the efficiency of Bio-BPfree as an alternative to both conventional and modern resource-saving techniques. These tests were conducted in Python 3.6 within a virtual environment to ensure consistency across the development process.

2) Classification Performance: In this study, we compare the performance of Res1DCNN trained with backpropagation, Bio-BPfree, and algorithms from [10] and [11] using the MIT-BIH database. The comparison is visualized using box plot data in Fig 4.

While Res1DCNN trained with Bio-BPfree shows slightly inferior performance compared to backpropagation, it provides reduced computational and memory costs, making it more suitable for wearable systems. Moreover, Bio-BPfree has the highest median sensitivity (97.67) compared to the other algorithms, with fewer low-sensitivity cases.

Algorithm [11] boasts the highest median specificity (99.87), and the narrowest range of specificity values (lower quartile 99.53, upper quartile 100) compared to algorithms Bio-BPfree (lower quartile 95.77, upper quartile 99.76) and [10] (lower quartile 93.29, upper quartile 99.62), indicating more consistent performance across cases.

Furthermore, algorithm Bio-BPfree has the highest median geometric mean (97.87) compared to the algorithms [10] (95.77) and [11] (98.15), with a similar range of geometric mean values to the algorithm [11] (lower quartile 96.12, upper quartile 99.01). Algorithm [10] exhibits a wider range (lower quartile 87.06, upper quartile 98), suggesting more inconsistent performance. Overall, algorithm Bio-BPfree shows better performance in terms of sensitivity and geometric mean, while algorithm [11] leads in specificity. Algorithm [10] generally performs the worst among the four algorithms in all performance metrics.

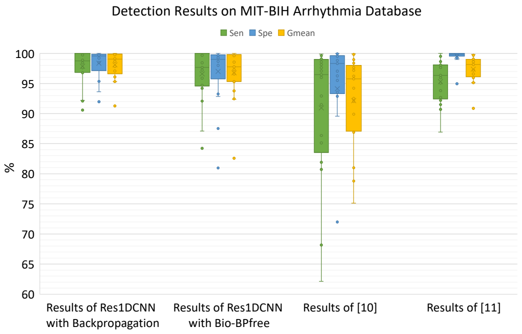


Fig. 4: Performance comparison of algorithms Res1DCNN trained with backpropagation, Res1DCNN trained with Bio-BPfree, [10], and [11] on MIT-BIH Arrhythmia Database in terms of sensitivity, specificity, and geometric mean. Box plots illustrate the distribution of performance metrics, with medians, lower and upper quartiles, and outliers for each algorithm.

V. Conclusion

This paper introduced the Bio-BPfree learning procedure, designed for training deep neural networks in low-power medical wearables. Bio-BPfree replaces conventional backpropagation with a more efficient method, using four forward passes for two-class tasks. Although Res1DCNN trained with Bio-BPfree shows slightly inferior performance, its reduced computational and memory costs make it preferable for biomedical wearables. The Bio-BPfree algorithm offers comparable accuracy to traditional methods, but with significantly lower memory consumption, making it a viable option for resource-limited systems in efficient biomedical signal analysis.

References

[1] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: Proceedings of COMPSTAT'2010. Springer, 2010, pp. 177–186.

[2] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: arXiv preprint arXiv:1412.6980 (2014).

[3] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 7132–7141.

[4] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: Proceedings of the IEEE international conference on computer vision. 2017, pp. 2223–2232.

[5] Aaron Oord et al. "Parallel wavenet: Fast high-fidelity speech synthesis". In: International conference on machine learning. PMLR. 2018, pp. 3918–3926.

[6] Saleh Baghersalimi et al. "Personalized Real-Time Federated Learning for Epileptic Seizure Detection". In: IEEE Journal of Biomedical and Health Informatics (2021), pp. 1–1. DOI: 10.1109/JBHI.2021.3096127.

[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: arXiv preprint arXiv:1607.06450 (2016).

[8] Matteo Carandini and David J Heeger. "Normalization as a canonical neural computation". In: Nature Reviews Neuroscience 13.1 (2012), pp. 51–62.

[9] George B Moody and Roger G Mark. "The impact of the MIT-BIH arrhythmia database". In: IEEE engineering in medicine and biology magazine 20.3 (2001), pp. 45–50.

[10] Naif A. Alajlan et al. "Detection of premature ventricular contraction arrhythmias in electrocardiogram signals with kernel methods". In: Signal, Image and Video Processing 8 (2014), pp. 931–942.

[11] M Sabarimalai Manikandan et al. "Robust detection of premature ventricular contractions using sparse signal decomposition and temporal features". In: Healthcare Technology Letters 2.6 (2015), pp. 141–148.