OPEN ACCESS
International Journal of Neural Systems, Vol. 35, No. 13 (2025) 2550053 (18 pages)

DOI: 10.1142/S0129065725500534

World Scientific
www.worldscientific.com

Deep Unfolded Variable Projection Networks

Gergő Bognár*
Department of Numerical Analysis
ELTE Eotvos Lorand University
Pázmány Péter stny 1/C, Budapest 1117, Hungary
bognargergo@staff.elte.hu

Manuel Feindert*
Institute of Signal Processing, Johannes Kepler University Linz
Altenberger Street 69, Linz 4040, Austria
manuel.feindert@jku.at

Christian Huber
Institute of Signal Processing, Johannes Kepler University Linz
Altenberger Street 69, Linz 4040, Austria
Silicon Austria Labs GmbH
Altenberger Street 69, Linz 4040, Austria

Michael Lunglmayr and Mario Huemer
Institute of Signal Processing, Johannes Kepler University Linz
Altenberger Street 69, Linz 4040, Austria

Péter Kovács
Department of Numerical Analysis, ELTE Eotvos Lorand University
Pázmány Péter stny 1/C, Budapest 1117, Hungary

Received 7 April 2025
Accepted 6 July 2025
Published Online 27 August 2025

In this paper, we present a hybrid learning framework that integrates two model-driven AI paradigms: Deep unfolding and Variable Projections (VPs). The core idea is to unfold the iterations of VP solvers for separable nonlinear least squares (SNLLS) problems into trainable neural network layers. As a consequence, the network is capable of learning optimal nonlinear VP parameters during inference, which is a form of model-based meta-learning. Furthermore, the architecture incorporates prior knowledge of the underlying SNLLS problem, such as basis function expansions and signal structure, which enhance interpretability, reduce model size, and lower data requirements. As a case study, we adapt the proposed deep unfolded VPNet to learn ECG representations for the classification of five arrhythmias. Experimental results on the MIT-BIH Arrhythmia Database show that VPNet achieves performance comparable to state-of-the-art ECG classifiers, attaining 95% accuracy while maintaining a compact architecture. Its low computational complexity enables efficient

*Corresponding authors.

training and inference, making it highly suitable for real-time, power-efficient edge computing applications. This is further validated through embedded implementation on STM32 microcontrollers.

Keywords: Variable projection; model-driven neural network; deep unfolding; ECG signal processing; Hermite functions; embedded systems.

1. Introduction

Recent advancements in Artificial Intelligence (AI) have significantly impacted healthcare, particularly in cardiology, where Machine Learning (ML) and Deep Learning (DL) enhance early disease detection, diagnosis, and personalized treatment planning. AI-powered models can analyze complex multimodal medical data,[1] including electrocardiograms (ECGs), photoplethysmograms (PPGs), and blood pressure (BP) signals, often collected through wearable devices that provide continuous 24-h monitoring. These models are capable of processing big and heterogenius medical data achieving diagnostic performance comparable to or even surpassing human experts in detecting cardiac abnormalities and arrhythmias.[2] However, integrating AI-based ECG analysis into clinical practice is challenging due to the lack of explainability and robustness.[3] Therefore, in safety-critical fields like healthcare, there is a need for AI models that are trustworthy, transparent, and interpretable.[4]

There are various strategies to explain how and why a machine made a decision. According to the taxonomy introduced by Lipton,[5] these approaches can be classified as either *post-hoc* explainable or transparent. The former includes techniques that explain the results of model-agnostic, black-box AI algorithms after training. In contrast, the methods in the latter category enforce interpretability by design, incorporating prior knowledge into the architecture, loss function, or training data of the AI model. Based on integrated informative priors, several learning paradigms exist, such as informed[6] and physics-informed ML,[7] model-based learning,[8] and knowledge-augmented DL.[9]

When prior knowledge about the underlying mathematical model of the target problem is incorporated through the selection of the AI model architecture, this is referred to as model-based AI.[8] This is an emerging field within explainable (X)AI and signal processing, which combines the advantages of mathematical modeling with data-driven disciplines by incorporating numerical algorithms (e.g. finite element methods),[10] mathematical and physical heuristics (e.g. sparsity, smoothness, dynamic characteristics, adaptive filtering) into the architecture, and the training loss of AI methods. Deep unfolding is a prominent approach within the model-based AI paradigm,[11] converting iterative algorithms into deep neural networks (DNNs) by structuring each layer to represent a single iteration. Unlike end-to-end DNN methods, deep unfolding leverages decades of advancements in signal and image processing by explicitly integrating domain knowledge and numerical heuristics — such as sparse recovery,[12–14] dictionary learning,[15,16] estimation theory,[17–19] and noise modeling[20,21] — into the learning framework. To date, these applications have demonstrated that this approach reduces resource requirements while improving power- and data-efficient training.

In this paper, we design the network topology based on the theoretical foundations of deep unfolding and integrate it with variable projections (VPs). VP is a numerical approach for solving separable nonlinear least squares (SNLLS) problems, where the model is linear in some parameters and nonlinear in others (to be referred to as linear and nonlinear parameters of the model).[22] This separation reduces the search space for the underlying numerical optimization and accelerates convergence,[23] making it applicable as an optimizer for training neural networks.[24] Since its introduction, VP has found numerous applications in scientific computing and engineering.[25,26] More recently, it has been incorporated into learning frameworks, where the initial layers are formulated as VP operators and trained jointly with subsequent neural networks.[27] This layer-wise SNLLS formulation incorporates prior knowledge through parameterized basis function expansions of the input signal, enhancing the interpretability of the learned representation. Note that

the underlying SNLLS problem and its VP reformulation are typically solved using iterative numerical methods,[28] such as trust-region or simple gradient descent approaches. Consequently, the iterations of the VP solver can be unfolded into a neural network as well. Leveraging these principles, we propose the deep unfolded VPNet and demonstrate its effectiveness in ECG arrhythmia classification.

In summary, the main contributions of this paper are as follows:

- Learning framework: A hybrid model-based XAI approach is proposed, combining the paradigms of deep unfolding and VPs.
- Embedded implementation: The proposed deep unfolded VPNet model has low computational complexity, making it well-suited for power-efficient edge computing applications. This is demonstrated through implementation on STM32H723ZG[29] and STM32F446RE[30] microcontrollers.
- Medical application: The model is adapted to learn morphological features of ECG signals and evaluated for classifying five arrhythmias according to AAMI standards.[31]

The rest of the paper is organized as follows. Section 2 reviews related work on the VP and deep unfolding paradigms. Section 3 presents the deep unfolded VPNet architecture and introduces a novel embedded implementation. The case study on ECG arrhythmia classification is described in Sec. 4. Finally, Secs. 5–7 discuss the results, limitations and future research, and conclude the work.

2. Background

In this section, we briefly review the related mathematical background and applications of VPs and deep unfolding.

2.1. *Variable projections*

VPs[22] provides a framework to solve SNLLSs problems, i.e. optimization problems involving both linear and nonlinear parameters. Consider nonlinear models of the following form:

$$\boldsymbol{x} \approx \sum_{k=0}^{n-1} c_k \Phi_k(\boldsymbol{\theta}) = \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{c} \tag{1}$$

and the corresponding least squares parameter optimization

$$\min_{\boldsymbol{\theta},\boldsymbol{c}} r(\boldsymbol{x},\boldsymbol{c},\boldsymbol{\theta}) = \min_{\boldsymbol{\theta},\boldsymbol{c}} \|\boldsymbol{x} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{c}\|_2^2, \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ is the observed input data to be modeled (approximated), $\Phi_k(\boldsymbol{\theta}) \in \mathbb{R}^m$ refers to a parametric function system of nonlinear system parameters $\boldsymbol{\theta} \in \mathbb{R}^\ell$, and c_k denotes the linear parameters of the model. In the following, we will consider the matrix notation with system matrix $\boldsymbol{\Phi}(\boldsymbol{\theta}) \in \mathbb{R}^{n\times m}$ and linear parameter vector $\boldsymbol{c} \in \mathbb{R}^n$, where $\boldsymbol{\Phi}(\boldsymbol{\theta})$ is the column-wise composition of $\Phi_k(\boldsymbol{\theta})$. It has been shown,[22] that the minimization of the nonlinear functional (2) is equivalent to the minimization of the VP functional

$$\min_{\boldsymbol{\theta}} r_2(\boldsymbol{x},\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \|\boldsymbol{x} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{x}\|_2^2, \tag{3}$$

i.e. the optimization needs to be carried out with respect to the nonlinear parameters $\boldsymbol{\theta}$ only, and the linear parameters can be obtained by $\boldsymbol{c} = \boldsymbol{\Phi}^+(\boldsymbol{\theta})x$. Here, $\boldsymbol{\Phi}^+(\boldsymbol{\theta})$ denotes the Moore–Penrose pseudoinverse of $\boldsymbol{\Phi}(\boldsymbol{\theta})$. We note the relation with best approximation problem in Hilbert spaces, where the space itself is also variable. Once we optimized the nonlinear parameters $\boldsymbol{\theta}$, the best approximation is actually the orthogonal projection to the column space of $\boldsymbol{\Phi}(\boldsymbol{\theta})$, and the linear parameters $\boldsymbol{c}$ can be interpreted as generalized Fourier coefficients (if the system $\Phi_k(\boldsymbol{\theta})$ is otherwise orthogonal). In this sense, VP can be viewed as the generalization of classical transformation methods, like Fourier and wavelets, leading to parametric and adaptive transformations.

Selection of the proper set of basis functions and their parametrization can be induced by former practical and theoretical finding of the target applications, such as decaying exponentials fits well to MRI imaging,[32] rational functions to system identification,[33] and adaptive Hermite functions to modeling of spike-like waveforms.[34] Notable signal processing applications include biomedical signal processing, such as ECG delineation,[35] classification,[36] and compression.[37] In addition, the VP formulation is applicable to various nonbiomedical signal processing problems,[25] including system identification,[38] multiple frequency estimation,[39] state estimation.[40] Furthermore, in signal processing, VP has further applications as a generic SNLLS solver.[28,41–43]

The numerical optimization of (3) is usually carried out as a gradient-based optimization method. Assuming that $\boldsymbol{\Phi}$ is differentiable with respect to its nonlinear parameters $\boldsymbol{\theta}$, the derivatives of the VP operators (e.g. the coefficient extraction operator $\boldsymbol{\Phi}^+(\boldsymbol{\theta})$, the projection operator $\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta})$, or the VP functional $r_2(\boldsymbol{x}, \boldsymbol{\theta})$) can be explicitly expressed as of Ref. 22.

During inference, the VP functional $r_2(\boldsymbol{x}, \boldsymbol{\theta})$ can be used to get insight into the quality of the prediction. If the predicted coefficients fail to model the input signal, the VP layer returns a large r_2 value, indicating a problem with the decomposition of the input. An insufficient number of coefficients in the VP layer can cause this. If the model yields many false predictions despite an r_2 value close to 0, the problem will likely be caused by network layers following the VP layer.

2.2. *VPNet*

Variable Projection Networks (VPNet)[27] is a model-driven neural network architecture involving VP as a trainable layer. The idea is motivated by the traditional applications of VP in signal processing, where VP is used for adaptive feature extraction and dimension reduction before the application of an ML method. Despite its success, the separate optimization of the feature extraction and ML parts might lead to suboptimal solutions of the specific target application. Instead of the separate optimization, VPNet combines the two influential concepts into a single feed-forward architecture, where the first layer of the network (referred to as the VP layer) performs a VP operator of the following form:

$$\boldsymbol{x} \mapsto \boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{x} \quad \text{or} \quad \boldsymbol{x} \mapsto \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{x},$$

where the nonlinear parameters $\boldsymbol{\theta}$ act as learnable parameters of the layer. Noting that VP operators are differentiable with respect to $\boldsymbol{\theta}$, and they are linear with respect to the layer input $\boldsymbol{x}$, the usual backpropagation method can be naturally extended to the VP layer as well.

In a theoretical point of view, VPNet can be interpreted as a representation learning scheme, where the model learns the best representation of the input data using the selected basis function system, with respect to the actual target application. Unlike generic DL approaches, like convolutional neural networks, the learned representation is interpretable and explainable, because of the domain knowledge incorporated in the selected basis function system.[44] Since its introduction,[27] the core idea behind VPNet has been extended to spiking neural networks[45] and support vector machines,[46] proving its efficiency in various real-world problems, such as cardiac arrhythmia classification,[27] intelligent tire sensor signal processing,[47] road type classification,[48] sensor fault detection,[46] classification of visually evoked potentials (VEPs),[49] ECG signal quality assessment,[50] and microneurographic waveform detection.[51]

2.3. *Deep unfolding*

Deep unfolding[11] is a model-based paradigm to design deep neural network architectures from iterative algorithms. Here, we introduce the notations adopted to VP operators only, otherwise following Ref. 11.

Consider a two-level optimization problem, where the hidden quantities $\boldsymbol{y}_i \in \mathbb{R}^s$ are to be estimated using the observed input data $\boldsymbol{x}_i \in \mathbb{R}^m$ by means of intermediate representations $\boldsymbol{\theta}_i \in \mathbb{R}^\ell$, where i denotes the sample index. The estimator model takes the general form

$$\hat{\boldsymbol{y}}_i := g_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \hat{\boldsymbol{\theta}}_i),$$

where $\boldsymbol{\vartheta}$ denotes the learnable parameters of the model. The intermediate representations are obtained at inference time by

$$\hat{\boldsymbol{\theta}}_i := \arg\min_{\boldsymbol{\theta}_i} F_{\boldsymbol{\vartheta}}(\boldsymbol{x}_i, \boldsymbol{\theta}_i),$$

where $F_{\boldsymbol{\vartheta}}$ refers to an inference objective function (e.g. to minimize reconstruction error). The model itself can be optimized discriminatively at training time by optimizing the parameters $\boldsymbol{\vartheta}$, i.e. by minimizing the loss function

$$J(\boldsymbol{\vartheta}) := \sum_i L(\boldsymbol{y}_i^*, \hat{\boldsymbol{y}}_i)$$

with respect to $\boldsymbol{\vartheta}$, where L denotes the point-wise loss function between the reference values $\boldsymbol{y}_i^*$ and the predictions $\hat{\boldsymbol{y}}_i$. Note that this formalization is directly connected to the traditional ML approaches, where the ML model is preceded by a separate feature extraction and optimization step (that produces the intermediate representation as a feature mapping).

Assuming that the intermediate representations $\boldsymbol{\theta}_i$ can be optimized (i.e. $F_{\boldsymbol{\vartheta}}$ can be minimized) iteratively, consider an appropriate iteration of the following form:

$$\boldsymbol{\theta}_i^{(k)} = f_{\boldsymbol{\vartheta}}\Big(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(k-1)}\Big),$$

where k denotes the iteration index. The deep unfolding of this iterative method unfolds the first K steps of the iteration as layers of an NN architecture, followed by the estimator model. The learnable parameters $\boldsymbol{\vartheta}$ can also be untied, i.e. different parameters are trained for each layer. The resulting network can be interpreted as a feedforward NN, where, given a predetermined starting value $\boldsymbol{\theta}^{(0)}$, the layers 1 to K compute the intermediate values $\boldsymbol{\theta}^{(1)}$, $\ldots, \boldsymbol{\theta}^{(K)}$ as of $\boldsymbol{\theta}_i^{(k)} := f_{\boldsymbol{\vartheta}_{k-1}}(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(k-1)})$, and the network output is given by $\hat{\boldsymbol{y}}_i := g_{\boldsymbol{\vartheta}_K}(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(K)})$. Assuming that f and g are differentiable, gradient backpropagation can be applied for the model training using the chain-rule.

In particular, the iteration can be derived from the gradient descent iteration

$$\boldsymbol{\theta}_i^{(k)} = \boldsymbol{\theta}_i^{(k-1)} - \delta \cdot \nabla_{\boldsymbol{\theta}_i} F(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(k-1)}),$$

or a projected gradient descent iteration

$$\boldsymbol{\theta}_i^{(k)} = \Pi(\boldsymbol{\theta}_i^{(k-1)} - \delta \cdot \nabla_{\boldsymbol{\theta}_i} F(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(k-1)})),$$

where δ is the step size, and Π is a projection operator. Furthermore, general transformations can be applied using standard NN components, like linear transformations, and nonlinear activation. In Ref. 17, for instance, a gradient iteration is unfolded followed by layer-wise transformations using multilayer perceptrons (MLPs) with learnable weights. There, the iteration can be expressed as

$$\boldsymbol{\theta}_i^{(k)} = \mathrm{MLP}(\boldsymbol{\theta}_i^{(k-1)} - \delta \cdot \nabla_{\boldsymbol{\theta}_i} F(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(k-1)})).$$

In this application, the learnable parameters $\boldsymbol{\vartheta}$ consist of the weights of the MLP, and the step size δ.

Note that deep unfolding can also be interpreted as a representation learning scheme, that combines the model-based feature extraction and the data-driven prediction to a single architecture, optimized together with respect to the target application. We also remark the connection with recurrent neural networks (RNNs) and backpropagation through time. If we use shared weights, the unfolded iteration layers could also be interpreted as an unrolled RNN.

3. Deep Unfolded VPNet

Motivated by the success of VP, VPNet, as well as deep unfolding in signal processing, we introduce a novel model-based NN architecture that is based on the deep unfolding of VP gradient iterations. In this section, we present the proposed NN architecture along with motivation, realization, and implementation on an embedded platform.

3.1. *Architecture*

Choosing the VP functional r_2 in (3) as objective function, and the nonlinear VP parameters $\boldsymbol{\theta}$ as intermediate representations (i.e. feature map), consider the VP gradient iteration of the form

$$f_{\mathrm{iter}}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta} - \delta \cdot \nabla_{\boldsymbol{\theta}} r_2(\boldsymbol{x}, \boldsymbol{\theta}). \tag{4}$$

Then, we propose the deep unfolded iterations

$$\boldsymbol{\theta}_i^{(k)} = \mathrm{MLP}(f_{\mathrm{iter}}(\boldsymbol{x}_i, \boldsymbol{\theta}_i^{(k-1)})),$$

i.e. a VP iteration step followed by an MLP with learnable weights and ReLU activation, as depicted in Fig. 1.

These unfolded layers serve as layers 1 to K of the proposed architecture. Then, a VP transformation is applied to compute the linear VP coefficients, that are treated as the learned feature representation of the input data, and that are forwarded to a dense, fully-connected NN part. Using the notations of the previous chapter, the estimator model is of the following form:

$$\hat{\boldsymbol{y}}_i = NN(\boldsymbol{\Phi}^+(\boldsymbol{\theta}^{(K)})\boldsymbol{x}_i),$$

where NN denotes the learnable dense part of the network. The overview of the complete architecture is presented in Fig. 2.

The motivation and reasoning behind the construction is the following. The traditional ML approach involved a separate VP feature extraction, where the nonlinear parameters $\boldsymbol{\theta}$ were optimized by minimizing (3). Despite of certain advantages, like the low dimension and the explainability of the

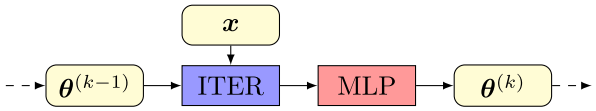

Fig. 1. Deep unfolded layer structure.

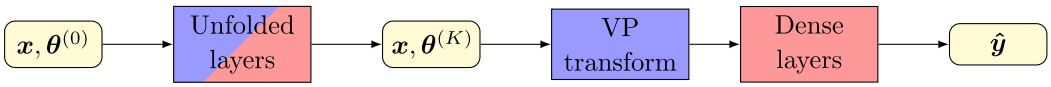

Fig. 2. Deep unfolded VPNet architecture, involving unfolded layers as of Fig. 1. Model-based, data-driven, and mixed components are depicted by blue, red, and split colors, respectively.

features, this paradigm is arguably suboptimal, because the feature representation is optimized separately from the ML model and from the target application. VPNet offers an improved approach with representation learning, where the VP feature extraction operator and the ML model form a joint architecture, resulting in a target-specific VP representation that is optimal for the whole dataset. VPNet combines the advantages of the model-based representation and the data-driven ML predictors. The only possible drawback is that the nonlinear parameters $\boldsymbol{\theta}$ are learnable parameters of VPNet, i.e. only a single parameter set is learned for a given dataset (cf. the traditional paradigm with separate feature extraction, where it was possible to optimize $\boldsymbol{\theta}$ individually, or groupwise as well, see e.g. Refs. 35–37). The deep unfolded VPNet proposed in this paper has the ability of *learning to learn*, i.e. the network learns in training time how to learn the best system parameters $\boldsymbol{\theta}$ in inference time, using only a few iteration steps and NN layers. In this sense, this development can be considered as moving VPNet towards meta-learning.

This approach eases the constraints given by the prior knowledge of the function system. Consequently, the system gains some degrees of freedom that have to be controlled by observations from the training data.

In more details, the explicit application of the VP gradient iteration enforces that the system parameters $\boldsymbol{\theta}$ provide a good fit to the input data, that enhances the explainability of the method. On the other hand, we allow target-specific transformations of the parameters, where the optimal transformation is trained via MLPs with one hidden layer and ReLU activation. The first K layers of the network together with VP transformation can be interpreted as a both domain- and target-specific representation learning, followed by a target-specific NN prediction. We note that further VP operators could be considered instead of coefficient extraction (like projections, or even residuals), and the dense layers could be interchanged with other popular NN designs.

3.2. *Forward and backpropagation*

We developed the prototype framework in PyTorch,[52] partly relying on its automatic differentiation module, yet we will provide a complete reference here. Both forward and backpropagation will be formulated using the work of Golub and Pereyra.[22]

The forward propagation requires the implementation of the VP operators, which are the same as in VPNet, and the gradient of r_2, where the component k can be formulated as

$$[\nabla_{\boldsymbol{\theta}} r_2(\boldsymbol{x},\boldsymbol{\theta})]_k = \frac{\partial r_2(\boldsymbol{x},\boldsymbol{\theta})}{\partial \theta_k} = -2\boldsymbol{x}^T(\boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))\frac{\partial \boldsymbol{\Phi}(\boldsymbol{\theta})}{\partial \theta_k}\boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{x}.$$

The proposed network can be discriminatively trained, similar to the original VPNet. Given a loss function J (e.g. mean squared error or cross-entropy), the usual backpropagation algorithm can be extended to the unfolded layers as well. This requires the gradient of the gradient, that can be expressed using the derivatives of the VP operators. The partial derivatives of the iteration steps take the following form:

$$\begin{aligned}\frac{\partial f_{\text{iter}}(\boldsymbol{x},\boldsymbol{\theta})}{\partial \theta_k} &= 2\boldsymbol{x}^T \frac{\partial(\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))}{\partial \theta_k}\frac{\partial \boldsymbol{\Phi}(\boldsymbol{\theta})}{\partial \theta_k}\boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{x} \\ &\quad - 2\boldsymbol{x}^T(\boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))\frac{\partial^2 \boldsymbol{\Phi}(\boldsymbol{\theta})}{\partial \theta_k^2}\boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{x} \\ &\quad - 2\boldsymbol{x}^T(\boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))\frac{\partial \boldsymbol{\Phi}(\boldsymbol{\theta})}{\partial \theta_k}\frac{\partial \boldsymbol{\Phi}^+(\boldsymbol{\theta})}{\partial \theta_k}\boldsymbol{x},\end{aligned}$$

where

$$\begin{aligned}\partial(\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta})) &= (\boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))\partial\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}) \\ &\quad + ((\boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))\partial\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}))^T\end{aligned}$$

and

$$\begin{aligned}\partial\boldsymbol{\Phi}^+(\boldsymbol{\theta}) &= -\boldsymbol{\Phi}^+(\boldsymbol{\theta})\partial\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta}) \\ &\quad + \boldsymbol{\Phi}^+(\boldsymbol{\theta})(\boldsymbol{\Phi}^+(\boldsymbol{\theta}))^T\partial\boldsymbol{\Phi}^T(\boldsymbol{\theta})(\boldsymbol{I} - \boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^+(\boldsymbol{\theta})) \\ &\quad + (\boldsymbol{I} - \boldsymbol{\Phi}^+(\boldsymbol{\theta})\boldsymbol{\Phi}(\boldsymbol{\theta}))\partial\boldsymbol{\Phi}^T(\boldsymbol{\theta})(\boldsymbol{\Phi}^+(\boldsymbol{\theta}))^T\boldsymbol{\Phi}^+(\boldsymbol{\theta}).\end{aligned}$$

We note that the explicit computation of derivatives $\boldsymbol{\Phi}^{+}(\boldsymbol{\theta})$ and $\boldsymbol{\Phi}(\boldsymbol{\theta})\boldsymbol{\Phi}^{+}(\boldsymbol{\theta})$ is advised even in combination with automatic differentiation in PyTorch because the current implementation of Moore–Penrose pseudoinverse relies on a singular value decomposition, that can lead to unstable gradients. On the other hand, the usual number of the system parameters $\boldsymbol{\theta}$ is small, and we need the derivatives with respect to $\boldsymbol{\theta}$ only, instead of every entry of the matrix $\boldsymbol{\Phi}(\boldsymbol{\theta})$. Meanwhile, the derivatives of f_{iter} can be safely obtained by the automatic differentiation, if each VP component is computed explicitly, and the function system in $\boldsymbol{\Phi}(\boldsymbol{\theta})$ is twice continuously differentiable.

In order to avoid the usual training problems of deep architectures (like divergence and overfitting), we investigated regularization techniques, and the regularization of the VP transformation in particular. Similar to Ref. 27, the loss function J can be extended with an ℓ_2 regularization in terms of percent root-mean-square difference (PRD), as of

$$J_{\text{VP}}(\boldsymbol{\vartheta}) := J(\boldsymbol{\vartheta}) + \frac{\alpha}{N}\sum_{i=1}^{N}\frac{r_2(\boldsymbol{x}_i;\boldsymbol{\theta}^{(K)})}{\|\boldsymbol{x}_i\|_2^2},$$

where the VP penalty $\alpha \geq 0$ controls the regularization effect.

3.3. *Embedded implementation*

Besides the high-level prototype implementation, we investigated the possibility of real-time inference on embedded systems supporting power-efficient edge computing applications.

In addition to C++, C is still one of the most widely used programming languages in embedded systems. To give a proof of concept for the efficiency of the network and its ability to be used within a variety of different embedded environments, a generally usable C implementation independent of the target platform was developed for the inference steps.

A straightforward implementation approach would be to translate the PyTorch version directly into C. We investigated intermediate representations, like the ONNX format.[53] This is partly supported for C implementation, but the layers of the proposed models have custom operators defined for forward and backpropagation, which has led to certain challenges. Due to this complications, we decided to implement the architecture and its layers directly in C, instead of using an ONNX model for translation.

C does not support object-oriented programming or mathematical vector/matrix/tensor operations. For function calls where no translation is possible or no C support is available, new functions were coded that reproduce the behavior as exactly as possible. The implementation can be divided into four main categories shown in Fig. 3. By defining a rather broad variety of linear algebra functions, those can be reused to further implement functions for the neural network layers and the custom VP layers required. This resulted in a larger but more universally applicable library.

Using structures for the C implementation of vectors, matrices, and tensors enables simple usability with similarities to object-oriented programming. With dynamic arrays being used, most of the function calls also do not return any values and instead operate by reference. This allows for easy code readability when compared to the original Python architecture.

The resulting implementation was coded using elementary C functionality allowing for compatibility on all devices that a proper C compiler exists for. Furthermore, the C implementation was done in a modular and parameterized fashion allowing arbitrary VPUnfold network structures to be implemented by adjusting the corresponding parameters.

During the implementation we faced further memory and computation-related challenges. Large batch sizes drastically increase the memory consumption of the C implementation, which would limit the field of application to high-memory devices. While the Python implementation especially benefits from larger batch sizes, many embedded devices offer only limited memory (564 kB and 128 kB in our case, see Table 5). For instance, a batch size of 4096 with

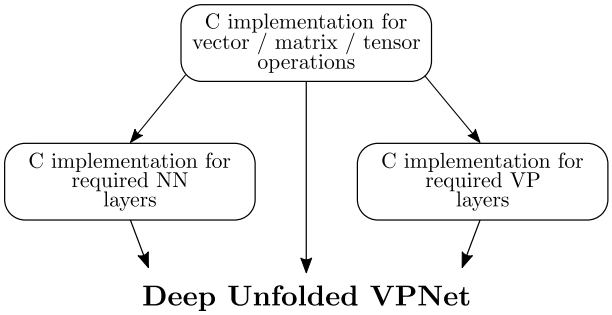

Fig. 3. Library hierarchy for the C implementation.

300 samples would, among other things, require tensor-like memory requirements of dimension 8192, 1,300 on a microcontroller. That is a total of

$$\underbrace{8192 \cdot 1 \cdot 300}_{\text{tensor dimensions}} \cdot \underbrace{4}_{\text{size of float}} \approx 9.375\,\text{M} \gg 564\,\text{kB}$$

which would require temporary tensors larger than the available memory space. Furthermore, since memory for dynamic arrays is allocated during runtime via malloc in the C implementation, selecting a batch size that is too large will not cause compilation warnings, but may lead to runtime errors. Therefore, knowing the maximum available memory is essential when choosing the batch sizes to be processed. We overcame this issue by drastically reducing the batch sizes for the evaluation of the STM32 implementation.

The computational challenge is related to pseudoinverse computation. Comparing the implementation in Python and C all results should be identical, only slight deviations are to be expected. PyTorch uses the SVD decomposition internally to calculate the pseudo-inverse $\mathbf{\Phi}^+(\boldsymbol{\theta})$. However, implementing this in C would result in a very computationally demanding code. In order to simplify the computations, we can make restrictions on the structure of $\mathbf{\Phi}(\boldsymbol{\theta})$. Assuming that $\mathbf{\Phi}(\boldsymbol{\theta})$ is full rank,

$$\mathbf{\Phi}^+(\boldsymbol{\theta}) = (\mathbf{\Phi}^T(\boldsymbol{\theta})\mathbf{\Phi}(\boldsymbol{\theta}))^{-1}\mathbf{\Phi}^T(\boldsymbol{\theta}).$$

Furthermore, if we assume that the function system is orthogonal, or approximately discrete orthogonal, then $\mathbf{\Phi}^+(\boldsymbol{\theta}) \approx \mathbf{\Phi}^T(\boldsymbol{\theta})$. Due to the high complexity of calculating the pseudoinverse, and the approximate orthogonality of investigated function systems, we decided to substitute $\mathbf{\Phi}^+(\boldsymbol{\theta})$ by $\mathbf{\Phi}^T(\boldsymbol{\theta})$. We note that this substitution might lead to numerical inaccuracies, but the architecture seems to be able to compensate for that. Furthermore, evaluations showed that inverting the term $\mathbf{\Phi}^T(\boldsymbol{\theta})\mathbf{\Phi}(\boldsymbol{\theta})$ can lead to numerical problems, which are also avoided by this substitution.

The numerical inaccuracy between using $\mathbf{\Phi}^+(\boldsymbol{\theta})$ and approximating it with $\mathbf{\Phi}^T(\boldsymbol{\theta})$ urges from uniformly sampling the used Hermite functions. When scaling the network, one can assume that the Hermite functions are sampled finer which improves the approximation of the pseudo-inverse since the Hermite orthogonality condition is fulfilled for continuous functions. The previously mentioned avoidance of numerical problems due to near-singularity of the matrix by approximating it is also a huge benefit.

4. Application

We demonstrated the performance of the proposed deep unfolded VPNet for the problem of ECG arrhythmia classification. In this section, we overview the problem statement and our experimental setup.

4.1. *ECG arrhythmia classification*

ECG, as the essential diagnostic tool in cardiology, can predict certain cardiovascular diseases and medical conditions, but requires the expert evaluation of the recorded signal waveforms. Computer-assisted ECG analysis is one of the most widely researched areas of biomedical signal processing, that can efficiently help the work of medical experts, e.g. by providing early warnings of possible abnormalities detected and by supporting the evaluation of long-range or real-time monitoring. This trend is further supported by commercial wearable devices that now measure a wide range of physiological parameters, including ECG. However, state-of-the-art AI models, often comprising billions of parameters, cannot be executed on embedded hardware, making them unsuitable for wearable integration.[54] This highlights the importance of dimensionality reduction techniques in decreasing model size.[55] The deep unfolded VPNet addresses these limitations by applying substantial dimensionality reduction within the VP layers and, consequently, in the classifier head, enabling efficient deployment on embedded hardware.

In this paper, we focus on heartbeat arrhythmia classification as of Ref. 31, evaluated on the benchmark MIT-BIH Arrhythmia Database,[56] accessible on PhysioNet.[57] The task is to classify heartbeats according to the five-class scheme, where the classes are 'normal beat' (N), 'ventricular ectopic beat' (V or VEB), 'supraventricular ectopic beat' (S or SVEB), 'fusion beat' (F), and 'unknown beat' (Q). We adopted the inter-patient (i.e. subject-oriented) paradigm proposed by Ref. 58, where the four paced records are excluded from the dataset, and the remaining 44 records are split evenly: 22 records for training and 22 for testing, to be referred as sets DS1 and DS2. This approach, aligned with Ref. 31, prevents data leakage between patients, and provides realistic evaluation circumstances.

We investigated two subproblems related to arrhythmia classification:

(1) A two-class, balanced scheme: the separation of normal beats and VEBs on a balanced subset, as of Ref. 27. Here, only 4260 normal beats and 4260 VEBs are extracted from DS1, and 3220 plus 3220 from DS2, covering all VEBs and the same amount of normal beats recordwise. Since the original database is highly unbalanced, with almost 90% of the heartbeats being normal, the balanced scheme focuses only on the ability of deep unfolded VPNet to distinguish between the two largest, morphologically different classes of the dataset, without being biased during the training. Furthermore, this case is directly comparable to the performance of the original VPNet architecture.[27]
(2) The standard five-class scheme: the classification of the above five classes trained and evaluated on the full DS1 and DS2 sets, respectively. This case provides direct comparison to state-of-the-art models in arrhythmia classification.[59]

4.2. *Preprocessing and segmentation*

We followed the preprocessing and heartbeat extraction methodology of Ref. 36, which involves the usual paradigms of the field, in accordance to Ref. 59. Namely, the baseline wandering is removed by a wavelet-based approach,[60] and the high frequency fluctuations are smoothed using a lowpass filter with cutoff frequency of 35 Hz. Then, fixed windows are extracted around the R peak annotations of the database. In the two-class, balanced dataset case, the window size is chosen to be 100 samples[27] (50 samples before and 50 after the R peak), corresponding to about 0.28 s, that is expected to include the QRS complexes only. In the five-class case, full dataset case, the window size is chosen to be 300 samples[36] (100 samples in front of and 200 after the R peak), that is designed to cover the complete heartbeat including P and T waves as well. The fixed windows are beneficial for batch computations, for both optimization purposes and in ML applications. However, the fixed windows are also affected by the heart rate variability that might cause overlapping in the heartbeat segments, and as the result of the segmentation the dynamic, rhythm-related information is completely lost. The potential overlapping is removed by adopting the adaptive segmentation approach of Ref. 36. In addition, we investigated RR interval features as dynamic descriptors, that are known to be able to discriminate certain heartbeat types.[59] We adopted four RR interval features derived from intervals between consecutive heartbeats, as of Ref. 61, that are skip connected to the fully connected, dense part of the network, concatenated to the VP output. The rationale behind this approach is that the representation learned by VPNet captures the morphological differences between the classes, but an ML classifier is expected to also benefit from further dynamic information, as it has already been demonstrated by Refs. 36 and 62. We also note that the RR interval features are simple time domain features that are easily obtained in real time.

4.3. *VPNet setup*

Domain knowledge is incorporated via the VP function system, that is chosen as the adaptive Hermite functions, which is directly connected to the ECG morphology. First, consider Hermite polynomials[63] given by the recurrence relation

$$H_0(t) = 1, \quad H_1(t) = 2t,$$
$$H_{k+1}(t) = 2tH_k(t) - 2kH_{k-1}(t) \quad (k \in \mathbb{N}^+, t \in \mathbb{R}),$$

that is a classical orthogonal polynomial system over the real line. Hermite functions are defined by

$$\Phi_k(t) = H_k(t)e^{-t^2/2}/\sqrt{\pi^{1/2}2^k k!} \quad (k \in \mathbb{N}),$$

that form an orthonormal system with respect to the usual scalar product

$$\langle f, g\rangle = \int_{-\infty}^{+\infty} f(t)g(t)\,dt \quad (f, g \in L^2(\mathbb{R})) \qquad (5)$$

in the space of square integrable functions over $\mathbb{R}$. Adaptive Hermite functions are the translated and dilated variations of Hermite functions in the following form:

$$\Phi_k(t; \tau, \lambda) = \sqrt{\lambda}\Phi_k(\lambda(t - \tau)) \quad (k \in \mathbb{N}, t \in \mathbb{R}),$$

that also leads to an orthonormal system with respect to (5). The nonlinear parameters $\boldsymbol{\theta} = (\tau, \lambda)$ represent general localization and shape properties of the Hermite waveforms in terms of time shift and waveform width, respectively.

Hermite functions have multiple applications in signal and image processing, and related fields. Adaptive Hermite systems are especially useful to model compactly supported, spike-like waveforms, and are also closely related to the ECG morphology, see e.g. Ref. 34. In fact, the use of parametrized Hermite functions for modeling ECG waveforms dates back to the foundational work of Sörnmo *et al.*[64] in which the shape of the QRS complex was approximated by dilated Hermite functions. Their study has inspired numerous follow-up efforts: Laguna *et al.*[65] and Kovács *et al.*[37] applied Hermite functions for ECG signal compression; Lagerholm *et al.*[66] utilized the approach for heartbeat clustering; Haraldsson *et al.*[67] extracted features via Hermite expansion for myocardial infarction detection; and Sandryhaila *et al.*[68] improved reconstruction accuracy using a discrete analogue of the dilated Hermite function system. More recently, the clinical relevance of Hermite-based ECG representations has been proved by Cohen's kappa statistics and validated in collaboration with cardiologists Ref. 69.

Namely, the nonlinear VP parameters depict the general localization and shape properties of the system, while the linear VP parameters describe the local morphological differences of the modeled waveforms. This not only leads to a compact and low-dimensional representation of the input signal, but the parameters involved are interpretable and explainable, being directly connected to medical descriptors of the common ECG waveforms (like P wave, QRS complex, and T wave). For more details, we refer to Ref. 27 for theoretical aspects and to Ref. 49 for practical evaluations about explainability.

Motivated by these previous results, we based the ECG evaluation of the proposed network on adaptive Hermite functions, with the note that the novel aspect compared to the original VPNet is that the deep unfolded VPNet learns the optimal translation and dilation (i.e. localization and shape) in inference time. Then, the linear coefficients of the projections are forwarded to the subsequent dense layers, in a similar to way to VPNet. Therefore, the deep unfolded VPNet has at least the same abilities in terms of explainability compared to the original VPNet.

In the practical implementation, we used the slightly different, but equivalent parametrization

$$\Phi_k(t;\tau,\lambda) = \lambda\Phi_k(\lambda^2(t-\tau)) \quad (k\in\mathbb{N}, t\in\mathbb{R}),$$

that provides better numerical properties. We considered a discretely sampled Hermite system of order n, i.e. the first n Hermite functions, restricted to a compact support interval $[a,b]$, and uniformly sampled at points $t_0, t_1, \ldots, t_{m-1} \in [a,b]$. The system matrix $\mathbf{\Phi}(\boldsymbol{\theta}) = \mathbf{\Phi}(\tau,\lambda)$ is composed column-wise, i.e.

$$[\mathbf{\Phi}(\boldsymbol{\theta})]_{jk} = \Phi_k(t_j,\tau,\lambda) \quad (0\le j<m, 0\le k<n).$$

Although nonuniform sampling techniques, like Gauss–Hermite quadrature, would provide better discretization properties, the uniform sampling fits the target application better. Since the ECG dataset is also sampled uniformly, VP can be applied directly without precomputation and resampling. The Hermite system is not discrete orthogonal over a uniform sampling, but as it was shown in Ref. 27, if the number of sampling points m is large enough, and the translation and dilation parameters are in a proper confidence region, the system has approximate discrete orthogonality, i.e. $\mathbf{\Phi}^+(\boldsymbol{\theta}) \approx \mathbf{\Phi}^T(\boldsymbol{\theta})$. We also note that the optimization of system parameters restricted to a confidence region is closely related to the projected gradient descent method and Ref. 17, which also served as a motivation to our work.

5. Results

The efficiency of the proposed method was evaluated in terms of classification performance and computational complexity, with special focus on the embedded implementation.

5.1. *ECG experiments*

We investigated the previously defined cases of ECG arrhythmia classification: the two-class classification on a balanced subset, the five-class classification on the full dataset, and the latter case involving RR interval features. In each of our experiments, the models are trained on DS1 and evaluated on DS2. The model training involves the optimization of the learnable parameters by means of offline backpropagation based on cross-entropy loss and VP regularization, using Adam[79] as optimizer. In addition, an exhaustive evaluation was carried out to find the optimal network configuration, i.e. we performed a grid-based hyperparameter optimization trained on DS1 and validated on DS2. Here, we considered learning-related hyperparameters (learning rate,

batch size, VP penalty), and network-related ones (number of linear VP coefficients, number of deep unfolded iterations, number of MLP neurons per deep unfolded layers, number of dense layer neurons). We note that the proposed approach provides a large variety of network configurations, allowing deep models with several deep unfolded layers, followed by multiple dense layers. In accordance to that VPNet provides a compact architecture with low parameter count, we focused on models with only a few deep unfolded layers, and only a single hidden dense layer. We also note that the step size δ in (4), and the deep unfolding starting parameters $\boldsymbol{\theta}^{(0)}$ could be considered as both learnable and hyperparameters. Here, δ is treated as hyperparameter, which is the same for all deep unfolded layers, while $\boldsymbol{\theta}^{(0)} = (0, 1)$ is fixed, i.e. the starting VP configuration is always the traditional Hermite system.

Classification accuracies of the optimal models are depicted in Tables 1 and 2. We focus on the total accuracy only for the comparisons, and provide the confusion matrix of the best performing model in Table 3. We can conclude that the performance is comparable to VPNet for the balanced subset, and the deep unfolded VPNet outperforms VPNet on the full dataset for the more realistic five-class scheme. We note that we also evaluated VPNet on the full dataset in the above manner for comparison purposes. Related to state of the art, we restricted our comparisons to literature examples that adopt the five-class AAMI problem,[31] following the inter-patient paradigm,[58] and provide evaluations for the whole database. Accordingly, the first block of Table 1 reviews the performance of traditional ML methods, where handcrafted features are extracted prior to classifier training. In contrast, DL approaches typically operate in an end-to-end manner, simultaneously learning features from the raw data and training the classifier, which are indicated in the second block of Table 1. The main drawback of these DL models is their size, which makes them unsuitable for implementation on embedded hardware, as they typically contain millions of trainable parameters. VPNet models, however, are compact in size, comprising only a few hundred parameters. In our former work,[27] VPNet was evaluated using a two-class scheme (normal versus VEB), with QRS complexes as input. In the third block of Table 1, we

Table 1. Test accuracies of the five-class scheme over the full dataset, compared to the state of the art.

Method	Description		Accuracy
de Chazal *et al.*[58]	Waveform + RR	weighted LD	86.1%
Llamedo and Martinez[70]	Waveform + wavelet + RR	weighted LD	93%
Ye *et al.*[61]	Wavelet + ICA + PCA + RR	SVM	86%
Lin and Yang[71]	Wavelet + normalized RR	weighted LD	93%
He[72]	Wavelet + RR + HOS	ensemble learning	91.4%
Dózsa *et al.*[62]	Hermite VP (LC + NLP + PRD) + RR	ensemble learning	93.6%
Bognár and Fridli[36]	Rational VP (LC + NLP) + RR	SVM	94.5%
State-of-the-art[59]	Handcrafted features		83-98%
Li *et al.*[73]	Waveform + Wavelet + RR + HOS	CraftNet	89.24%
Li *et al.*[74]	Raw ECG data	Deep residual CNN	89.99%
Gan *et al.*[75]	Raw ECG data	DenseNet-BiLSTM	92.37%
Zhang *et al.*[76]	Raw ECG data + RR	Adversarial CNN	94.7%
Niu *et al.*[77]	Raw ECG data + RR	CNN-FNN	92.3%
He *et al.*[72]	Raw ECG data + RR	CNN-denseNN	93.0%
State-of-the-art[78]	Handcrafted + learned features		88–99%
	Hermite VPNet		91.9%
	Hermite VPNet + RR		93.2%
Proposed	Hermite Deep Unfolded VPNet		93.5%
Proposed	Hermite Deep Unfolded VPNet + RR		**95.2%**

Notes: Abbreviations: ICA, independent component analysis; LC, linear coefficient; LD, linear discriminants; NLP, nonlinear parameters; PCA, principal component analysis; PRD, percent root-mean-square difference; RR, RR interval features; SVM, support vector machines, FNN, fully-connected neural networks; HOS, higher-order statistics. Bold face highlights the fact that the proposed method's performance is the highest in the table.

Table 2. Test accuracies of the two-class, balanced scheme.

Method	Accuracy
VPNet	96.65%
Deep unfolded VPNet	95.90%

Table 3. Confusion matrix of the best performing model.

		Prediction					
Ground truth		N	S	V	F	Q	Σ
	N	**43825**	184	165	64	0	**44238**
	S	885	**760**	327	0	0	**1972**
	V	379	42	**2795**	4	0	**3220**
	F	330	1	4	**53**	0	**388**
	Q	5	0	2	0	**0**	**7**
	Σ	**45424**	**987**	**3293**	**121**	**0**	**49825**

extended this basic VPNet model (without unfolding) to a five-class scheme using whole heartbeats as input. The fourth block presents the performance of the proposed deep unfolded VPNet variants, demonstrating a clear improvement over both the basic VPNet model without unfolding and the other classifier models listed in Table 1.

Besides performance evaluation, it is worth to discuss the optimal hyperparameter configurations. We have found that the PyTorch implementation prefers larger batch sizes, that could be interpreted as the deep unfolded VPNet favors group-wise (e.g. patient-wise) representation learning compared to heartbeat-wise representations. Furthermore, the results show that usually a few, 2–3 iteration steps are enough to achieve a good performing model.

Besides performance optimization and evaluation, we performed ablation studies and statistical significance analysis. The results in Table 1 show that the deep unfolded layers followed by a VP transformation improve the network performance compared to a fixed VP transformation (as in VPNet), and that the RR interval features are also beneficial by providing time-domain information that is previously lost during the heartbeat segmentation. The neural components of the proposed architecture consist of the simplest MLP that is still a universal approximator, containing only one hidden layer. The hyperparameter optimization further proved that even minimal components provide state-of-the-art performance. In addition to the numeric ablation analysis, we applied statistical significance tests to compare the performance of the VPNet and the proposed deep unfolded VPNet architecture, similar to Ref. 36. Based on paired-sample t-test and McNemar's test (with significance level 5%), we can conclude that the best proposed model in Table 1 is significantly different from the best VPNet model there. In detail, the contingency table of McNemar's test is presented in Table 4. Here, $\chi^2 \approx 604$ with the corresponding p-value of $p < 3.2 \cdot 10^{-130}$, estimated by the Chernoff bound. In summary, we can conclude that all proposed components are necessary and advisable, and the proposed model significantly improves the performance of the original VPNet.

5.2. *Embedded implementation*

The C implementation was tested on STM32 H723ZG[29] and STM32F446RE[30] microcontrollers, with their available memory and clock frequency shown in Table 5. Both the balanced and the full dataset have been evaluated. The main differences to the previous section are that here we focused purely on the deep unfolded VPNet architecture without RR interval features, and the training was performed offline, and only the inference is delegated to the device.

To ensure the identical behavior of C and Python, the results of testing the network with the training

Table 4. McNemar's test for statistical significance.

		Deep unfolded VPNet		
		Positive	Negative	Total
VPNet	Positive	46162	297	46459
	Negative	1271	2095	3366
	Total	47433	2392	49825

Table 5. Specifications of the used STM32 microcontrollers.

	STM32H723ZG	STM32F446RE
Clock speed	550 MHz	180 MHz
Flash size	1024 kB	512 kB
Total RAM	564 kB	128 kB

and test data should be equal. If a batch is misclassified and cannot be traced back to numerical inaccuracy, the implementation is acting incorrectly. With enough memory available (e.g. in a desktop environment), a reduction in batch size is not required, which made it possible to directly computationally compare the two implementations.

Testing the C implementation on an embedded system like the STM32H723ZG microcontroller requires a reduction in batch size due to the above-mentioned memory requirements within the network. Reducing the batch size of a network that was trained with a larger batch size will lead to a change in accuracy and yield different results. Table 6 shows that testing the inference steps with batch size reduced to 10 resulted in an accuracy decrease of $\approx$ 1.29% for the balanced dataset. Effectively no change in accuracy occurs for a batch size reduction using the full dataset. Here, a deviation of 0.01% corresponds to a different classification result for 5 out of 49825 batches.

The substitution of $\mathbf{\Phi}^+(\boldsymbol{\theta})$ with $\mathbf{\Phi}^T(\boldsymbol{\theta})$ is reasonable, because the chosen Hermite system is approximately orthogonal in the discrete domain. To compensate for the expected decrease in accuracy, we retrained the model and allowed changes in the network architecture. The results of this evaluation is displayed in Table 7. Interestingly, the performance drop is minimal for the balanced set, and negligible for the full dataset. The optimal hyperparameters in Table 8 show that the network was able to compensate the numerical inaccuracies by adopting its configuration.

This demonstrates that the deep unfolded VPNet architecture can be implemented on cost-efficient low-power microcontrollers. This allows including its detection capability in medical devices with low costs providing additional online information to the medical personnel, enabling quick response to pathological patterns. Price investigations revealed that, at the time of writing, the chip costs fall within the single-digit dollar range. This demonstrates the tremendous potential of including the described intelligence in medical devices at low cost.

Table 6. Accuracies with a reduced batch size.

Dataset	Change in accuracy
Balanced dataset	95.90% $\rightarrow$ 94.61%
Full dataset	93.47% $\rightarrow$ 93.48%

Table 7. Test accuracy using $\mathbf{\Phi}^+/\mathbf{\Phi}^T$.

Dataset	Using $\mathbf{\Phi}^+$	Using $\mathbf{\Phi}^T$
Balanced dataset	96.18%	95.90%
Full dataset	93.47%	93.47%

5.3. *Complexity*

In inference time, the VP layer of the original VPNet architecture acts only as a linear layer, where the VP operators can be precomputed after the training phase. Deep unfolded VPNet learns the system parameters in inference time, meaning that the VP operators (along with the pseudoinverse) needs to be recomputed by every iteration. This leads to an inherent computational overhead of the deep unfolded VPNet, that can be reduced by substituting $\mathbf{\Phi}^+(\boldsymbol{\theta})$ by $\mathbf{\Phi}^T(\boldsymbol{\theta})$, as discussed earlier. Additionally, the number of layers and the learnable parameters can also be increased. With all this, the deep unfolded VPNet can still be considered a low complexity, light-weight model: in case of the adaptive Hermite system, the number of nonlinear VP parameters is 2, and the evaluations show that 1–3 unfolded layers with 4–16 linear coefficients already lead to good performing models. The total number of learnable

Table 8. Architecture of the best performing networks.

	Dataset	Learning rate	Step size δ	VP linear coefficient	Number of iterations	MLP neurons	Dense neurons
Using $\mathbf{\Phi}^+$	Full	0.001	0.0001	16	3	8	16
Using $\mathbf{\Phi}^T$	Full	0.001	0.0001	16	3	8	16
Using $\mathbf{\Phi}^+$	Balanced	0.01	0.0001	16	3	8	8
Using $\mathbf{\Phi}^T$	Balanced	0.001	0.001	16	2	16	16

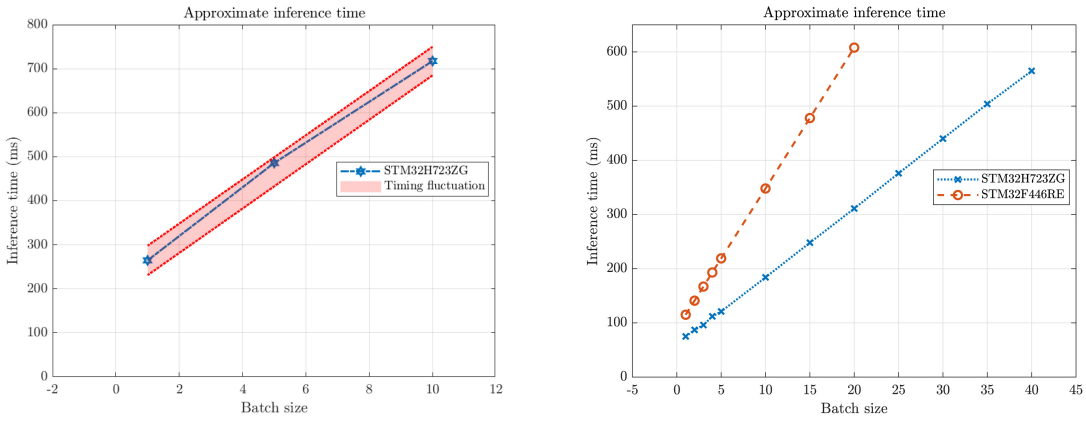

Fig. 4. Inference time for the full dataset (left) and for the balanced dataset (right).

NN parameters ranges between 67 and 685, which is comparable to the parameter count of the original VPNet architecture.

Implementing the architecture on a μC posed several challenges as described previously. Besides the limited CPU performance (addressed by replacing the pseudo-inverse by the above-mentioned approximation) the limited memory posed the most severe limitation. To address this challenge, we optimized the memory consumption by utilizing efficient C data structures. Furthermore, we reduced the batch size. Embedded performance was measured by testing with batches sizes of $(1, 5, 10)$, resulting in inference times displayed in Fig. 4. Due to the full dataset having a larger input size, the implementation did not fit the STM32F446RE, and was only tested on the STM32H723ZG. Given that the original ECG signal was sampled with a sampling rate of 360 Hz,[56] and the inference time for a batch size of (1×300) samples (that covers the full heartbeat) is ≈ 298 ms (according to Fig. 4), we have

$$298\,\text{ms} \ll 300 \cdot \frac{1}{360\,\text{Hz}} = 833.33\,\text{ms}.$$

This concludes that the hardware implementation of the deep unfolded VPNet is feasible for a real-time medical application, delegated to an embedded or edge-computing device.

6. Limitations and Future Research Directions

The proposed VP layer and its unfolded variant employ Hermite functions to model ECG signals. While the function system is a modular element of the VPNet architecture and can be substituted with any other parameterized and differentiable function family, it is currently limited to functions of type $\Phi : \mathbb{R} \to \mathbb{R}$. This constraint restricts the model's applicability to one-dimensional input signals. To overcome this limitation, two-dimensional extensions can be considered — for example, constructing functions as direct products of 1D bases, such as $\Phi(x, y) = \Phi(x) \cdot \Phi(y)$. This modification could open the door to image processing applications, including medical use cases such as artifact and noise reduction in CT imaging,[80] tumor segmentation and classification in MRI,[81] and more.

The interpretability of the VP layer has been studied in prior works. In fact, the trainable Hermite-VP parameters τ and λ can capture the position and width of ECG waveforms, such as the QRS complex,[27] while the output VP coefficients c_k reflect morphological features.[34] However, these do not directly correspond to clinically relevant metrics like QT duration or ST elevation. This gap could be bridged by generating clinically meaningful text explanations using GPT-based models. Therefore, future work will investigate the use of cardiology-specialized GPT models, such as Ref. 82, not to explain the high-dimensional input space, but rather to generate text explanations of the VP latent space, which has significantly lower dimensionality.

Despite its advantages, the proposed deep unfolded VPNet model relies on supervised training, which requires labeled data. This presents challenges for personalized clinical applications, as cardiologically validated labels are typically unavailable for

new patient data. To mitigate this limitation, VPNet could be integrated with self-supervised learning approaches.[83,84] Additionally, combining VPNet with other classification models through ensemble learning techniques may further enhance performance.[85,86]

7. Conclusion

We developed a novel model-based neural network architecture, which combines the representation abilities of the traditional feature extraction methods and the generalization abilities of neural networks, by integrating the deep unfolding paradigm with VPs and VPNet. The proposed architecture can be interpreted as a representation learning scheme that generalizes the expression ability of VPNet. The network is able to learn the optimal nonlinear VP parameters in inference time, instead of using a fixed set of parameters learned during the training, which leads toward model-based meta-learning in the framework of VP.

The concept, similar to VPNet, supports the direct incorporation of target-specific domain knowledge in terms of parametric function systems, and also fits to the XAI paradigm. As a case study, we investigated the well-known problem of arrhythmia classification of ECG heartbeats, following the AAMI standards. Here, the a priori knowledge is exploited by choosing adaptive Hermite functions as parametric system, that provide interpretable and medically explainable morphological representations.

Despite a slight computational overhead compared to VPNet, the proposed deep unfolded VPNet is still a low-complexity, light-weight model that can be operated at a low parameter count. We provided the mathematical and computational background to support the effective and stable evaluation of both forward and backpropagation. We also demonstrated, that the inference can be efficiently implemented on embedded systems, i.e. on microcontrollers, that opens the possibility for real-time applications in power-efficient edge computing environments as well.

In summary, we can conclude that the proposed model efficiently generalizes the VPNet architecture along with a computationally tractable implementation, and provides competitive classification results compared to the state-of-the-art in the target application. We also believe that the proposed idea has broader impact in the related fields: it might induce the development of knowledge-augmented AI models, support real-world applications with specific hardware requirements, and facilitates the investigation of learning models for meta-learning purposes.

Acknowledgments

Project Nos. K146721 and TKP2021-NVA-29 have been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the K_23 "OTKA" and the TKP2021-NVA funding schemes. This work was supported by the University Excellence Fund of Eötvös Loránd University, Budapest, Hungary (ELTE). This project was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. This work is supported by the COMET-K2 "Center for Symbiotic Mechatronics" of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria. G. Bognár and M. Feindert contributed equally to this paper.

ORCID

Gergő Bognár https://orcid.org/0000-0001-7818-5760
Manuel Feindert https://orcid.org/0009-0002-8723-9251
Christian Huber https://orcid.org/0000-0002-5766-5332
Michael Lunglmayr https://orcid.org/0000-0002-4014-9681
Mario Huemer https://orcid.org/0000-0002-3164-7232
Péter Kovács https://orcid.org/0000-0002-0772-9721

References

1. E. Idrobo-Ávila, G. Bognár, D. Krefting, T. Penzel, P. Kovács and N. Spicher, Quantifying the suitability of biosignals acquired during surgery for multimodal analysis, *IEEE Open J. Eng. Med. Biol.* **5** (2024) 250–260.
2. F. W. Asselbergs and A. G. Fraser, Artificial intelligence in cardiology: The debate continues, *Eur. Heart J. Digit. Health* **2** (2021) 721–726.
3. T. Nakamura and T. Sasano, Artificial intelligence and cardiology: Current status and perspective, *J. Cardiol.* **79**(3) (2022) 326–333.

4. A. Holzinger, C. Biemann, C. S. Pattichis and D. B. Kell, What do we need to build explainable AI systems for the medical domain? arXiv:1712.09923.
5. Z. C. Lipton, The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery, *Queue* **16**(3) (2018) 31–57.
6. L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage and J. Schuecker, Informed machine learning – A taxonomy and survey of integrating prior knowledge into learning systems, *IEEE Trans. Knowl. Data Eng.* **35**(1) (2023) 614–633.
7. G. E. Karniadakis, G. Kevrekidis, Ioannis, L. Lu, P. Perdikaris, S. Wang and L. Yang, Physics-informed machine learning, *Nat. Rev. Phys.* **3** (2021) 422–440.
8. N. Shlezinger, J. Whang, Y. C. Eldar and A. G. Dimakis, Model-based deep learning, *Proc. IEEE* **111**(5) (2023) 465–499.
9. Z. Cui, T. Gao, K. Talamadupula and Q. Ji, Knowledge-augmented deep learning and its applications: A survey, *IEEE Trans. Neural Netw. Learn. Syst.* **36**(2) (2025) 2133–2153.
10. D. R. Pereira, M. A. Piteri, A. N. Souza, J. P. Papa and H. Adeli, FEMa: A finite element machine for fast learning, *Neural Comput. Appl.* **32** (2020) 6393–6404.
11. J. R. Hershey, J. L. Roux and F. Weninger, Deep unfolding: Model-based inspiration of novel deep architectures, arXiv:1409.2574.
12. K. Gregor and Y. LeCun, Learning fast approximations of sparse coding, in *Proc. 27th Int. Conf. Machine Learning, ICML' 10* (Omnipress, Madison, WI, 2010), pp. 399–406.
13. S. Wu, A. Dimakis, S. Sanghavi, F. Yu, D. Holtmann-Rice, D. Storcheus, A. Rostamizadeh and S. Kumar, Learning a compressed sensing measurement matrix via gradient unrolling, in *Proc. 36th Int. Conf. Machine Learning*, Proceedings of Machine Learning Research Vol. 97 (PMLR, 2019), pp. 6828–6839.
14. J. Xiang, Y. Dong and Y. Yang, Fista-net: Learning a fast iterative shrinkage thresholding network for inverse problems in imaging, *IEEE Trans. Med. Imaging* **40**(5) (2021) 1329–1339.
15. T. Chang, B. Tolooshams and D. Ba, Randnet: Deep learning with compressed measurements of images, in *2019 IEEE 29th Int. Workshop Machine Learning for Signal Process (MLSP)* (IEEE, 2019), pp. 1–6.
16. B. Tolooshams, A. Song, S. Temereanca and D. Ba, Convolutional dictionary learning based auto-encoders for natural exponential-family distributions, in *Proc. 37th Int. Conf. Machine Learning*, Proceedings of Machine Learning Research, Vol. 119 (PMLR, 2020), pp. 9493–9503.
17. N. Samuel, T. Diskin and A. Wiesel, Learning to detect, *IEEE Trans. Signal Process.* **67**(10) (2019) 2554–2564.
18. H. He, C.-K. Wen, S. Jin and G. Y. Li, Model-driven deep learning for mimo detection, *IEEE Trans. Signal Process.* **68** (2020) 1702–1715.
19. S. Baumgartner, G. Bognár, O. Lang and M. Huemer, Neural network approaches for data estimation in unique word OFDM systems, *IEEE Trans. Veh. Technol.* **73**(3) (2024) 3690–3706.
20. Y. Li, M. Tofighi, J. Geng, V. Monga and Y. C. Eldar, Efficient and interpretable deep blind image deblurring via algorithm unrolling, *IEEE Trans. Comput. Imaging* **6** (2020) 666–681.
21. O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. G. van Sloun and Y. C. Eldar, Deep unfolded robust PCA with application to clutter suppression in ultrasound, *IEEE Trans. Med. Imaging* **39**(4) (2020) 1051–1063.
22. G. H. Golub and V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, *SIAM J. Numer. Anal.* **10** (1973) 413–432.
23. J. H. Hong, C. Zach and A. Fitzgibbon, Revisiting the variable projection method for separable nonlinear least squares problems, in *2017 IEEE Conf. Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2017), pp. 5939–5947.
24. E. Newman, L. Ruthotto, J. Hart and B. van Bloemen Waanders, Train like a (var)pro: Efficient training of neural networks with variable projection, *SIAM J. Math. Data Sci.* **3**(4) (2021) 1041–1066.
25. G. H. Golub and V. Pereyra, Separable nonlinear least squares: The variable projection method and its applications, *Inverse Probl.* **19**(2) (2003) R1–R26, doi: 10.1088/0266-5611/19/2/201.
26. V. Pereyra and G. Scherer, Imaging applications with variable projections, *Am. J. Comput. Math.* **9**(4) (2019) 261–281.
27. P. Kovács, G. Bognár, C. Huber and M. Huemer, VPNET: Variable projection networks, *Int. J. Neural Syst.* **32**(01) (2022) 2150054.
28. D. P. O'Leary and B. W. Rust, Variable projection for nonlinear least squares problems, *Comput. Optim. Appl.* **54**(3) (2013) 579–593, doi: 10.1007/s10589-012-9492-9.
29. STMicroelectronics, STM32H723ZG datasheet (2023), Accessed 25 March 2025.
30. STMicroelectronics, STM32F446RE Datasheet (2021), Accessed 25 March 2025.
31. Association for the Advancement of Medical Instrumentation (AAMI), Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms, American National Standards Institute (ANSI) ANSI/AAMI/ISO EC57 (1998-(R)2008), (2012).
32. M. Paluszny, M. Lentini, M. Martin-Landrove and Torres, Recovery of relaxation rates in MRI T2 weighted brain images via exponential fitting, in *Exponential Data Fitting and its Applications*, eds.

V. Pereyra and G. Scherer (Bentham Science, 2010), pp. 52–70.
33. T. Dózsa, M. Szabari, A. Soumelidis and P. Kovács, Pole identification using discrete Laguerre expansion and variable projection, in *Proc. 22nd Int. Federation of Automatic Control (IFAC) World Congress* (Elsevier, 2023), pp. 1–6 (accepted).
34. P. Kovács, C. Böck, T. Dózsa, J. Meier and M. Huemer, Waveform modelling by adaptive weighted Hermite functions, in *Proc. 44th IEEE Int. Conf. Acoustics Speech Signal Processing (ICASSP)* (IEEE, 2019), pp. 1080–1084.
35. C. Böck, P. Kovács, P. Laguna, J. Meier and M. Huemer, ECG beat representation and delineation by means of variable projection, *IEEE Trans. Biomed. Eng.* **68**(10) (2021) 2997–3008.
36. G. Bognár and S. Fridli, ECG heartbeat classification by means of variable rational projection, *Biomed. Signal Process. Control* **61** (2020) 102034.
37. P. Kovács, S. Fridli and F. Schipp, Generalized rational variable projection with application in ECG compression, *IEEE Trans. Signal Process.* **68** (2020) 478–492.
38. G.-Y. Chen, X.-X. Su, M. Gan, W. Guo and C. L. P. Chen, Robust variable projection algorithm for the identification of separable nonlinear models, *IEEE Trans. Autom. Control* **69**(9) (2024) 6293–6300.
39. Y. E. Garcia, P. Kovács and M. Huemer, Variable projection for multiple frequency estimation, in *ICASSP 2020 — 2020 IEEE Int. Conf. Acoustics Speech Signal Process (ICASSP)* (IEEE, 2020), pp. 4811–4815.
40. J. Zhang, A. M. Pace, S. A. Burden and A. Aravkin, Offline state estimation for hybrid systems via nonsmooth variable projection, *Automatica* **115** (2020) 108871.
41. L. Chen, J.-B. Chen, G.-Y. Chen, M. Gan and C. L. P. Chen, Nuisance parameter estimation algorithms for separable nonlinear models, *IEEE Trans. Syst. Man Cybern. Syst.* **52**(11) (2022) 7236–7247.
42. L. Kaufman, A variable projection method for solving separable nonlinear least squares problems, *BIT Numer. Math.* **15** (1975) 49–57.
43. M. Gan and H.-X. Li, An efficient variable projection formulation for separable nonlinear least squares problems, *IEEE Trans. Cybern.* **44**(5) (2014) 707–711.
44. P. Kovács, Interpretable representation learning for biosignals via variable projection, in *Proc. Workshop Biosignals* (Gttingen Research Online, 2024), pp. 1–4.
45. P. Kovács and K. Samiee, Arrhythmia detection using spiking variable projection neural networks, in *2022 Computing in Cardiology (CinC)*, Vol. 498 (IEEE, 2022), pp. 1–4.
46. T. Dózsa, F. Deuschle, B. Cornelis and P. Kovács, Variable projection support vector machines and its applications using adaptive Hermite expansions, *Int. J. Neural Syst.* **34**(1) (2024) 1–22.
47. T. Dózsa, J. Radó, J. Volk, A. Kisari, A. Soumelidis and P. Kovács, Road abnormality detection using piezoresistive force sensors and adaptive signal models, *IEEE Trans. Instrum. Meas.* **71** (2022) 1–11.
48. T. Dózsa, V. Jurdana, S. B. Šegota, J. Volk, J. Radó, A. Soumelidis and P. Kovács, Road type classification using time-frequency representations of tire sensor signals, *IEEE Access* **12** (2024) 53361–53372.
49. T. Dózsa, C. Böck, J. Meier and P. Kovács, Weighted Hermite variable projection networks for classifying visually evoked potentials, *IEEE Trans. Neural Netw. Learn. Syst.* **36**(7) (2025) 12415–12428.
50. K. Samiee and P. Kovács, On edge wearable ECG signal quality assessment using residual Hermite projection and liquid state machine, a hierarchical domain adaptation approach, in *2024 Computing in Cardiology (CinC)*, Vol. 499 (IEEE, 2024), pp. 1–4.
51. D. Darabos, T. Alina, F. Andrea, M. Anna, K. Ekaterina, N. Barbara, B. Gergő and P. Kovács, Explainable spike detection algorithm on C-fiber microneurography data using weighted Hermite variable projection neural networks, in *Proc. 49th Annual Int. Conf. IEEE Engineering in Medicine and Biology Society (EMBC)* (IEEE, 2025), pp. 1–7.
52. J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, S. Zhang, M. Suo, P. Tillet, X. Zhao, E. Wang, K. Zhou, R. Zou, X. Wang, A. Mathews, W. Wen, G. Chanan, P. Wu and S. Chintala, PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation, in *Proc. 29th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems, ASPLOS '24*, Vol. 2 (Association for Computing Machinery, New York, NY, 2024), pp. 929–947.
53. Open Neural Network Exchange (ONNX), https://onnx.ai/, Accessed 25 March 2025.
54. Z. Sankari and H. Adeli, Heartsaver: A mobile cardiac monitoring system for auto-detection of atrial fibrillation, myocardial infarction, and atrio-ventricular block, *Comput. Biol. Med.* **41**(4) (2011) 211–220.
55. R. J. Martis, U. R. Acharya, H. Adeli, H. Prasad, J. H. Tan, K. C. Chua, C. L. Too, S. W. J. Yeo and L. Tong, Computer aided diagnosis of atrial arrhythmia using dimensionality reduction methods on transform domain representation, *Biomed. Signal Process. Control* **13** (2014) 295–305.
56. G. B. Moody and R. G. Mark, The impact of the MIT-BIH Arrhythmia Database, *IEEE Eng. Med. Biol. Mag.* **20** (2001) 45–50.

57. A. L. Goldberger, L. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng and H. E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals, *Circulation* **101** (2000) e215–e220.
58. P. de Chazal, M. O'Dwyer and R. B. Reilly, Automatic classification of heartbeats using ECG morphology and heartbeat interval features, *IEEE Trans. Biomed. Eng.* **51** (2004) 1196–1206.
59. E. J. D. S. Luz, W. R. Schwartz, G. Cámara-Chávez and D. Menotti, ECG-based heartbeat classification for arrhythmia detection: A survey, *Comput. Methods Programs Biomed.* **127** (2016) 144–164.
60. D. Zhang, Wavelet approach for ECG baseline wander correction and noise reduction, in *2005 IEEE Engineering Medicine and Biology 27th Annual Conf.* (IEEE, 2005), pp. 1212–1215.
61. C. Ye, B. V. K. Vijaya Kumar and M. T. Coimbra, Heartbeat classification using morphological and dynamic features of ECG signals, *IEEE Trans. Biomed. Eng.* **59** (2012) 2930–2941.
62. T. Dózsa, G. Bognár and P. Kovács, Ensemble learning for heartbeat classification using adaptive orthogonal transformations, in *Comput. Aided Systems Theory — EUROCAST 2019*, eds. R. Moreno-Díaz *et al.*, Lecture Notes in Computer Science (Springer, 2020), pp. 355–363.
63. G. Szegő, *Orthogonal Polynomials*, 3rd edn. (AMS Colloquium Publications, New York, 1967).
64. L. Sörnmo, P. O. Borjesson, M.-E. Nygards and O. Pahlm, A method for evaluation of QRS shape features using a mathematical model for the ECG, *IEEE Trans. Biomed. Eng.* **BME-28**(10) (1981) 713–717.
65. P. Laguna, R. Jane, S. Olmos, N. V. Thakor, H. Rix and P. Caminal, Adaptive estimation of QRS complex wave features of ECG signal by the Hermite model, *Med. Biol. Eng. Comput.* **34**(1) (1996) 58–68.
66. M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandth and L. Sörnmo, Clustering ECG complexes using Hermite functions and self-organizing maps, *IEEE Trans. Biomed. Eng.* **47**(7) (2000) 838–848.
67. H. Haraldsson, L. Edenbrandt and M. Ohlsson, Detecting acute myocardial infarction in the 12-lead ECG using Hermite expansions and neural networks, *Artif. Intell. Med.* **32**(2) (2004) 127–136.
68. A. Sandryhaila, S. Saba, M. Püschel and J. Kovacevic, Efficient compression of QRS complexes using Hermite expansion, *IEEE Trans. Signal Process.* **60**(2) (2012) 947–955, doi: 10.1109/TSP.2011.2173336.
69. P. Kovács, C. Böck, T. Tschoellitsch, M. Huemer and J. Meier, Diagnostic quality assessment for low-dimensional ECG representations, *Comput. Biol. Med.* **150** (2022) 106086.
70. M. Llamedo and J. P. Martinez, Heartbeat classification using feature selection driven by database generalization criteria, *IEEE Trans. Biomed. Eng.* **58**(3) (2011) 616–625.
71. C.-C. Lin and C.-M. Yang, Heartbeat classification using normalized RR intervals and morphological features, *Math. Probl. Eng.* **2014** (2014) 1–11.
72. J. He, J. Rong, L. Sun, H. Wang, Y. Zhang and J. Ma, A framework for cardiac arrhythmia detection from IoT-based ECGs, *World Wide Web* **23** (2020) 2835–2850.
73. Y. Li, Z. He, H. Wang, B. Li, F. Li, Y. Gao and X. Ye, CraftNet: A deep learning ensemble to diagnose cardiovascular diseases, *Biomed. Signal Process. Control* **62** (2020) 102091.
74. Y. Li, R. Qian and K. Li, Inter-patient arrhythmia classification with improved deep residual convolutional neural network, *Comput. Methods Programs Biomed.* **214** (2022) 106582.
75. Y. Gan, J. Cheng Shi, W. Ming He and F. Jia Sun, Parallel classification model of arrhythmia based on DenseNet-BiLSTM, *Biocybern. Biomed. Eng.* **41**(4) (2021) 1548–1560.
76. J. Zhang, A. Liu, D. Liang, X. Chen and M. Gao, Interpatient ECG heartbeat classification with an adversarial convolutional neural network, *J. Healthc. Eng.* **2021**(1) (2021) 9946596.
77. L. Niu, C. Chen, H. Liu, S. Zhou and M. Shu, A deep-learning approach to ECG classification based on adversarial domain adaptation, *Healthcare* **8**(4) (2020) 1–16.
78. X. Qiao, K. Lee, S. A. Mokhtar, I. Ismail, A. L. B. Md Pauzi, Q. Zhang and P. Y. Lim, Deep learning-based ECG arrhythmia classification: A systematic review, *Appl. Sci.* **13**(8) (2023) 1–25.
79. D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
80. A. H. M. S. Hoque, G. Bognár and S. Fridli, Radial harmonic fourier moments for CT-based quantitative radiomics, *Acta Cybern.* **27**(2) (2025) 175–196.
81. Z. Boga, C. Sándor and P. Kovács, A multidimensional particle swarm optimization-based algorithm for brain MRI tumor segmentation, *Sensors* **25**(9) (2025) 2800.
82. G. Fu, J. Zheng, I. Abudayyeh, C. Ani, C. Rakovski, L. Ehwerhemuepha, H. Lu, Y. Guo, S. Liu, H. Chu and B. Yang, CardioGPT: An ECG interpretation generation model, *IEEE Access* **12** (2024) 50254–50264.
83. M. H. Rafiei, L. V. Gauthier, H. Adeli and D. Takabi, Self-supervised learning for electroencephalography, *IEEE Trans. Neural Netw. Learn. Syst.* **35**(2) (2024) 1457–1471.
84. M. H. Rafiei, L. V. Gauthier, H. Adeli and D. Takabi, Self-supervised learning for near-wild cognitive workload estimation, *J. Med. Syst.* **48**(107) (2024) 1–21.
85. K. M. R. Alam, N. Siddique and H. Adeli, A dynamic ensemble learning algorithm for neural networks, *Neural Comput. Appl.* **32** (2020) 8675–8690.
86. M. H. Rafiei and H. Adeli, A new neural dynamic classification algorithm, *IEEE Trans. Neural Netw. Learn. Syst.* **28**(12) (2017) 3074–3083.