OPEN ACCESS
International Journal of Neural Systems, Vol. 32, No. 1 (2022) 2150054 (19 pages)
© The Author(s)
DOI: 10.1142/S0129065721500544

VPNET: Variable Projection Networks

Péter Kovács[*,¶], Gergő Bognár[*,†,‡], Christian Huber[§] and Mario Huemer[†,‡]
[*]*Department of Numerical Analysis, Eötvös Loránd University*
Pázmány Péter stny. 1/C, Budapest 1117, Hungary

[†]*Institute of Signal Processing*
Johannes Kepler University Linz
Altenberger str. 69, Linz 4040, Austria

[‡]*JKU LIT SAL eSPML Lab, Silicon Austria Labs*
Altenberger str. 69, Linz 4040, Austria

[§]*Embedded AI Research Group, Silicon Austria Labs GmbH*
Altenberger str. 69, Linz 4040, Austria
[¶]*kovika@inf.elte.hu*

Received 18 February 2021
Accepted 1 September 2021
Published Online 13 October 2021

In this paper, we introduce VPNet, a novel model-driven neural network architecture based on variable projection (VP). Applying VP operators to neural networks results in learnable features, interpretable parameters, and compact network structures. This paper discusses the motivation and mathematical background of VPNet and presents experiments. The VPNet approach was evaluated in the context of signal processing, where we classified a synthetic dataset and real electrocardiogram (ECG) signals. Compared to fully connected and one-dimensional convolutional networks, VPNet offers fast learning ability and good accuracy at a low computational cost of both training and inference. Based on these advantages and the promising results obtained, we anticipate a profound impact on the broader field of signal processing, in particular on classification, regression and clustering problems.

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

1. Introduction

Until recently, signal processing was dominated by conventional model-based algorithms, which rely on mathematical and physical models of the real world. They are inherently interpretable and often incorporate domain knowledge such as statistical assumptions, smoothness, structure of the model space, and origin of the noise. However, this approach can become mathematically intractable if problems are complex. Machine learning (ML) provides an alternative approach to this challenge by building data-driven mathematical models. Neural networks (NNs) and supervised learning in particular offer a proper framework for various signal-processing problems.[1] Below, we briefly review a few recent trends that served as motivation for developing the proposed variable projection network (VPNet).

[¶]Corresponding author.

A traditional ML approach is to decompose the problem into separate feature extraction and learning steps.[2] In this case, the data is preprocessed in order to extract static features based on the given domain knowledge. These features are inputs to conventional ML algorithms. Although the dimension of the original data is significantly reduced in the first step, these handcrafted features are usually suboptimal with respect to the whole learning process.[3] Deep learning provides alternatives to the traditional approach, overcoming some of its drawbacks.[1] Learned features of NNs can be used as input for non-NN methods, like discriminant correlation filters, as well.[4] Reference 5 combined traditional kernel-based Support Vector Machines (SVMs) with deep learning approaches. Another common method would be to use the features as input for one or multiple other NN for multi-target prediction.[6–9]

Using more hidden layers in deep neural networks (DNNs) increased the learning abilities of NNs.[10] This enables DNNs to use the first layers for feature extraction and further layers for performing operations on the features learned. Convolutional neural networks (CNNs) are special, optionally deep architectures and are the leading ML approaches in 2D and 3D image processing and computer vision.[11–16] Here, the built-in feature extraction layers perform multiple convolutional filtering and dimension-reduction (pooling) steps. Despite their advantages, DNNs and CNNs continue to raise several concerns. Their improved efficiency comes at the cost of higher computational complexity and numerical difficulties in the training process (see, e.g. overfitting and divergence). Due to the large number of nonlinear connections between the model parameters, DNN and CNN approaches can be considered as black-box methods, where the parameters have no or little physical meaning and are difficult or impossible to interpret. Additionally, training these networks requires vast amounts of labeled data, which is problematic to collect in many applications, such as telecommunications,[17] and biomedical engineering.[18,19] Although data augmentation, transfer learning, outlier removal, and ensemble methods can mitigate this problem, reducing the data hunger of deep learning approaches is still a major challenge in this field.

Despite the popularity of deep learning, traditional ML algorithms continue to dominate in many 1D signal-processing tasks,[20] especially in biomedical signal classification, for example, of electroencephalograms (EEGs), electromyograms (EMGs), and ECGs. The main reason for this lies in the nature of clinical applications, where both accuracy and explainability are important. These cannot be guaranteed by the previously mentioned NN approaches, since they do not extract medically interpretable features. VPNet, however, breaks this impasse by harnessing the theory of variable projection (VP) to provide a framework for solving nonlinear least-squares problems, whose parameters can be separated into linear and nonlinear ones. In many fields of signal processing, there are a large number of linear parameters, which are driven by a smaller number of nonlinear variables (see Eq. (3)). For example, signal compression, representation, and feature-extraction algorithms are often based on linear coefficients of some transformation, such as Fourier and wavelet transforms, which can be parameterized via properties of the window function, mother wavelet, etc.

The VPNet was designed to merge the expert knowledge used by traditional model-based approaches with the learning abilities of NNs. The proposed architecture is inspired by the so-called model-driven NN concept, which is an emerging trend in signal processing. In Sec. 2, we review the existing literature on incorporating model-based information into machine learning. The theoretical background, the general formulation of VPNet, and the corresponding forward and backpropagation algorithm are discussed in Sec. 3. Section 4 describes multiple experiments we performed to evaluate and compare the performance of VPNet to that of other NNs. Finally, Sec. 5 presents conclusions and the expected broader impact of our research.

2. Related Works

Approximation theory gives a general framework to approach the fundamental task in machine learning that is to learn a good representation of the data.[3] Classical methods in approximation theory build up complicated functions by using linear combinations of elementary functions, whereas neural networks use compositions of simple functions. The structure of these compositions constrains the feasible region where we search for the solution of the corresponding

ML task. The model-driven NN concept implements these constraints such that the design of the NN architecture resembles the solution to well understood mathematical problems, such as ordinary or partial differential equations[21,22] (ODE, PDE), signal[23–25] and image[26–28] processing, optimization,[29] and control.[30–32]

ODE- and PDE-constrained learning strategies belong to a family of model-driven ML techniques that relates the rigorous mathematical background of differential equations to deep learning problems. On the one hand, numerical solvers provide various ways to derive and to interpret the output of NN architectures, such as residual neural networks,[21] Hamiltonian networks,[22] based on the discretization scheme of the corresponding ODE and PDE. On the other hand, deep learning can incorporate domain knowledge automatically which would otherwise require a significant human effort,[4,26,28] e.g. good insights into the problem, and mathematical formulation of *a priori* information. Although this approach does not necessarily reduce the number of trainable weights, it helps to design reversible architectures that allow for memory-efficient implementations.[33]

Another branch of model-driven NNs, such as deep unfolding[23] or Wiener-,[32] and Hammerstein-type[25] NNs, originates from signal processing problems. The former approach unfolds the iterations of classical model-based algorithms into layer-wise NN structures whose parameters are optimized based on the training data. This way the resulting NN retains the powerful learning ability of DNNs, inherits expert knowledge, and reduces the size of the training data.[17] Wiener-[32] and Hammerstein-type[25] NNs are alternatives that combine the advantages of model-based methods and deep learning techniques. These networks comprise cascades of static nonlinear elements and dynamic linear blocks that represent NNs and linear time-invariant (LTI) systems, respectively. Recently, these methods have shown great potential in many fields, for instance, in system identification,[25] control engineering,[32] sparse approximation theory,[34,35] and telecommunication.[36,37]

The motivation behind integrating optimization problems into DNN architectures is similar to the ODE/PDE-driven networks, namely, designing optimization problems to real-world processes is a labor-intensive work which also needs expert knowledge. To date, several new NN architectures have been proposed in order to learn these optimization problems automatically from data. Solving ill-posed inverse problems is a typical example for such neural networks. In this case, each layer is constrained by a penalized linear least-squares problem where the parameters of the regularization term, such as threshold values, linear kernels, weights of the shrinkage functions, constitute the trainable weights.[27,38,39] OptNet[29] gives the most general framework in this family, where the layers encode convex quadratic programming (QP) problems. The Hessian matrix of the QP's objective function along with its equality and inequality constraints are learnable parameters. The representation power of an OptNet layer is higher than that of the two-layer ReLU networks, which can reduce the overall depth of DNN architectures (see Theorems 2 and 3 in Ref. 29). Besides its advantages, the forward/backward passes of an OptNet layer are much more computationally expensive than a linear or convolutional layer. This is due to the fact that constrained QP problems have no closed form solution in general, thus the forward pass requires the use of iterative numerical solvers in each layer for each update. We acknowledge that there are many other model-based[40] and model-free approaches.[41–45] Especially, for time series data there are methods based on spiking neural networks[46,47] including their variations[48,49] which are beyond the scope of this paper.

To the best of our knowledge, this is the first time that the VP operators have been exploited in the context of learning end-to-end systems. However, we note that the proposed VPNet can be considered a special case of OptNet. Indeed, a VP layer forwards the solution of an unconstrained separable nonlinear least-squares (SNLLS) problem to the next layer (cf. Eq. (1) in Ref. 29). The corresponding nonlinear parameters are the trainable weights of the VP layer, and the linear ones are the extracted features, which are forwarded to the next layer. In contrast to a general OptNet layer, both the solution and the gradients of a VP layer can be calculated analytically that is provided by the theoretical framework of variable projection.[50] This speeds up the training and the inference, which can be further improved by the use of orthogonal and discrete orthogonal function systems (see, e.g. Sec. 3.3).

3. Variable Projection Networks

3.1. *Variable projections*

VP[50] provides a framework for addressing nonlinear modeling problems of the form

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

where $x \in \mathbb{R}^m$ and $\Phi_k \in \mathbb{R}^m$ denote the input data to be approximated and a parametric function system, respectively. The symbol $\Phi(\theta)$ refers to both the function system itself and a matrix of size $\mathbb{R}^{m\times n}$. The linear parameters $c \in \mathbb{R}^n$ and the nonlinear parameters $\theta \in \mathbb{R}^p$ of the function system Φ are separated. The least-squares fit of this problem means minimization of the nonlinear functional

$$r(c,\theta) := \|x - \Phi(\theta)c\|_2^2.$$

Without nonlinear parameters (i.e. if θ is fixed), the model is linear in the coefficients c. The minimization of r with respect to c leads to the well-known linear least-squares approximation. Note that it is in fact the best approximation problem in Hilbert spaces. The optimal solution can be expressed by means of Fourier coefficients and orthogonal projection operators $\mathcal{P}_{\Phi(\theta)}$:

$$c = \Phi^+(\theta)x, \quad \hat{x} = \mathcal{P}_{\Phi(\theta)}x = \Phi(\theta)\Phi^+(\theta)x, \qquad (2)$$

where $\Phi^+(\theta)$ denotes the Moore–Penrose pseudoinverse of matrix $\Phi(\theta)$. The concept is closely related to mathematical transformation methods, such as Fourier and wavelet transforms, that can be interpreted as orthogonal projections by a given function system with a predefined θ. From a practical point of view, the coefficients c can be interpreted as features extracted by VP, and $\hat{x}$ is a result of low-pass filtering and dimension reduction. The minimization of r in the general case can be decomposed into the minimization by the nonlinear parameters θ, while the linear parameters c are computed by the orthogonal projection. Thus, according to the work of Golub and Pereyra,[50] minimizing r is equivalent to minimizing the following VP functional:

$$r_2(\theta) := \|x - \Phi(\theta)\Phi^+(\theta)x\|_2^2. \qquad (3)$$

In Ref. 51, a robust gradient-based Matlab implementation was provided for the numerical optimization of r_2. Mathematically, VP is a formalization for adaptive orthogonal transformations that allows filtering and feature extraction by means of parametric function systems. If a nonlinear optimization problem can be separated into linear and nonlinear parameters, VP may also act as a solver, which opens up other possible applications.[52,53]

In the ML context, VP can be used as a feature extraction method and as a modeling technique for the training procedure.[54] Pereyra *et al.* proposed VP as an optimization method for a given class of feedforward NNs. They modeled the whole network with VP and used the VP optimization method from Ref. 50 as an alternative to stochastic gradient methods. This methodology is, however, limited to NNs with only one hidden layer. Approaching VP from a different and novel direction, based on its feature extraction ability, we introduce VPNet.

Previous results have shown that several biomedical signal-processing problems can be addressed efficiently with variable projection by means of adaptive rational and Hermite functions as well as B-splines.[55,56] VP features have been used in particular for ECG and EEG representation, compression, classification, and segmentation.[57–65] The results show that VP provides a very compact, yet morphologically accurate, representation of signals with respect to the target problem. Additionally, the nonlinear parameters themselves carry direct morphological information about the signals, and they are usually human-interpretable.

3.2. *VPNet architecture*

The key idea of this architecture is to create a network that combines the representation abilities of VP and the prediction abilities of NNs in the form of a composite model. The basic VPNet architecture is a feedforward NN, where the first layer(s) applies a VP operator that is forwarded to a fully connected, potentially deep NN (see Fig. 1). The construction is similar to that of CNNs in the sense that the first layer(s) of the network can be interpreted as a built-in feature extraction method. Note that more complex VPNet architectures are also possible, for instance, based on the models of U-Net[14] and AutoEncoder,[66] which will be investigated as part of our future work.

Depending on its target application, the VP layer we propose has two possible behaviors. It either

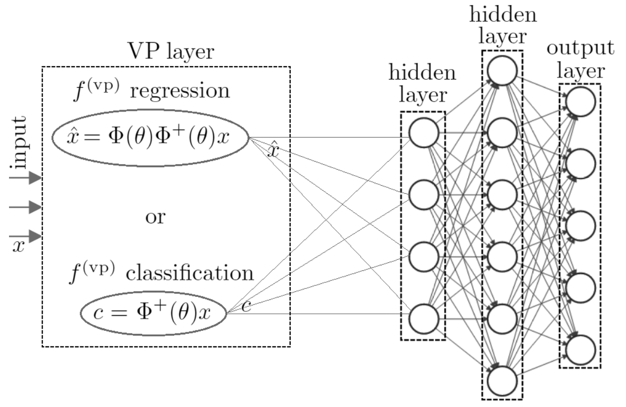

VP layer
$f^{(\mathrm{vp})}$ regression
$\hat{x} = \Phi(\theta)\Phi^{+}(\theta)x$
or
$f^{(\mathrm{vp})}$ classification
$c = \Phi^{+}(\theta)x$
input
x
$\hat{x}$
c
hidden layer
hidden layer
output layer

Fig. 1. VPNet architecture.

performs a filtering of the form

$$f^{(\mathrm{vp})}(x) := \Phi(\theta)\Phi^{+}(\theta)x = \hat{x} \quad (x \in \mathbb{R}^m), \tag{4}$$

or a feature extraction of the form

$$f^{(\mathrm{vp})}(x) := \Phi^{+}(\theta)x = c \quad (x \in \mathbb{R}^m), \tag{5}$$

where $\theta \in \mathbb{R}^p$ denotes the nonlinear system parameters of the given function system Φ, as defined above. These VP operators refer to the orthogonal projection and the general Fourier coefficients of the input x by means of the parametric system $\Phi(\theta)$, as in Eq. (2). The filter method may be better suited to regression problems, while the feature extraction is suitable for classification problems. The nonlinear system parameter vector θ comprises the learnable parameters of the VP layer. Note that many inverse problems[52] can be viewed as SNLLS data fitting problems including a small set of adjustable nonlinear parameters θ with direct physical interpretations. For instance, the function system $\Phi_k(t; \tau_k, \lambda_k) = \cos(\lambda_k t + \tau_k)$ can be used in frequency estimation and in EEG, where the network would learn dominant frequencies λ_k and phases τ_k that characterize a certain class of signals, such as seizures in EEG recordings.[67–69] MRI imaging is another setting,[70] where $\Phi_k(t; \lambda_k) = \exp(-\lambda_k t)$ with $\lambda_k \in \mathbb{R}^+$ yields information about the tissue type. The previously mentioned properties and advantages of the VP operator are implicitly built into VPNet:

- *Role*: A novel model-driven network architecture for 1D signal-processing problems.
- *Generality*: VPNet can be built from arbitrary parameterized function systems, which allows the direct incorporation of domain knowledge into the network.
- *Interpretability*: The VP layer can be explained as a built-in feature-extraction method. Further, the layer parameters are the nonlinear VP system parameters, which have an interpretable meaning. They are usually directly connected to morphological properties of the input data (see, e.g. Sec. 4.2).
- *Simplicity*: Since the VP layer is usually driven by only a few system parameters, VPNet may provide a compact alternative to CNNs and DNNs. In fact, the VP layer can significantly decrease the number of parameters in a DNN.

3.3. *VP forward propagation*

In order to calculate the forward pass of the VP layer, a linear least-squares (LLS) problem has to be solved for a certain value of θ in each training iteration (see Eqs. (4) and (5)). Several numerical methods exist to solve such problems, among which QR factorization and singular value decomposition (SVD) are the most common techniques. The QR method (requires $\sim 2mn^2 - 2n^3/3$ flops) is fast and reliable for well-conditioned problems, but may fail when $\Phi(\theta) \in \mathbb{R}^{m \times n}$ is nearly rank-deficient. Therefore, in our implementation, we utilize the SVD (requires $\sim 2mn^2 + 11n^3$ flops) that is the most stable way to solve unconstrained LLS problems.[71] Although it is computationally more demanding than the QR factorization in cases when $m \sim n$, their complexity is approximately the same if $m \gg n$. Note that the latter inequality usually holds in practice, since in VPNet m stands for the length of the input signal, which is much greater than the number of extracted features n.

The low computational complexity is based on the fact that the nonlinearity is precomputed and stored in the matrix Φ. As a consequence, during evaluation, the VP layer just performs a matrix multiplication. Further, since the number of features computed by the VP layer is typically very low, the following layers can have lower complexity as well. The weight matrix of a fully connected layer, following the VP layer, is element of $\mathbb{R}^{n \times l}$ instead of $\mathbb{R}^{m \times l}$ without the VP layer, with n is the number of coefficients, m is the length of the input signal and l is the number of neuron in the fully connected layer. Since n is usually by far smaller than m, the weight matrix is significantly smaller for a fixed number or neuron l.

For shallow neural networks, when only a few hidden layers are connected to the VP layer, solving the corresponding LLS problem in each training iteration is obviously the bottleneck of VPNet that influences both the computational complexity and the numerical accuracy. In the following, we provide a realization of the VP layer with Hermite functions, and we demonstrate how the choice of the function system and its parametrization influence the conditionality of $\Phi(\theta)$.

3.3.1. *Adaptive Hermite system*

In order to alleviate the computational burden of the VP layer, a straightforward option is to parametrize orthogonal function systems. As a case study, let us consider Hermite polynomials,[72] which are defined by the three-term recurrence relation:

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

where $H_0(t) = 1$ and $H_1(t) = 2t$. These classical orthogonal polynomials can be parametrized via dilation and translation:

$$\Phi_k(t; \tau, \lambda) = \sqrt{\lambda}\Phi_k(\lambda(t - \tau)), \qquad (6)$$

where

$$\Phi_k(t) = H_k(t)e^{-t^2/2}/\sqrt{\pi^{1/2}2^k k!} \quad (k \in \mathbb{N}^+). \qquad (7)$$

The functions $\Phi_k(t; \tau, \lambda)$ are the translated and dilated variations of the well-known Hermite functions, thus we refer to them as “adaptive Hermite functions”.

The forward propagation of the corresponding Hermite-VP layer can be defined by the matrix $\Phi(\theta)$ in Eq. (3). For a given parameter value $\theta = (\tau, \lambda)$, the kth column of $\Phi(\theta)$ is equal to the values of the kth adaptive Hermite function evaluated at some predefined points $t_0, t_1, \ldots, t_{m-1} \subseteq [a; b]$, where $[a; b]$ stands for the sampling interval. In the case of proper discretization,[73] the columns of $\Phi(\theta)$ are pairwise orthogonal and unit vectors for all θ; therefore, $\Phi^+(\theta) = \Phi^T(\theta)$, which speeds up the computation of both the forward and the backward passes.

There are two strategies for choosing the discretization points: nonuniform and uniform sampling. The former relies on the Gauss–Hermite quadrature rules, which associates the points $t_0, t_1, \ldots, t_{m-1} \subseteq [a; b]$ with the roots of Hermite polynomials.[74] This approach is the most accurate way to define discrete orthogonal systems, but it requires both the precomputation of the roots and the resampling of the input signals at these nonequidistant points. Therefore, we consider the computationally simpler uniform discretization instead. This sampling scheme, although less accurate, satisfies discrete orthogonality, and thus the identity $\Phi^+(\theta) = \Phi^T(\theta)$ holds, provided that the number of sampling points m is large enough, and $\theta \in \Gamma$, where

$$\Gamma = \left\{(\tau, \lambda) \in \mathbb{R} \times \mathbb{R}_+ : \tau + \frac{3}{\lambda} \leq b,\ \tau - \frac{3}{\lambda} \geq a\right\}.$$

If $\theta \notin \Gamma$, it can happen that the adaptive Hermite functions $\Phi_k(t, \tau, \lambda)$ are not discrete orthogonal anymore. In the worst-case scenario, they can be linearly dependent, which results in a rank deficient matrix $\Phi(\theta)$. In Fig. 2, we demonstrate this phenomenon by evaluating the condition number of $\Phi(\theta) \in \mathbb{R}^{m \times n}$ for $m = 1000$, $n = 3$, and for a range of parameters $\theta = (\tau, \lambda) \in [500, 1100] \times [0.05, 0.012]$. It can be seen that the condition number diverges from the ideal case (green dashed line) as we change τ and λ irrespective of Γ. This can be avoided if we choose the parameters from the feasible region Γ. The rationale behind this behavior is given in Appendix A.

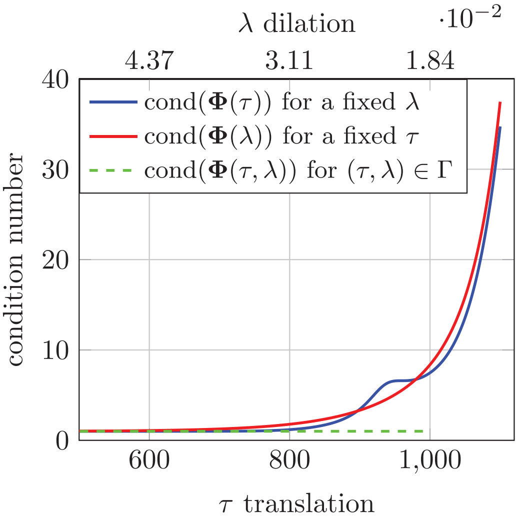


Fig. 2. Relationship between the parameters τ, λ and the condition number of the matrix $\mathbf{\Phi}(\tau, \lambda)$.

3.4. *VP backpropagation*

Let us discuss the training of a general feedforward NN in a supervised manner. Let

$$(x_i, y_i) \quad (i = 1, 2, \ldots, N)$$

be the annotated input-target pairs of the training data, where the input vector $x_i \in \mathbb{R}^m$ and the target vector $y_i \in \mathbb{R}^s$ (in the case of regression) or the target label $y_i \in \mathbb{N}$ or probabilities $y_i \in [0,1]^c$ (in the case of classification). A general feedforward NN can be expressed as the composition of layer functions of the form

$$NN_{\boldsymbol{\theta}}(x) = \big(f^{(L)}_{\theta^{(L)}} \circ \cdots \circ f^{(\ell)}_{\theta^{(\ell)}} \circ \cdots \circ f^{(2)}_{\theta^{(2)}} \circ f^{(1)}_{\theta^{(1)}}\big)(x),$$

where $x \in \mathbb{R}^m$ stands for the input samples, $f^{(\ell)}_{\theta^{(\ell)}}$ and $\theta^{(\ell)}$ denote the function and the parameters of layer ℓ, respectively. The symbol $\boldsymbol{\theta}$ refers to the set of parameters $\theta^{(\ell)}$. The layer functions $f^{(\ell)}$ may refer to linear mappings, convolutional filters, nonlinear activations, pooling, VP operators, etc. Let

$$\hat{y}_i := NN_{\boldsymbol{\theta}}(x_i) \quad (i = 1, 2, \ldots, N)$$

denote the predicted values for each input. The training of the network can be addressed as a minimization problem, involving a proper loss (i.e. cost) function J that evaluates the error between predicted and target values. Common loss functions are the Mean Squared Error (MSE), that is, the least-squares cost function (regression problems, $y_i \in \mathbb{R}^s$), and the Binary Cross Entropy (BCE) loss (binary classification, $y_k \in \{0,1\}$):

$$J_{\text{MSE}}(\boldsymbol{\theta}) := \frac{1}{N}\sum_{i=1}^{N} \|y_i - \hat{y}_i\|_2^2,$$

$$J_{\text{BCE}}(\boldsymbol{\theta}) := -\frac{1}{N}\sum_{i=1}^{N}(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)).$$

In our experiments, we used the Cross Entropy loss J_{CE}, which is the multi-class extension of BCE (classification, $y_k \in \mathbb{N}$); see also, Sec. 4.

The state-of-the-art method for training feedforward networks is backpropagation,[75] where J is minimized by means of a stochastic gradient-descent optimization (see, e.g. Adam,[76] Adagrad,[77] RMSprop[78]). There are multiple implementation of the propagation algorithm for different programming languages, target hardware platforms and machine learning frameworks.[79–82] The gradient descent update formula for each layer parameter is

$$\theta^{(\ell)} := \theta^{(\ell)} - \eta\frac{\partial J}{\partial \theta^{(\ell)}},$$

where $\eta > 0$ is called the learning rate. Briefly, backpropagation provides a recursive way of computing the gradients above based on the chain rule:

$$\frac{\partial J}{\partial f^{(\ell-1)}} = \frac{\partial J}{\partial f^{(\ell)}} \cdot \frac{\partial f^{(\ell)}}{\partial f^{(\ell-1)}},$$

$$\frac{\partial J}{\partial \theta^{(\ell)}} = \frac{\partial J}{\partial f^{(\ell)}} \cdot \frac{\partial f^{(\ell)}}{\partial \theta^{(\ell)}}.$$

This way, only the partial derivatives of the layer function $f^{(\ell)}$ with respect to its input $(\partial f^{(\ell)}/\partial f^{(\ell-1)})$ and to its parameters $(\partial f^{(\ell)}/\partial \theta^{(\ell)})$ must be calculated. These derivatives are usually well known for the common layer types and can also be directly calculated for the VP layers. Based on Ref. 50, the partial derivatives of the VP operators with respect to their input and nonlinear parameters can be expressed as follows. In the case of a filtering-type VP layer (see Eq. (4)):

$$f^{(\text{vp})}(x) = \Phi(\theta)\Phi^+(\theta)x, \quad \frac{\partial f^{(\text{vp})}}{\partial x} = [\Phi(\theta)\Phi^+(\theta)]^T,$$

$$\frac{\partial f^{(\text{vp})}}{\partial \theta_j} = \frac{\partial[\Phi(\theta)\Phi^+(\theta)]}{\partial \theta_j}x,$$

where

$$\partial[\Phi(\theta)\Phi^+(\theta)] = (I - \Phi\Phi^+)\partial\Phi\Phi^+ + [(I - \Phi\Phi^+)\partial\Phi\Phi^+]^T.$$

In the case of a feature-extraction-type VP layer (see Eq. (5)):

$$f^{(\text{vp})}(x) = \Phi^+(\theta)x, \quad \frac{\partial f^{(\text{vp})}}{\partial x} = [\Phi^+(\theta)]^T,$$

$$\frac{\partial f^{(\text{vp})}}{\partial \theta_j} = \frac{\partial \Phi^+}{\partial \theta_j}x,$$

where

$$\partial\Phi^+ = -\Phi^+\partial\Phi\Phi^+ + \Phi^+[\Phi^+]^T\partial\Phi^T(I - \Phi\Phi^+) + (I - \Phi^+\Phi)\partial\Phi^T[\Phi^+]^T\Phi^+.$$

The naive implementation of the backpropagation, particularly in the case of DNNs, can lead to numerical issues, such as divergence and overfitting. In order to avoid this, a regularization term in the form of an ℓ_2 penalty on the weight parameters is added to the loss.[66] Here, we introduce a percent root-mean-square difference (PRD) regularization that can be applied to a single feature-extraction VP layer in the case of a classification problem. The modified loss function we propose is

$$\begin{aligned} J_{\mathrm{VP}}(\boldsymbol{\theta}) &:= J_{\mathrm{CE}}(\boldsymbol{\theta}) + \frac{\alpha}{N}\sum_{i=1}^{N} \frac{r_2(x_i;\theta^{(\mathrm{vp})})}{\|x_i\|_2^2} \\ &= J_{\mathrm{CE}}(\boldsymbol{\theta}) + \frac{\alpha}{N}\sum_{i=1}^{N} \frac{\|x_i - \Phi(\theta^{(\mathrm{vp})})\Phi^{+}(\theta^{(\mathrm{vp})})x_i\|_2^2}{\|x_i\|_2^2}, \end{aligned}$$

where $\alpha \geq 0$ controls the penalty effect. The motivation behind this regularization is twofold: First, it is based on the previous results that incorporate VP as feature extraction, which show that the precise VP approximation may lead to "good" features and therefore to high classification accuracy. Second, we expect that the optimal VPNet classifier extracts the main characteristics of the input signals, which means that we presume "good" approximation. This penalty term seemingly breaks the formulation of the backpropagation, but the original method can easily be extended by a bypass step that is applied to the VP layer only. The gradient with respect to the VP parameters is modified as follows:

$$\frac{\partial J_{\mathrm{VP}}}{\partial \theta^{(\mathrm{vp})}} = \frac{\partial J_{\mathrm{CE}}}{\partial \theta^{(\mathrm{vp})}} + \frac{\alpha}{N}\sum_{i=1}^{N} \frac{1}{\|x_i\|_2^2} \cdot \frac{\partial r_2}{\partial \theta^{(\mathrm{vp})}},$$

where

$$\partial r_2 = -2x_i^T(I - \Phi\Phi^{+})\partial\Phi\Phi^{+}x_i.$$

We just developed the formulas for attaining the necessary gradient information for training VPNet via backpropagation. This allows us to train VPNets in the same way as convolutional and fully connected NNs.

4. Experiments

Using supervised classification problems inspired by particular biomedical signal-processing applications, we evaluated VPNet and compared it to fully connected and 1D convolutional networks. We present the details of the experiments, specifically about the network architectures, the VP system of choice, and the synthetic and real datasets.

4.1. *Network architecture*

Here, we provide details about the networks we compared, the learning methods, and the network parameters. The networks were feedforward, consisting of the following layers:

- *VPNet*: a VP layer, a fully connected (FC) layer with ReLU activation, an FC layer with SoftMax activation.
- *Fully connected NN*: one or two FC layers with ReLU, an FC layer with SoftMax.
- *CNN*: a 1D convolutional and pooling layer, an FC layer with ReLU, an FC layer with SoftMax.

For signal-classification tasks, the inputs were $\mathbb{R}^m$ samples and the outputs were interpreted as a probability distribution over predicted output classes. The FC layers performed linear mappings with nonlinear activation (ReLU or SoftMax). The VP layer was of the feature-extraction type (see Eq. (5)), and the CNN implemented 1D convolution and mean or maximum pooling as in Ref. 18.

Based on cross entropy loss with VP regularization (see Sec. 3.4), offline backpropagation with Adam optimizer[76] was applied for learning. The hyperparameters and the parameter selection strategies were as follows:

- *Learning parameters*: learning rate, VP penalty (VPNet only), batch size, and the number of epochs. The last two were fixed (512 and 10–100, respectively). The optimal learning rate and penalty can be found by a grid search.
- *Network parameters*: number of layers, number of neurons, VP dimension n (VPNet only), convolutional and pooling kernel sizes (CNN only). Here, we either used fixed dimensions so that the three architectures are comparable or evaluated possible configurations by a grid search.
- *Layer parameters*: linear weights and biases, nonlinear VP parameters (VPNet only), kernel weights and biases (CNN only). These parameters were optimized by backpropagation. Initialization was random for the linear and kernel parameters. However, the VP parameters have interpretable meaning, which may lead to special initialization. We investigated two options: a grid search on the intervals of possible values and initialization by means of pretraining the VP layer to reconstruct input data (i.e. minimizing r_2 in Eq. (3)). The

latter approach is especially useful in the case of complex waveforms which possibly need more VP parameters to learn.

4.2. *VP system of choice*

Although Hermite functions have shown great potential in many fields, such as molecular biology,[83] computer tomography,[84] radar,[85] and physical optics,[86] their main application area is 1D biomedical signal processing. The shape features of Hermite functions are well suited to producing models of compactly supported waveforms such as spikes,[87–91] which is why we used them in ECG heartbeat classification.

The nonlinear parameters τ and λ in Eq. (6) represent the time shift and the width of the modeled waveforms, respectively. Thus, the network learns the positions and the shapes of those waves/spikes which separate one class from another. For instance, in electrocardiography, a heartbeat signal comprises three individual waveforms (i.e. the QRS, T, and P waves), which represent different phases of the cardiac cycle, and their properties are directly used by medical experts for diagnosis. These features are learned by the VP layer: The amplitude and shape information is extracted by the linear coefficients c_k, while position and width of the waves are represented by τ and λ (see Fig. 3). This approach is essentially different from CNN-based methods, where no direct connections exist between learned and medical descriptors.

4.3. *Synthetic data*

Our goal was, on the one hand, to synthesize a dataset where we know the actual structure of the data depending on the generator parameters. On the other hand, the dataset had to have practical relevance (i.e. be related to actual signal-processing problems). The generator system of choice was the adaptive Hermite system, which seemed to fulfill these expectations due to its applications in signal processing (see Sec. 4.2). The principles we followed to generate the dataset are discussed in what follows.

Let us consider a general signal model by means of a linear combination of adaptive Hermite functions of the form

$$x_i = \Phi(\tau_i, \lambda_i) \cdot c^{(i)} = \sum_{k=0}^{n-1} c_k^{(i)} \Phi_k(\tau_i, \lambda_i),$$

where (τ_i, λ_i) and $c^{(i)}$ $(i = 1, 2, \ldots, M)$ refer to the sample-specific nonlinear parameters and coefficients, respectively. Based on the completeness of the Hermite system in $L^2(\mathbb{R})$, this formula provides a general approximation for arbitrary signals. However, the signal-processing applications of VP and the Hermite system show that proper selection of the nonlinear parameters may lead to accurate low-order approximations. Further investigation into this topic revealed that the nonlinear parameters correspond to coarse changes in the signal morphologies, while the coefficients reflect fine details.[92] For instance, we refer to Ref. 56, where the nonlinear parameters were utilized as global, patient-specific and the coefficients as heartbeat-specific descriptors. Motivated by these aspects, we sought to construct a dataset where the nonlinear parameters are close to each other and the coefficients form noticeably separable classes.

More precisely, we considered five coefficients (i.e. $c^{(i)} \in \mathbb{R}^5$) so that the points $(c_1^{(i)}, c_2^{(i)}, c_3^{(i)}) \in \mathbb{R}^3$

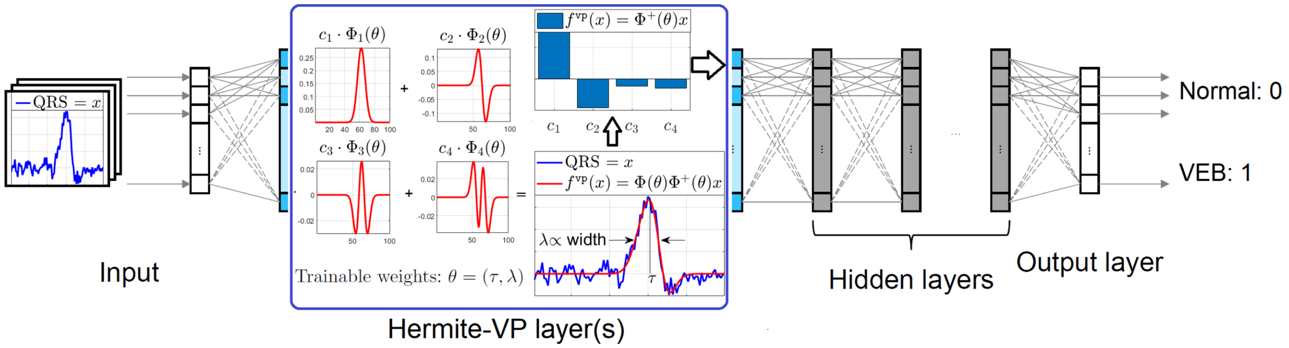
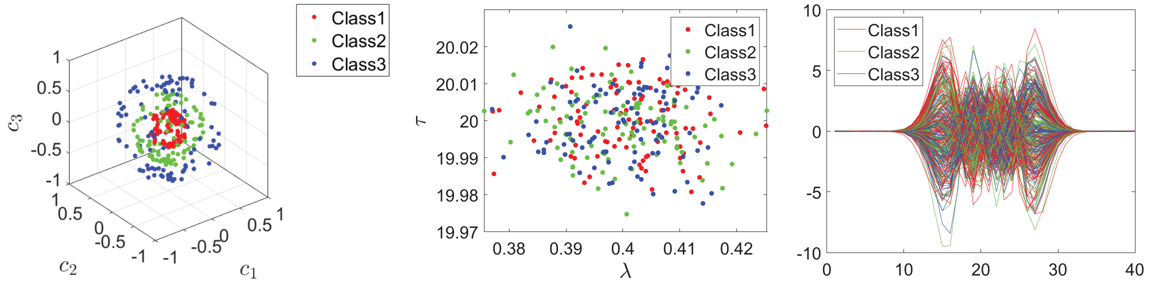

QRS = x
Input
$c_1 \cdot \Phi_1(\theta)$
0.25
0.2
0.15
0.1
0.05
20 40 60 80 100
+
$c_2 \cdot \Phi_2(\theta)$
0.1
0.05
0
-0.05
-0.1
50 100
$c_3 \cdot \Phi_3(\theta)$
0.02
0
-0.02
50 100
+
$c_4 \cdot \Phi_4(\theta)$
0.02
0
-0.02
50 100
=
$f^{\mathrm{vp}}(x) = \Phi^{+}(\theta)x$
c_1 c_2 c_3 c_4
QRS = x
$f^{\mathrm{vp}}(x) = \Phi(\theta)\Phi^{+}(\theta)x$
$\lambda \propto$ width
τ
Trainable weights: $\theta = (\tau, \lambda)$
Hermite-VP layer(s)
Hidden layers
Output layer
Normal: 0
VEB: 1

Fig. 3. VPNet architecture for QRS classification: the VP layer takes the whole signal as input, decomposes the QRS complexes into linear combinations of adaptive Hermite functions, and then forwards the coefficients of the Hermite components to the next fully connected layer.



(a) Coefficients.

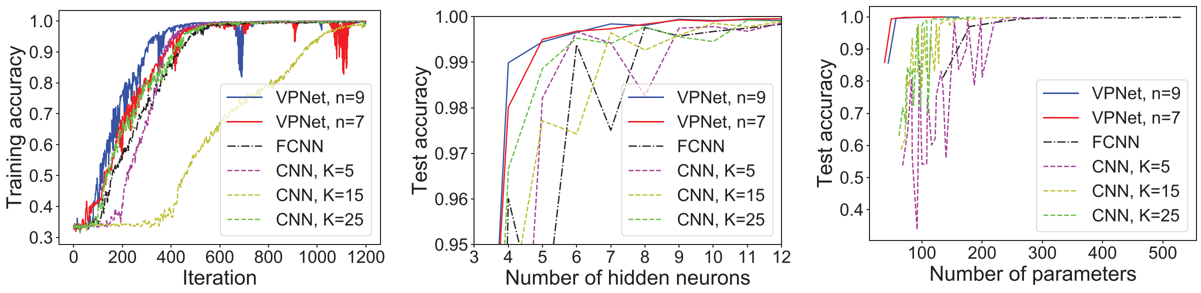
(b) System parameters.

(c) Samples.

Fig. 4. Synthetic dataset: first, the coefficients (a) and the parameters of the Hermite functions (b) are generated; which are then used to compute the input samples (c).

formed three separable spherical shells that correspond to the target class labels (see Fig. 4(a)). The motivation behind spherical shells was twofold. They are simple enough for human interpretation, but sufficiently complex to require complex networks. The last two coefficients, $c_4^{(i)}$ and $c_5^{(i)}$, served as random factors and for amplitude normalization. Their effect is to mislead the classifier, but at the same time to decrease the chance of overfitting. The nonlinear parameters τ_i and λ_i are similar for each sample up to a random factor, and the sample-specific parameter values are generated randomly with given mean and variance (see Fig. 4(b)). This random factor simulates the nonlinear noise in the measurement. Figure 4(c) presents the samples. We conclude that the simulation met our expectations: the resulting samples were difficult to separate, but the underlying structure was easy to interpret. Note that this is a standard process to generate synthetic data which was utilized by other authors as well.[93]

In the actual implementation, 5000 samples per class were generated for both the training and test sets. We evaluated a total of more than 8000 possible hyperparameter configurations of the three network architectures. A range of numbers of neurons in the hidden layer, various numbers of VP dimensions, and various CNN kernel and pooling sizes, learning rates and VP initializations were considered. The VP penalty was initially fixed to 0.1. The simulations showed that the VP regularization can not only increase the learning speed, but also ensure convergence of an otherwise divergent configuration. In this regard, 0.1 was found to be a good choice. The aggregated results are presented in Figs. 5(a) and 5(b). Therefore, the configurations are grouped into six categories: VPNets of dimension $n = 7$ and $n = 9$ in Eq. (1), fully connected NNs (FCNN), and CNNs with kernel sizes of 5, 15, and 25. Figure 5(a) shows the training accuracy curves corresponding to the best hyperparameter combination in each category. In Figs. 5(b) and 5(c), the best test accuracies are plotted against the number of neurons in the hidden layer and the total number of learnable network parameters, respectively, for each category. We note that the y-axis of Fig. 5(b) is restricted to the interval between 95% and 100% for better



(a) Best training curves.

(b) Best test accuracies.

(c) Best test accuracies.

Fig. 5. Evaluation on synthetic data.

visual interpretability. In the following, we compare the performance of VPNet with respect to different network complexities.

The results demonstrate the efficiency and potential capabilities of VPNet. Figure 5(a) indicates its fast learning ability. In fact, VPNet may converge faster than the other network architectures. Figures 5(b) and 5(c) show that VPNet can potentially outperform FCNNs and CNNs in terms of the best accuracies on the test set. Although all architectures achieved accuracies close to 100%, VPNet achieved this with low structural complexity, which refers not only to the number of neurons, but also to the total number of network parameters (see Fig. 5(c)). In this regard, VPNet is superior, because with FCNNs and CNNs of the same effective receptive field, the number of parameters (i.e. the linear and kernel weights and biases) grows linearly with sample size and number of neurons. With VPNet, in contrast, the number of nonlinear parameters ($p = 2$) is independent of sample size and output dimension. For the sake of clarity, we remark that the kernel size or the number of convolutional layers in a CNN do not necessarily depend on the input size. Although, in order to detect global morphologic behavior of signals (e.g. heartbeats), the CNN is expected to have a large enough effective receptive field, that requires larger kernels or multiple layers stacked together in a linear scale. See also Ref. 94.

In addition to Fig. 5(c), the best test accuracies depending on the number of learnable parameters are given in Table 1. Here, the number of parameters is grouped into bins for easier interpretation. The results show that the VPNet outperforms the CNNs and FCNNs for each bin, and reaches peak performance earlier than the other two. Besides the numerical comparison, statistical hypothesis testing were also performed for each bin, if applicable. The differences between the best performing VPNets and CNNs are statistically significant by both paired-sample t-tests and McNemar's tests with significance level 5%.

Table 1. Evaluation on synthetic data: best test accuracies versus number of parameters.

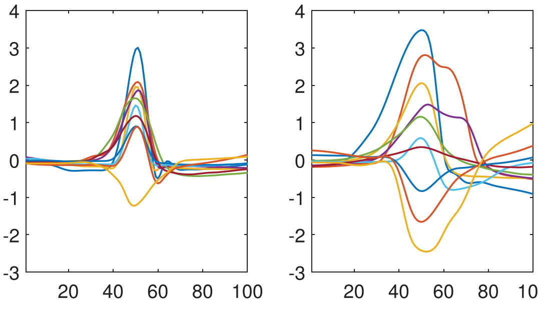
#	VPNet	CNN	FCNN
30–39	85.86%		
40–49	99.41%		
50–59	99.57%		
60–69	99.64%	71.65%	
70–79	99.87%	84.32%	
80–89	99.85%	93.33%	
90–99	99.94%	97.71%	
100–119	99.97%	98.86%	
120–139	99.98%	99.41%	81.14%
140–159	99.97%	99.77%	
160–179	99.96%	99.92%	97.01%
180–199		99.90%	
200–239		99.85%	98.34%
240–279		99.89%	99.47%
280–319		99.86%	99.65%
320–359			99.68%
360–399			99.77%
400–479			99.67%
480			99.91%

4.4. *Real ECG data*

We sought to prove the relevance of VPNet not only in simulation, but also using real signal-processing data. We chose a particular ECG signal-processing problem: classification of heartbeat arrhythmia (see Ref. 95). The state of the art is supervised ML by traditional approaches (see Refs. 19, 96, 97 and Sec. 2), including VP-based static feature extraction.[56,63,65] Here, we focused on a related subproblem, where we could compare the performance of the selected network configurations.

In detail, we investigated the separation of the two largest arrhythmia classes: normal and ventricular ectopic beats (VEBs). The source of the data is the benchmark MIT-BIH Arrhythmia Database,[98] available from PhysioNet.[99] The database is split into sets DS1 and DS2 according to Ref. 100, for training and inference, respectively. The whole database contains more than 100,000 annotated heartbeats, but it is heavily biased towards the normal class, that usually distorts the performance evaluation. Here, we investigated two cases for data acquisition. First, a balanced subset was extracted: all VEBs and the same number of normal beats from each record. This yielded 4260 plus 4260 heartbeat signals for training (set DS1), and 3220 plus 3220 signals for testing (set DS2). This balanced subset is expected to provide undistorted evaluation and fair comparison of the NN architectures. The second, unbalanced subset consists of all normal beats and VEBs of the whole database, yielding around 50,000 heartbeats for both training and testing. This unbalanced subset represents a more realistic scenario, and supports partial comparability to the state-of-the-art. Note that the DS1 and DS2 heartbeats come from different patients, which means that there is no data leakage in either cases. We used the preprocessing and heartbeat extraction methods discussed in Ref. 63, but chose a window size of 100 samples (~0.28 s) around the R peak annotations. This window was expected to cover the whole QRS complex and potentially the PR and ST segments of each heartbeat. Example heartbeats of the two classes are displayed in Fig. 6.

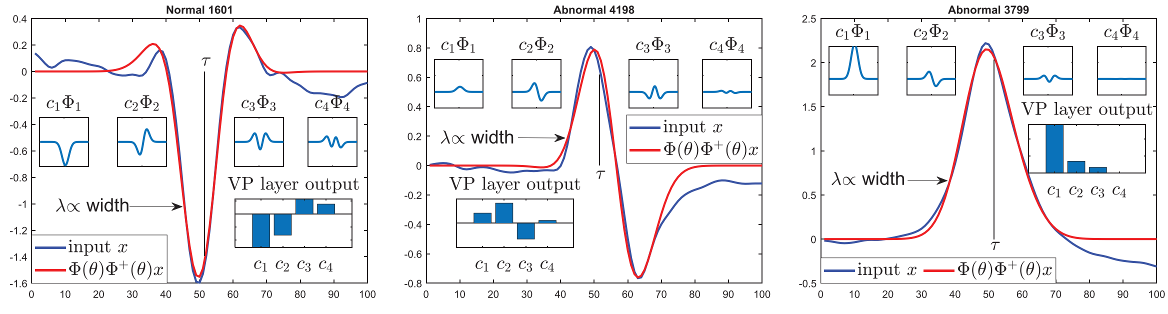
To demonstrate the interpretability of the results, we depicted the response of a trained VP layer to three input QRS complexes in Fig. 7. It can be seen that the Hermite-VP layer learned in fact the position τ and the width λ of the QRS complexes such that it gives an approximation (red) to the meaningful part of the original (blue) curves. In addition to the QRS complex, the input data window may include irrelevant information, such as baseline wander, noise, part of the P and the T waves. However, these irrelevant information are discarded due to the optimization of τ and λ, and thus only the meaningful part of the input signal is approximated at the end of the training. Consequently, the VP layer is likely to be more tolerant to noise as well. In fact, the Hermite-VP representation of ECG recordings can simultaneously cope with various noise sources such as baseline wander, and power-line interference.[92] The layer can also retain diagnostically important morphological information via the extracted coefficients. In Fig. 7, the red curve is equal to the linear combination of the Hermite functions, whose coefficients are the output of the VP layer. The magnitude of these coefficients indicate the presence of each elementary components in the signal. For instance, Fig. 7(b) shows an asymmetric QRS complex, which is reflected in a high coefficient c_2 that corresponds to an odd Hermite function. In contrast, Fig. 7(c) plots a highly symmetric QRS complex, which resembles to a Gaussian function indicated by the high value of c_1. Therefore, both the parameters τ, λ and the output c_i's of the VP layer are interpretable. Note that the level of interpretability tends to decrease as we connect more and more hidden layers to the network. The reason behind is that the whole network does not seek to reconstruct the parameters with which the data where constructed but it rather searches for the parameters that maximize the distinctness of the classes. Since the term presented in Sec. 3.4 penalizes the model for not reconstructing the original signal, a larger value for α mitigates the decreased interpretability. However, the VP layer provides a fully transparent feature extractor, which directly influences the output of the network due to the least-squares penalty in the modified loss function J_{VP}. Therefore, a trained VP layer can be used to improve the generalization properties of DNNs



(a) Normal.

(b) VEB.

Fig. 6. Example heartbeats of the training set.

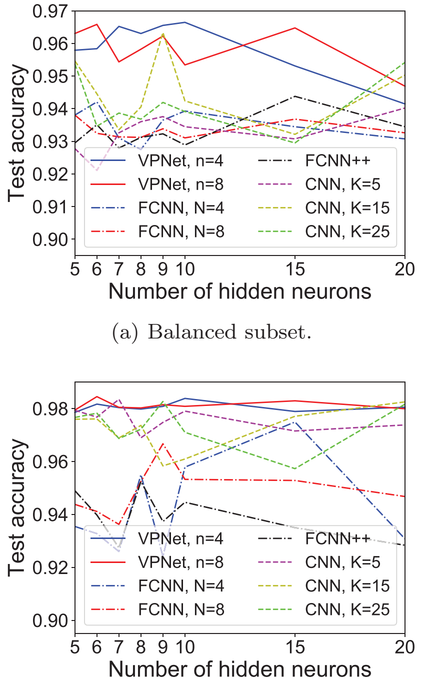


(a) Normal beat.

(b) Abnormal beat.

(c) Abnormal beat.

Fig. 7. Output of a trained VP layer: for a normal beat (a) and two abnormal beats (b), (c).



(a) Balanced subset.

(b) Unbalanced subset.

Fig. 8. Evaluation on real data, best test accuracies.

by synthesizing more realistic data samples in the learned feature space.[101,102]

The performance of VPNet was measured in a similar way as in the synthetic case, with more than 3500 possible hyperparameter configurations examined. The aggregated results are presented in Figs. 8(a) and 8(b), for the balanced and unbalanced case, respectively. Here, the FCNN and CNN cases were restricted so that the output dimensions of the first layer were similar to the VP dimensions, and only the number of neurons in the hidden layer were varied. Note that VPNet again required a remarkably low number of network parameters. We also evaluated another, larger FCNN configuration (FCNN++), where the number of neurons in the first layer was not restricted to that of the VP dimension n, but had the same number as in the second, hidden layer. The structure and distribution of training and test data were more complex than in the synthetic case, which clearly made the classification task more difficult for all network architectures. Again, we conclude that VPNet can outperform FCNNs and CNNs for low-complexity networks. Note that VPNet reaches peak performance at low network complexity (at low number of hidden neurons, i.e. at low number of system parameters), and the performance starts to decrease early if we increase the complexity. This behavior is slightly different for CNNs and FCNNs. A possible reason behind is that the first layer of the VPNet acts as a model-based feature extraction, i.e. provides a low-dimensional sparse representation of the input (4 or 8 features for 100 samples). Increasing the complexity of the fully-connected layers of VPNet without increasing the VP parameters or features will lead to over-parametrization and overfitting.

In addition to the total accuracies, the usual performance metrics are also provided in Table 2. Namely, sensitivity/precision (Se) and positive predictivity/recall ($+P$) was evaluated for each classes, as

$$\mathrm{Se} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \quad \text{and} \quad +\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \qquad (8)$$

where TP, FP and FN are the true positive, false positive, and false negative matches, respectively.

Table 2. Performance evaluation on real data.

Case/method		Total accuracy	Normal		VEB	
			Se	+P	Se	+P
Balanced	VPNet	96.65%	99.38%	94.23%	93.91%	99.34%
	FCNN	94.38%	93.79%	94.91%	94.97%	93.86%
	CNN	96.34%	97.76%	95.05%	94.91%	97.70%
Unbalanced	VPNet	98.45%	99.57%	98.78%	83.07%	93.37%
	FCNN	97.49%	98.50%	98.81%	83.70%	80.21%
	CNN	98.35%	99.39%	98.85%	84.07%	90.93%
State-of-the-art[19]		N/A	80–99%	85–99%	77–96%	63–99%

Reference intervals of the state-of-the-art are also given according to the survey Ref. 19. We note that the direct comparison is not always possible, since most of these results refer to a 3 or 5 class classification of the database.

5. Conclusion

We developed a novel model-driven NN which incorporates expert knowledge via variable projections. The VP layer is a generic, learnable feature extractor or filtering method that can be adjusted to several 1D signal-processing problems by choosing an application-specific function system. The proposed architecture is simple, which means it has only a few, interpretable parameters. Our case studies showed that VPNet can achieve similar or slightly better classification accuracy than its fully connected and CNN counterparts while using a smaller number of parameters. In our tests, the convergence of the VPNet was slightly better than that of the CNN and the FCNN counterparts. However, the VP layer required only two parameters for learning in all cases, whereas the number of weights and biases for the FCNN and CNN grew linearly with the length of the input signals. These results show that VPNet can be applied effectively to various problems in 1D signal processing including classification, regression, and clustering, which we will investigate as part of future work.

Broader Impact

We have proposed a new compact and interpretable neural network architecture that can have a broader impact in mainly two fields: machine learning and signal processing. The key idea is to create a network that combines the representation abilities of variable projections and the prediction abilities of NNs in the form of a composite model. This concept can be generalized to other machine learning algorithms. For instance, VP-SVM, and other combined VP methods, such as VP-K-means and VP-C-means, can extend the potential areas of application, including classification, regression, and clustering problems. Since the nonlinear parameters of the VP layer are interpretable, they can also be used in feature-space augmentation, where new data is generated from existing one in order to improve the generalization properties of DNNs.[101,102]

Signal-processing aspects of VPNet were discussed in the ECG heartbeat classification case study. Additionally, VPNet may have great potential in a wide range of applications especially where VP has proven to be an efficient estimation method (cf. Sec. 4.2). Note that many already existing adaptive signal models have been reformulated as VP problems[52,53]; however, parameterized wavelets[103] have not yet been studied in this context. Therefore, we encourage researchers to study this class of wavelets in the framework of VPNet.

Model-driven neural network solutions can have a great impact in biomedical engineering and healthcare informatics, where medical data classification alone is usually not enough, as physiological interpretation and explainability of the results are also important. However, special care should be taken to avoid automation bias when these approaches are applied to real-world problems.[104] These clinical decision-support systems are difficult to validate, since this requires medical expertise and vast amounts of data. The latter is naturally unbalanced in the sense that one class of signals (e.g. from healthy patients), is overrepresented compared to the others. In order to address these potential biases, VPNet should be tested in various scenarios that include, for instance, noisy and incomplete measurements, or unbalanced data.

Acknowledgments

This work has been supported by the University SAL Labs initiative of Silicon Austria Labs (SAL) and its Austrian partner universities for applied fundamental research for electronic based systems; and 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; and EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies. The Project is supported by the Hungarian Government and co-financed by the European Social Fund. This paper was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

The data and code that support the findings of this study are available at Ref. 105.

Appendix A

Let us consider the Hilbert space $L^2(\mathbb{R})$ endowed with the usual scalar product and norm:

$$\langle f, g\rangle = \int_{-\infty}^{\infty} f(t)g(t)\mathrm{d}t, \quad \|f\|_2 = \sqrt{\langle f, f\rangle},$$

where $f, g \in L^2(\mathbb{R})$. It is well-known that the Hermite functions Φ_k ($k \in \mathbb{N}$) in Eq. (7) are pairwise orthogonal, i.e. $\delta_{kj} = \langle \Phi_k, \Phi_j\rangle$, where δ_{kj} stands for the Kronecker delta symbol.

Another useful property of the Hermite functions Φ_k is that they converge rapidly to zero as $t \longrightarrow \pm\infty$. Therefore, in practice, we can assume that each Φ_k has a compact support. This can be used to satisfy the approximate orthogonality relation:

$$\delta_{kj} = \langle \Phi_k, \Phi_j\rangle \approx \int_a^b \Phi_k(t)\Phi_j(t)\mathrm{d}t,$$

provided that the supports of both Φ_k and Φ_j are embedded in a finite interval $[a; b]$. Note that the first Hermite function $\Phi_0(t)$ is equal, up to a constant factor, to the probability density function of the standard normal distribution $\mathcal{N}(0, 1)$. Therefore, the three-sigma rule applies, which means that around 68%, 95%, 99% of the overall integral of $\Phi_0(t)$ lies within the intervals $[-\ell; \ell]$ for $\ell = 1, 2, 3$, respectively. Therefore, in the case of $k = 0$, we choose the sampling interval such that $[-3; 3] \subseteq [a; b]$ holds. For larger indices $k > 0$, a heuristic empirical relation $\mathrm{supp}(\Phi_k) \sim 1.05^k \cdot [-3; 3] \subseteq [a; b]$ can be applied.

In practice, we typically use only the first few Hermite functions to model compactly supported waveforms (see Sec. 4), therefore the condition $[-3; 3] \subseteq [a; b]$ is sufficient. The same reasoning applies to the scalar product of the adaptive Hermite functions:

$$\langle \Phi_k(\cdot; \tau, \lambda), \Phi_j(\cdot; \tau, \lambda)\rangle \approx \int_{\lambda(a-\tau)}^{\lambda(b-\tau)} \Phi_k(s)\Phi_j(s)\mathrm{d}s,$$

where we simplified the integral on the right-hand side by substitution $s = \lambda(t - \tau)$. In order to satisfy the approximate orthogonality relation, the condition $[-3; 3] \subseteq [\lambda(a-\tau); \lambda(b-\tau)]$ must hold, i.e.:

$$-3 \geq \lambda(a - \tau), \quad \text{and} \quad \lambda(b - \tau) \geq 3,$$

which implies the feasible set Γ in Sec. 3.3.1.

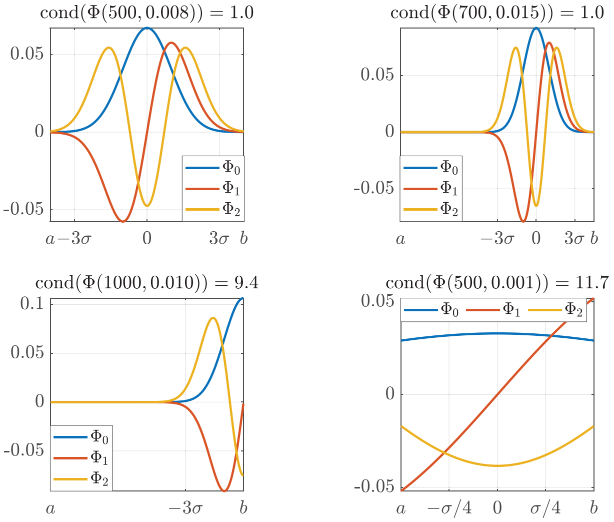


Fig. A.1. Ideal and extreme cases for $\mathrm{cond}(\mathbf{\Phi}(\tau, \lambda))$.

In Fig. A.1, we show four realizations of the first three adaptive Hermite functions sampled at $m = 1000$ number of equidistant points. The top figures demonstrate ideal cases when $(\tau, \lambda) \in \Gamma$, whereas the bottom figures show extreme examples with too large translation τ and too small dilation λ.

References

1. J. Schmidhuber, Deep learning in neural networks: An overview, *Neural Netw.* **61** (2015) 85–117.
2. E. Alpaydin, *Introduction to Machine Learning*, 4th edn. (The MIT Press, 2020), p. 640.
3. Y. Bengio, A. Courville and P. Vincent, Representation learning: A review and new perspectives, *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8) (2013) 1798–1828.
4. T. Yang, C. Cappelle, Y. Ruichek and M. El Bagdouri, Multi-object tracking with discriminant correlation filter based deep learning tracker, *Integr. Comput.-Aided Eng.* **26**(3) (2019) 273–284.
5. D. Daz-Vico, J. Prada, A. Omari and J. Dorronsoro, Deep support vector neural networks, *Integr. Comput.-Aided Eng.* **27**(4) (2020) 389–402.
6. O. Reyes and S. Ventura, Performing multi-target regression via a parameter sharing-based deep network, *Int. J. Neural Syst.* **29**(9) (2019) 1–22.

7. P. Mishra, C. Piciarelli and G. L. Foresti, A neural network for image anomaly detection with deep pyramidal representations and dynamic routing, *Int. J. Neural Syst.* **30**(10) (2020) 1–14.
8. R. B. Girshick, Fast R-CNN, preprint (2015), arXiv:abs/1504.08083.
9. S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, preprint (2015), arXiv:1506.01497.
10. G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science* **313**(5786) (2006) 504–507.
11. A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems*, eds. F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, Vol. 25 (Curran Associates, 2012), pp. 1097–1105.
12. M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, in *Computer Vision — ECCV 2014*, eds. D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars (Springer International Publishing, Cham, 2014), pp. 818–833.
13. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, Going deeper with convolutions, in *2015 IEEE Conf. Computer Vision and Pattern Recognition* (*CVPR*) (IEEE, 2015), pp. 1–9.
14. O. Ronneberger, P. Fischer and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *Int. Conf. Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241.
15. O. M. Manzanera, S. K. Meles, K. L. Leenders, R. J. Renken, M. Pagani, D. Arnaldi, F. Nobili, J. Obeso, M. R. Oroz, S. Morbelli and N. M. Maurits, Scaled subprofile modeling and convolutional neural networks for the identification of parkinson disease in 3d nuclear imaging data, *Int. J. Neural Syst.* **28** (2019) 1–15.
16. M. Leming, J. M. Górriz and J. Suckling, Ensemble deep learning on large, mixed-site fmri datasets in autism and other tasks, *Int. J. Neural Syst.* **30**(7) (2020) 1–16.
17. H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li and Z. Xu, Model-driven deep learning for physical layer communications, *IEEE Wireless Commun.* **26**(5) (2019) 77–83.
18. S. Kiranyaz, T. Ince and M. Gabbouj, Real-time patient-specific ECG classification by 1-d convolutional neural networks, *IEEE Trans. Biomed. Eng.* **63**(3) (2016) 664–675.
19. E. J. S. Luz, W. R. Schwartz, G. Cámara-Cháveza and D. Menotti, ECG-based heartbeat classification for arrhythmia detection: A survey, *Comput. Methods Program. Biomed.* **127** (2016) 144–164.
20. S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj and D. J. Inman, 1D convolutional neural networks and applications: A survey, arXiv: 1905.03554.
21. R. T. Q. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, Neural ordinary differential equations, *32nd Conf. Neural Information Processing Systems* (*NeurIPS 2018*) (NIPS, 2018), pp. 6572–6583.
22. L. Ruthotto and E. Haber, Deep neural networks motivated by partial differential equations, arXiv: 1804.04272.
23. J. R. Hershey, J. L. Roux and F. Weninger, Deep unfolding: Model-based inspiration of novel deep architectures, arXiv:1409.2574.
24. N. Samuel, T. Diskin and A. Wiesel, Learning to detect, *IEEE Trans. Signal Process.* **67**(10) (2019) 2554–2564.
25. H. Yu, J. Peng and Y. Tang, Identification of nonlinear dynamic systems using Hammerstein-type neural network, *Math. Problem. Eng.* **2014** (2014) 1–9.
26. Y. Chen and T. Pock, Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration, *IEEE Trans. Pattern Anal. Machine Intell.* **39**(6) (2018) 1256–1272.
27. K. Kunisch and T. Pock, A bilevel optimization approach for parameter learning in variational models, *SIAM J. Imag. Sci.* **6**(2) (2013) 938–983.
28. R. Liu, Z. Lin, W. Zhang and Z. Su, Learning PDEs for image restoration via optimal control, in *Computer Vision — ECCV 2010* (Springer, Berlin, 2010), pp. 115–128.
29. B. Amos and J. Z. Kolter, Optnet: Differentiable optimization as a layer in neural networks, *34th Int. Conf. Machine Learn.* (*ICML 2017*) (PMLR, 2017), pp. 136–145.
30. E. Weinan, A proposal on machine learning via dynamical systems, *Commun. Math. Statistic.* **5** (2017) 1–11.
31. M. Ławryńczuk, Computationally efficient nonlinear predictive control based on neural Wiener models, *Neurocomputing* **74**(1–3) (2010) 401–417.
32. M. Ławryńczuk, Practical nonlinear predictive control algorithms for neural Wiener models, *J. Process Control* **23**(5) (2013) 696–714.
33. B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham, Reversible architectures for arbitrarily deep residual neural networks, *The 32nd AAAI Conf. Artific. Intell.* (*AAAI-18*) (AAAI, 2018), pp. 2811–2818.
34. M. Borgerding, P. Schniter and S. Rangan, AMP-inspired deep networks for sparse linear inverse problems, *IEEE Trans. Signal Process.* **65**(16) (2017) 4293–4308.
35. D. Ito, S. Takabe and T. Wadayama, Trainable ISTA for sparse signal recovery, *IEEE Trans. Signal Process.* **67**(12) (2019) 3113–3125.

36. A. Balatsoukas-Stimming and C. Studer, Deep unfolding for communications systems: A survey and some new directions, *2019 IEEE Int. Workshop Signal Processing System* (*SiPS*) (IEEE, 2019), pp. 266–271.
37. A. T. Kristensen, A. Burg and A. Balatsoukas-Stimming, Identification of nonlinear RF systems using backpropagation, arXiv:2001.09877.
38. U. Schmidt and S. Roth, Shrinkage fields for effective image restoration, *IEEE Conf. Comput. Vision Pattern Recognition* (IEEE, 2014), pp. 2768–2781.
39. K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock and F. Knoll, Learning a variational network for reconstruction of accelerated mri data, *Magnet. Resonance Med.* **79**(6) (2018) 3055–3071.
40. 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**(10) (2020) 6393–6404.
41. P. Lara-Bentez, M. Carranza-Garca, J. Garca-Gutirrez and J. C. Riquelme, Asynchronous dual-pipeline deep learning framework for online data stream classification, *Integr. Comput.-Aided Eng.* **27**(2) (2020) 101–119.
42. P. Peng, L. Xie and H. Wei, A deep fourier neural network for seizure prediction using convolutional neural network and ratios of spectral power, *Int. J. Neural Syst.* **31** (2021) 2150022.
43. K. Alam, N. Siddique and H. Adeli, A dynamic ensemble learning algorithm for neural networks, *Neural Comput. Appl.* **32**(10) (2020) 8675–8690.
44. M. Ahmadlou and H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integr. Comput.-Aided Eng.* **17**(3) (2010) 197–210.
45. R. Sánchez-Reolid, A. Martínez-Rodrigo, M. T. López and A. Fernández-Caballero, Deep support vector machines for the identification of stress condition from electrodermal activity, *Int. J. Neural Syst.* **30**(7) (2020) 1–16.
46. L. Pan, G. Păun, G. Zhang and F. Neri, Spiking neural P systems with communication on request, *International Journal of Neural Systems* **28** (2017) 1–13.
47. T. Wu, F.-D. Blbe, A. Păun, L. Pan and F. Neri, Simplified and yet Turing universal spiking neural P systems with communication on request, *Int. J. Neural Syst.* **28** (2018) 1–19.
48. P. Lara-Bentez, M. Carranza-Garca and J. C. Riquelme, An experimental review on deep learning architectures for time series forecasting, *Int. J. Neural Syst.* **31**(3) (2021) 1–26.
49. X. Song, L. Valencia-Cabrera, H. Peng, J. Wang and M. J. Prez-Jimnez, Spiking neural P systems with delay on synapses, *Int. J. Neural Syst.* **31**(1) (2021) 1–19.
50. G. H. Golub and V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, *SIAM J. Num. Anal.* (*SINUM*) **10** (1973) 413–432.
51. D. P. O'Leary and B. W. Rust, Variable projection for nonlinear least squares problems, *Comput. Opt. Appl.* **54**(3) (2013) 579–593.
52. G. H. Golub and V. Pereyra, Separable nonlinear least squares: The variable projection method and its applications, *Inverse Problem.* **19**(2) (2003) R1–R26.
53. G.-Y. Chen, M. Gan, S. Wang and C. L. P. Chen, Insights into algorithms for separable nonlinear least squares problems, *IEEE Trans. Image Process.* **30**(10) (2020) 1207–1218.
54. V. Pereyra, G. Scherer and F. Wong, Variable projections neural network training, *Math. Comput. Simul.* **73**(1) (2006) 231–243.
55. P. Kovács, S. Fridli and F. Schipp, Generalized rational variable projection with application in ECG compression, *IEEE Trans. Signal Process.* **68**(16) (2020) 478–492.
56. T. Dózsa, G. Bognár and P. Kovács, Ensemble learning for heartbeat classification using adaptive orthogonal transformations, *Computer Aided Systems Theory–EUROCAST 2019*: *Part II*, Lecture Notes in Computer Science, eds. R. Moreno-Díaz *et al.*, Vol. 12014 (Springer, Cham, 2020), pp. 355–363.
57. P. Kovács, C. Böck, J. Meier and M. Huemer, ECG segmentation using adaptive Hermite functions, in *Proc. 51st Annual Asilomar Conf. Signals*, *Systems*, *and Computers* (IEEE, 2017), pp. 1476–1480.
58. P. Kovács, C. Böck, T. Dózsa, J. Meier and M. Huemer, Waveform modeling by adaptive weighted Hermite functions, in *Proc. 44th IEEE Int. Conf. Acoustics*, *Speech and Signal Processing* (*ICASSP*) (IEEE, 2019), pp. 1080–1084.
59. S. Fridli, L. Lócsi and F. Schipp, Rational function system in ECG processing, *Computer Aided Systems Theory–EUROCAST 2011*: *Part I*, Lecture Notes in Computer Science, eds. R. Moreno-Díaz *et al.*, Vol. 6927 (Springer-Verlag, Berlin, 2012), pp. 88–95.
60. S. Fridli, P. Kovács, L. Lócsi and F. Schipp, Rational modeling of multi-lead QRS complexes in ECG signals, *Ann. Univ. Sci. Budapest. Sect. Comp.* **37** (2012) 145–155.
61. G. Bognár, S. Fridli, P. Kovács and F. Schipp, Adaptive rational transformations in biomedical signal processing, in *Progress in Industrial Mathematics at ECMI 2018*, eds. S. Péter *et al.* (Springer, Cham, 2019), pp. 239–247.
62. K. Samiee, P. Kovács and M. Gabbouj, Epileptic seizure classification of EEG time-series using rational discrete short time Fourier transform, *IEEE*

Transactions on Biomedical Engineering **62**(2) (2014) 541–552.

63. G. Bognár and S. Fridli, Heartbeat classification of ECG signals using rational function systems, in *Computer Aided Systems Theory–EUROCAST 2017*: *Part II*, Lecture Notes in Computer Science, eds. R. Moreno-Díaz *et al.*, Vol. 10672 (Springer, Cham, 2018), pp. 187–195.
64. G. Bognár and S. Fridli, ECG segmentation by adaptive rational transform, *Computer Aided Systems Theory–EUROCAST 2019*: *Part II*, eds. R. Moreno-Díaz *et al.*, LNCS, Vol. 12014 (Springer, Cham, 2020), pp. 347–354.
65. G. Bognár and S. Fridli, ECG heartbeat classification by means of variable rational projection, *Biomed. Signal Process. Control* **61** (2020) 102034.
66. I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning* (MIT Press, 2016).
67. A. T. Tzallas, M. G. Tsipouras and D. I. Fotiadis, Epileptic seizure detection in EEGs using time-frequency analysis, *IEEE Trans. Inf. Technol. Biomed.* **13**(5) (2009) 703–710.
68. G. Liu, W. Zhou and M. Geng, Automatic seizure detection based on s-transform and deep convolutional neural network, *Int. J. Neural Syst.* **30** (2020) 1–15.
69. L.-C. Lin, C.-S. Ouyang, R.-C. Wu, R.-C. Yang and C.-T. Chiang, Alternative diagnosis of epilepsy in children without epileptiform discharges using deep convolutional neural networks, *Int. J. Neural Syst.* **30** (2020) 1–10.
70. M. Paluszny, M. Lentini, M. Martin-Landrove and Torres, Recovery of relaxation rates in MRI T2 weighted brain images via exponential fitting, *Exponential Data Fitting and its Applications*, eds. V. Pereyra and G. Scherer (Bentham Science, 2010), pp. 52–70.
71. L. N. Trefethen and III. B. Bau, *Numerical Linear Algebra* (SIAM, Philadelphia, USA, 1997).
72. G. Szegő, *Orthogonal Polynomials*, 3rd edn. (AMS Colloquium Publications, New York, 1967).
73. W. Gautschi, *Orthogonal Polynomials*, *Computation and Approximation* (Oxford University Press, Oxford, 2004).
74. W. Gautschi, Orthogonal polynomials (in Matlab), *J. Comput. Appl. Math.* **178**(10) (2005) 215–234.
75. D. E. Rumelhart and J. L. McClelland, Learning internal representations by error propagation, *Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition*: *Foundations*, eds. D. E. Rumelhart and J. L. McClelland (MIT Press, 1986), pp. 318–362.
76. D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.
77. J. Duchi, E. Hazan and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Machine Learn. Res.* **12**(61) (2011) 2121–2159.
78. T. Tieleman and G. Hinton, Lecture 6.5 — RmsProp: Divide the gradient by a running average of its recent magnitude COURSERA: Neural networks for machine learning **4**(2) (2012) 26–31.
79. S. L. Hung and H. Adeli, Parallel backpropagation learning algorithms on CRAY y-MP8/864 supercomputer, *Neurocomputing* **5**(6) (1993) 287–302.
80. S.-L. Hung and H. Adeli, Object-oriented backpropagation and its application to structural design, *Neurocomputing* **6**(1) (1994) 45–55.
81. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, Automatic differentiation in pytorch, *Neural Inf. Process. Syst.* **31** (2017) 1–4.
82. M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://tensorflow.org.
83. G. Leibon, D. N. Rockmore, W. Park, R. Taintor and G. S. Chirikjian, A fast Hermite transform, *Theoretical Comput. Sci.* **409**(2) (2008) 211–228.
84. E. Moya-Albor, B. Escalante-Ramírez and E. Vallejob, Optical flow estimation in cardiac CT images using the steered Hermite transform, *Signal Process. Image Commun.* **28**(3) (2013) 267–291.
85. S. Stanković, I. Orović and A. Krylov, The two-dimensional Hermite S-method for high resolution inverse synthetic aperture radar imaging applications, *IET Signal Process.* **4**(4) (2010) 352–362.
86. P. Lazaridis, G. Debarge and P. Gallion, Discrete orthogonal Gauss–Hermite transform for optical pulse propagation analysis, *J. Opt. Soc. Amer. B* **20**(7) (2003) 1508–1513.
87. L. R. Lo Conte, R. Merletti and G. V. Sandri, Hermite expansions of compact support waveforms: Applications to myoelectric signals, *IEEE Trans. Biomed. Eng.* **41**(12) (1994) 1147–1159.
88. 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** (2000) 717–838.
89. 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.
90. H. Haraldsson, L. Edenbrandt and M. Ohlsson, Detecting acute myocardial infarction in the 12-lead ECG using Hermite expansions and neural networks, *Artific. Intell. Med.* **32** (2004) 127–136.
91. M. Brajović, I. Orović, M. Daković and S. Stanković, On the parameterization of Hermite transform with application to the compression of QRS complexes, *Signal Process.* **131** (2017) 113–119.
92. C. Böck, P. Kovács, J. Meier and M. Huemer, Ecg beat representation and delineation by means

of variable projection, *IEEE Trans. Biomed. Eng.* (2021) 1–12.

93. S.-H. Wang, Y.-D. Zhang, M. Yang, B. Liu, J. Ramirez and J. M. Gorriz, Unilateral sensorineural hearing loss identification based on double-density dual-tree complex wavelet transform and multinomial logistic regression, *Integr. Comput.-Aided Eng.* **26**(4) (2019) 411–426.
94. W. Luo, Y. Li, R. Urtasun and R. Zemel, Understanding the effective receptive field in deep convolutional neural networks, arXiv:1701.04128.
95. 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).
96. R. J. Martis, U. R. Acharya and H. Adeli, Current methods in electrocardiogram characterization, *Comput. Biol. Med.* **48** (2014) 133–149.
97. 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.
98. G. B. Moody and R. G. Mark, The impact of the MIT-BIH Arrhythmia database, *IEEE Eng. Med. Biol. Mag.* **20**(3) (2001) 45–50.
99. A. L. Goldberger *et al.*, PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals, *Circulation* **101**(23) (2000) e215–e220.
100. 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**(7) (2004) 1196–1206.
101. Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. Breuel, Y. Chherawala, M. Cisse, M. Cté, D. Erhan, J. Eustache, X. Glorot, X. Muller, P. S. Lebeuf, R. Pascanu, S. Rifai, F. Savard and G. Sicard, Deep learners benefit more from out-of-distribution examples, in *Proc. 14th Int. Conf. Artificial Intelligence and Statistics* (JMLR Workshop and Conf. Proc., 2011), pp. 164–172.
102. T. DeVries and G. W. Taylor, Dataset augmentation in feature space, in *Proc. Int. Conf. Learning Representations* (*ICLR*) *Workshop* (ICLR, 2017), pp. 1–12.
103. C. S. Burrus, A. R. Gopinath and H. Guo, *Introduction to Wavelets and Wavelet Transforms*: *A Primer*, Chap. 5, 1st edn. (Prentice Hall, New Jersey, 1997).
104. K. Goddard, A. Roudsari and J. C. Wyatt, Automation bias: A systematic review of frequency, effect mediators, and mitigators, *J. Amer. Med. Inf. Assoc.* **19**(1) (2012) 121–127.
105. P. Kovács, G. Bognár, C. Huber and M. Huemer, VPNet — Variable Projection Network (2021), https://git.silicon-austria.com/pub/sparseestimation/vpnet.