END-TO-END TRAINABLE GAUSSIAN FILTERING FOR ELECTROCARDIOGRAM SIGNAL CLASSIFICATION USING DEEP LEARNING

Angelos Nalmpantis[1,2], Nikolaos Passalis[1], and Anastasios Tefas[1]
[1]*Computational Intelligence and Deep Learning Group, AIIA lab., Department of Informatics*,
Aristotle University of Thessaloniki, Thessaloniki, Greece
[2]University of Amsterdam, Amsterdam, The Netherlands
{*angelossn, passalis, tefas*}*@csd.auth.gr*

***Abstract*—Deep Learning (DL) is increasingly used in electrocardiograms (ECGs)-based signal analysis allowing for diagnosing a variety of cardiac disorders and enabling the development of automated diagnosis and prognosis systems. However, ECG signals are typically noisy requiring a series of prepossessing steps before feeding them into DL models for analysis. The main contribution of this paper is a simple, yet very effective end-to-end hybrid trainable filtering and feature extraction pipeline, that is formulated as a set of differentiable layers that can be incorporated into any DL model and employ Gaussian filters as a prior model. Indeed, it is experimentally demonstrated that the proposed method can increase the performance of ECG-based classification both for baseline architectures (e.g., MLPs, CNNs, LSTMs) as well as for state-of-the-art architectures proposed in the literature for ECG signal analysis. Given the simplicity and effectiveness of the proposed method, we believe that it constitutes a practical tool that can readily be incorporated into any DL pipeline that is used for time-series analysis, potentially further increasing its performance with minimal computational overhead.**

***Index Terms*—electrocardiogram signal classification, data preprocessing, deep learning, trainable signal denoising, end-to-end trainable filtering**

I. Introduction

Electrocardiograms (ECGs) are produced by placing electrodes on the skin and encode information about the heart. Electrocardiography is the main method to capture heart activity, since it is easy to apply and is painless for patients. Doctors frequently utilize them to diagnose a variety of cardiac disorders and, hence, it is essential to develop systems for automatic categorization that can drastically aid clinicians' diagnoses. The recent success of DL methods in tackling difficult problems [1], draw the attention of the corresponding communities and motivated the application of DL models in ECG signal analysis and classification, with recent research mainly focusing on developing supervised deep learning (DL) models [2], [3], [4], [5].

Despite the success of DL models in such applications, they are still susceptible to the inherent *noise* of ECGs [6], which is usually captured during the electrocardiography process. A variety of noise sources can affect such signals with the predominant ones stemming from baseline wander, powerline interference, and muscle artifacts [7], [8]. As the patients' welfare is dependent on it, it is vital that systems' miscalculations be prevented and noise removal techniques be developed. Typically, low-pass filters are frequently employed to mitigate the effects of signal noise and provide more appropriate input representations to models. However, the cut-off frequency of such filters is fixed during the training process, and, in most cases, it is either manually selected by practitioners or specified by heuristics.

The main hypothesis examined in this work is whether it is possible to construct end-to-end trainable pipelines for time-series filtering in order to allow DL models to appropriately adjust the way data are pre-processed in order to better accommodate the task at hand. This will allow for letting the models select - through backpropagation - the filtering parameters. Indeed, there are some DL approaches that attempt to tackle this issue, e.g., denoising autoencoders [9], denoising convolutional neural networks (CNNs) [10], etc. However, these methods do not employ any strict prior regarding the way filtering should be performed. As a result, such methods are still vulnerable to noise, since without any strong prior, they tend to overfit the data.

Based on these observations, in this work, we propose a simple yet effective end-to-end trainable neural data-driven pipeline that combines a Gaussian filtering prior as a model-based component, resulting in a hybrid approach that integrates both model-based and data-driven components [11]. Using Gaussian filters as a prior model for filtering restricts the expression ability of the proposed preprocessing layers. However, at the same time it mitigates overfitting issues that often arise with noisy signals, allowing for increasing the generalization abilities of the subsequent model. At the same time, the proposed method can be also used to generate different *views* of the data by using higher order filtering. As a result, the proposed method can not only provide a filtered version of the input signal but can also learn how

to extract higher order features that can be directly exploited by the subsequent model. Furthermore, the proposed method is easy to use and can be used with essentially any DL architecture. Indeed, we demonstrate that the proposed method can increase the performance of ECG-based classification both for baseline architectures (e.g., MLPs, CNNs, LSTMs) as well as for state-of-the-art architectures proposed in the literature. Given the simplicity and effectiveness of the proposed method, we believe that it constitutes a practical tool that can readily be incorporated into any DL pipeline that is used for time-series analysis, potentially further increasing its performance with minimal computational overhead. An easy to use implementation of the proposed method is available at https://github.com/cidl-auth/deep_gaussian to allow for easily using and extending the proposed method.

The rest of this paper is structured as follows. First, we introduce the proposed method in Section II. Then, the experimental evaluation is provided in Section III. Finally, conclusions are drawn in Section IV.

II. Proposed Method

Let $\mathbf{x}$ denote a one channel ECG signal, i.e., a one dimensional vector of size N. This is without loss of generality since the proposed method can similarly handle multivariate data. Also, let $f_{\mathbf{W}}(\mathbf{x})$ denote a DL model that is used to analyze this signal, e.g., to perform classification. The proposed Deep Gaussian Filtering (DGF) method receives $\mathbf{x}$ as input and generates a number of filtered versions of the input signal that are fed to subsequent DL model $f_{\mathbf{W}}(\cdot)$. In this way, the proposed method performs both denoising, as well as generates different *views* of the input signal that can be directly exploited by the subsequent model. The main building block of the DGF is a trainable Gaussian filter defined as:

$$g(h, \sigma, k) = \frac{1}{\sqrt{2\pi}\sigma} \frac{d^k K(h, \mu, \sigma)}{dh^k}, \tag{1}$$

where σ is a trainable parameter that controls the width of the input filter, k is the filter order and $K(\cdot)$ denotes the Gaussian kernel defined as:

$$K(h, \mu, \sigma) = e^{-\frac{(h-\mu)^2}{2\sigma^2}}. \tag{2}$$

Then, this filter is convoluted with the input signal $\mathbf{x}$ in order to obtain the filtered version of the input. Also, note that all Gaussian kernels used in this work are centered to 0, i.e., $\mu = 0$, since adjusting the centering can lead to instabilities when finite approximations of the filters are used.

It is easy to see that setting different values for the order k allows for obtaining different versions of the filter. For example, for $k = 0$ we obtain:

$$g(h, \sigma, 0) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{h^2}{2\sigma^2}} \tag{3}$$

for $k = 1$ we obtained:

$$g(h, \sigma, 1) = \frac{-h}{\sigma^3\sqrt{2\pi}} e^{-\frac{h^2}{2\sigma^2}} \tag{4}$$

while for $k = 2$ we get:

$$g(h, \sigma, 2) = \frac{h^2 - \sigma^2}{\sigma^5\sqrt{2\pi}} e^{-\frac{h^2}{2\sigma^2}}. \tag{5}$$

Note that each of these three filters uses the same trainable bandwidth factor. In order to understand why we might want to include higher order filters during the input prepossessing stage we need to consider the convolution's commutative and associative properties. Indeed, performing convolution with these filters is equivalent to calculating the corresponding derivatives of the input and then performing filtering. In this way, different properties of the input signal can be directly processed by the subsequent DL model without any further analysis and risking losing critical information by the filtering process. Therefore, the proposed method also provides an efficient way to obtain these representations at a very low cost, since most DL frameworks and hardware is already optimized for such calculations. For the rest of this section, we will assume the three filters defined by (3), (4) and (5) are used. Note that this is without loss of generality since any number of such filters can be used depending on the needs of each application. However, including additional higher order filters yields diminishing returns.

Note that the bandwidth parameter σ in (3), (4), (5) is a trainable parameter that can be updated with any gradient-based optimization method along with the rest of the model using backpropagation. In order to efficiently implement the proposed method we have constructed finite length kernels of size M, which were populated dynamically with values drawn from the corresponding Gaussian filters. Intuitively, the parameters σ determine how much denoising will be applied to the corresponding versions of the input. By incorporating it into the learning process, it can be adapted to the given data. The greater the σ value, the smoother the signal will be. If σ becomes large enough, then the Gaussian kernel will converge to the mean filter. For initializing the filters, a default value of $\sigma = 1$ can be used. Furthermore, the denominator of the first term in each function can be also omitted as it only provides a scaling factor that can be trivially incorporated into the learning process.

At this point, it is important to mention that the proposed method differs significantly from traditional convolutional layers. Instead of adjusting a kernel that is randomly initialized, the proposed method learns the parameters of specific filtering functions that are directly approximated by computing their corresponding discrete filters, imposing a very restrictive prior model to the filtering process and reducing the effect of overfitting. In this way, the kernels follow a formulated structure and are not freely adjusted. This is a critical point for the proposed method, since information that has been removed by early layers cannot be then recovered by subsequent layers [12]. Indeed, as we show later in the paper, convolutional layers alone cannot extract the same features that appear to be beneficial to the model's performance, due to the lack of an appropriate prior - especially when not enough data are available. On the other hand, combining the proposed

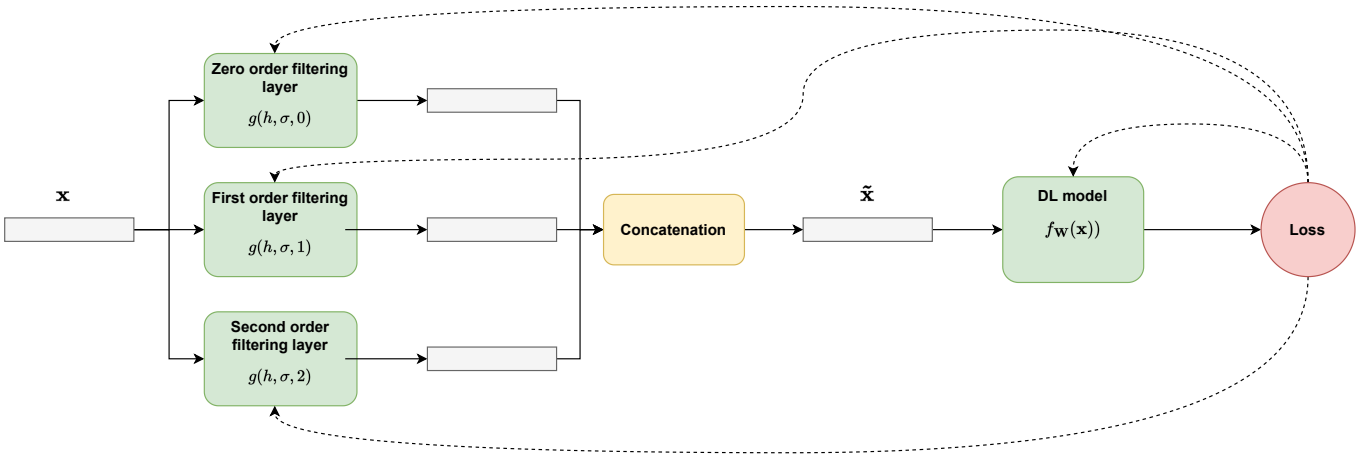

Fig. 1. Proposed end-to-end trainable pipeline of the proposed method. Dotted lines indicate flow of gradients, indicating backpropagating the gradients into the filtering layers.

method with convolutional layers provides significant benefits, indicating that the features learned using the proposed method are indeed more useful for the subsequent analysis task and that the employed prior is not too restrictive for the model's functions.

The complete pipeline of the proposed method is depicted in Fig. 1. The input signal $\mathbf{x}$ is filtered using the three trainable layers that employ the filters $g(h, \sigma, 0)$ to $g(h, \sigma, 2)$. Then the output of these filters is concatenated (i.e., the number of input channels is tripled) and fed to the subsequent DL model. The whole pipeline can be then trained with any gradient descent-based method in order to minimize a loss function $\mathcal{L}$.

III. Experimental Evaluation

The proposed method was evaluated using two datasets: a) the PTB Diagnostic ECG (PTBDB) dataset [13] and b) the MIT-BIH Arrhythmia (MITBIH) dataset [14]. For both datasets, we used the versions available at [15]. For all the conducted experiments we used the Adam optimizer [16] using a learning rate of 10^{-3} and the default hyper-parameters. The total number of epochs and batch size was set to 64 and 20 for the PTBDB dataset, while for the MITBIH we ran the optimization for 10 epochs using larger batches of 256 samples. These values were used for all the conducted experiments regardless of the employed architecture. For each experiment, we repeated the training process five times (using different initializations) and we report the average results. For selecting the best model during the evaluation we split the training set into 80% training and 20% validation data. All models were trained for classification using the negative loss likelihood loss. For the proposed method, we set the size of the filter used for the conducted experiment to $M = 11$.

First, we performed a set of evaluations in order to examine the performance of the proposed method using three baseline neural architectures: 1) a multilayer perceptron (MLP) with 3 hidden layers (each of which contained 16 neurons that employ the PReLu activation[17]), b) a convolutional neural network (CNN) consisting of one convolutional layer with 16 filters of size 3 and 3) a long-short-term-memory (LSTM) with 1 layer and hidden size 16. For the CNN and LSTM models, we used an MLP with the same architecture (except for the input size) for performing classification using the representations extracted from the CNN and LSTM. The experimental results, where we report the accuracy, F1 score, precision, and recall, are reported in Table I. In all the evaluated cases using the proposed method significantly increases the evaluation metrics. For example, F1 increases by 2% (more than 3.5% for the MITBIH dataset) when the proposed method is combined with an LSTM, while similar improvements are also observed for the MLP and CNN models.

TABLE I
Evaluation of the proposed method with three baseline architectures. The average of each metric of 5 different runs is reported.

Model	Accuracy	F1	Precision	Recall
PTBDB dataset				
MLP	0.9285	0.9096	0.9118	0.9117
Proposed + MLP	**0.9367**	**0.9190**	**0.9266**	**0.9154**
CNN	0.9412	0.9269	0.9239	0.9332
Proposed + CNN	**0.9542**	**0.9425**	**0.9424**	**0.9452**
LSTM	0.9553	0.9429	0.9505	0.9378
Proposed + LSTM	**0.9749**	**0.9683**	**0.9694**	**0.9683**
MITBIH dataset				
MLP	0.9469	0.7178 0.6214	0.8496	
Proposed + MLP	**0.9637**	**0.8202**	**0.7639**	**0.8854**
CNN	0.9603	0.8096	0.7378	**0.8969**
Proposed + CNN	**0.9671**	**0.8392**	**0.7938**	0.8901
LSTM	0.9658	0.8221	0.7518	0.9070
Proposed + LSTM	**0.9741**	**0.8603**	**0.8123**	**0.9142**

Next, we evaluated the performance of the proposed method using four state-of-the-art methodologies reported in the literature [2], [4], [5], [18]. The model proposed in [4] can be used with a combination of convolutional and bidirectional LSTM layers by either feeding the convolutional part to

TABLE II
EVALUATION OF THE PROPOSED METHOD WITH STATE-OF-THE-ART ARCHITECTURES, AS THEY ARE PROPOSED IN THE CORRESPONDING LITERATURE. THE AVERAGE OF EACH METRIC OF 5 DIFFERENT RUNS IS REPORTED.

Model	Accuracy	F1	Precision	Recall
PTBDB dataset				
[18]	0.9855	0.9815	0.9844	0.9795
Proposed + [18]	**0.9873**	**0.9840**	**0.9861**	**0.9826**
[2]	0.9759	0.9696	0.9699	0.9705
Proposed + [2]	**0.9920**	**0.9901**	**0.9921**	**0.9886**
[4](feed)	0.9940	0.9926	0.9927	0.9929
Proposed + [4](feed)	**0.9958**	**0.9948**	**0.9947**	**0.9950**
[4](concat.)	0.9918	0.9898	0.9901	0.9900
Proposed + [4](concat.)	**0.9943**	**0.9927**	**0.9933**	**0.9925**
[5]	0.9807	0.9754	0.9722	0.9799
Proposed + [5]	**0.9872**	**0.9838**	**0.9837**	**0.9846**
MITBIH dataset				
[18]	0.9787	0.0834	0.8496	0.9199
Proposed + [18]	**0.9799**	**0.8945**	**0.8679**	**0.9228**
[2]	0.9751	0.8640	0.8235	0.9086
Proposed + [2]	**0.9826**	**0.9084**	**0.8768**	**0.9424**
[4](feed)	0.9830	0.9076	0.8815	**0.9354**
Proposed + [4](feed)	**0.9847**	**0.9143**	**0.8964**	0.9330
[4](concat)	0.9836	0.9102	0.8834	0.9388
Proposed + [4](concat)	**0.9840**	**0.9139**	**0.8891**	**0.9401**
[5]	0.9769	0.8826	0.8536	0.9136
Proposed + [5]	**0.9786**	**0.8913**	**0.8547**	**0.9313**

the LSTM (abbreviated as "feed.") or concatenating their outputs (abbreviated as "concat."). Therefore, we evaluated both options. Table II summarizes the evaluation for these methodologies. Again, using the proposed method led to significant improvements in all the evaluated cases, demonstrating that appropriate data pre-processing can positively impact both the baseline models, as well as the most sophisticated ones.

Finally, we have performed an ablation study to quantify the impact of learning the parameters of the Gaussian kernels and validate that the positive impact on the accuracy does not arise only from the filtering process. The experimental results are reported in Table III, where we compare the results by using three Gaussian filters (as in the proposed method) but fixing the bandwidth to 1 and learning the bandwidth using backpropagation. Even though positive results are reported for the filtering process, possibly also due to the feature extraction process that allows for capturing different aspects of the data, in all cases the proposed method leads to the best evaluation results, confirming the importance of learning the parameters of the filters.

IV. CONCLUSIONS

In this paper, we proposed a simple, yet very effective end-to-end trainable filtering and feature extraction pipeline, that is formulated as a series of differentiable layers that can be incorporated into any DL model and employ Gaussian filters as prior. As it was experimentally demonstrated, the proposed method can increase the performance of ECG-based classification both for baseline architectures (e.g., MLPs, CNNs, LSTMs) as well as for state-of-the-art architectures

TABLE III
ABLATION STUDY EVALUATING THE EFFECT OF USING FIXED KERNELS (BANDWIDTH SET TO 1) AND TRAINING THE PROPOSED LAYERS IN AN END-TO-END FASHION. RESULTS ARE REPORTED IN THE PTBDB DATASET.

Model	Accuracy	F1	Precision	Recall
Proposed (no train) + MLP	0.9263	0.9058	0.9161	0.9008
Proposed + MLP	**0.9367**	**0.9190**	**0.9266**	**0.9154**
Proposed (no train) + CNN	0.9457	0.9318	0.9285	0.9382
Proposed + CNN	**0.9542**	**0.9425**	**0.9424**	**0.9452**
Proposed (no train) + LSTM	0.9701	0.9627	0.9576	0.9698
Proposed + LSTM	**0.9749**	**0.9683**	**0.9694**	**0.9683**

proposed in the literature. Furthermore, the conducted ablation studies validated the importance of learning the parameters of the models in an end-to-end fashion, instead of using fixed filters. The effectiveness of the proposed method paves the way for applications in time-series applications, where noisy data typically exist, such as financial time-series analysis [19]. Finally, the proposed method can be potentially combined with end-to-end trainable normalization schemes, such as DAIN [20] and Global Adaptive Normalization [21], to lead to powerful normalization and filtering pipelines that can further adapt to the input data and improve the accuracy of DL models

V. ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Xuexiang Xu and Hongxing Liu, "ECG heartbeat classification using convolutional neural networks," *IEEE Access*, vol. 8, pp. 8614–8619, 2020.

[3] Sebastian Goodfellow, Andrew Goodwin, Danny Eytan, Robert Greer, Mjaye Mazwi, and Peter Laussen, "Towards understanding ECG rhythm classification using convolutional neural networks and attention mappings," 08 2018.

[4] Jiacheng Wang and Weiheng Li, "Atrial fibrillation detection and ECG classification based on cnn-bilstm," 2020.

[5] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh, "ECG heartbeat classification: A deep transferable representation," *Proceedings of the IEEE International Conference on Healthcare Informatics (ICHI)*, Jun 2018.

[6] Jenny Venton, Peter M Harris, Ashish Sundar, Nadia AS Smith, and Philip J Aston, "Robustness of convolutional neural networks to physiological electrocardiogram noise," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2212, pp. 20200262, 2021.

[7] Rahul Kher, "Signal processing techniques for removing noise from ECG signals," *J. Biomed. Eng. Res*, vol. 3, no. 101, pp. 1–9, 2019.

[8] Shubhojeet Chatterjee, Rini Smita Thakur, Ram Narayan Yadav, Lalita Gupta, and Deepak Kumar Raghuvanshi, "Review of noise removal techniques in ECG signals," *IET Signal Processing*, vol. 14, no. 9, pp. 569–590, 2020.

[9] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008, pp. 1096–1103.

[10] Hsin-Tien Chiang, Yi-Yen Hsieh, Szu-Wei Fu, Kuo-Hsuan Hung, Yu Tsao, and Shao-Yi Chien, "Noise reduction in ECG signals using fully convolutional denoising autoencoders," *IEEE Access*, vol. 7, pp. 60806–60813, 2019.

[11] Chaitanya Sankavaram, Bharath Pattipati, Anuradha Kodali, Krishna Pattipati, Mohammad Azam, Sachin Kumar, and Michael Pecht, "Model-based and data-driven prognosis of automotive and electronic systems," in *IEEE International Conference on Automation Science and Engineering*, 2009, pp. 96–101.

[12] Naftali Tishby and Noga Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, 2015, pp. 1–5.

[13] "Ptb diagnostic ECG database," https://www.physionet.org/content/ptbdb/1.0.0/.

[14] "Mit-bih arrhythmia database," https://www.physionet.org/content/mitdb/1.0.0/.

[15] "ECG heartbeat categorization dataset," https://www.kaggle.com/shayanfazeli/heartbeat.

[16] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[18] V. V. Kuznetsov, V. A. Moskalenko, and N. Yu. Zolotykh, "Electrocardiogram generation and feature extraction using a variational autoencoder," 2020.

[19] Konstantinos Saitas Zarkias, Nikolaos Passalis, Avraam Tsantekidis, and Anastasios Tefas, "Deep reinforcement learning for financial trading using price trailing," in *Proceedings of the 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3067–3071.

[20] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis, "Deep adaptive input normalization for time series forecasting," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3760–3765, 2019.

[21] Nikolaos Passalis and Anastasios Tefas, "Global adaptive input normalization for short-term electric load forecasting," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 1–8.