PAC and PVC diagnosis for PPG signal using Deep Learning approaches

Niccolò Balestrieri[1], Andrea Bertogalli[2], Nicolò Tombini[3]

[1]Politecnico di Milano, Italy

niccolo.balestrieri@mail.polimi.it, andrea.bertogalli@mail.polimi.it nicolo.tombini@mail.polimi.it

Abstract -

This project explores the application of deep learning in analyzing Photoplethysmography (PPG) signals for cardiovascular pathology detection. Traditionally, manual feature extraction and classification faced limitations in capturing early-stage cardiac abnormalities. Deep learning eliminates the need for manual feature engineering, offering advantages in diagnostic accuracy and scalability. The study aims to develop binary and multi-class classifiers using deep learning techniques to distinguish between healthy individuals and those with cardiovascular abnormalities, such as Premature Atrial Complex (PAC) or Premature Ventricular Complex (PVC).

Keywords -

PPG; Hearth; PACs; PVCs

1 Introduction

In recent years, the advent of deep learning techniques has revolutionized various domains of biomedical signal processing, offering unprecedented opportunities for accurate diagnosis and prognosis. One such domain is the analysis of Photoplethysmography (PPG) signals, where deep learning algorithms have shown promising results in distinguishing between healthy individuals and those afflicted with cardiovascular abnormalities, such as Premature Atrial Complex (PAC) or Premature Ventricular Complex (PVC). PPG signals, acquired non-invasively through optical sensors, provide valuable insights into cardiovascular dynamics by capturing variations in blood volume during each cardiac cycle. Traditionally, analyzing PPG signals for pathology detection involved manual feature extraction and classification algorithms, which often proved limited in capturing the intricate patterns indicative of early-stage cardiac abnormalities.

The application of deep learning methodologies in PPG signal analysis offers several advantages over conventional approaches. Deep neural networks can automatically learn discriminative features directly from raw PPG data, eliminating the need for handcrafted feature engineering and potentially enhancing the diagnostic accuracy. Furthermore, the scalability of deep learning models allows for the integration of large-scale datasets, facilitating robust and generalized classifiers. This project aims to explore the efficacy of deep learning techniques in two distinct tasks within PPG signal analysis. Firstly, we endeavor to develop a binary classifier capable of accurately discerning between healthy individuals and those presenting with either PAC or PVC. Subsequently, we seek to extend this framework by constructing a multi-class classifier capable of differentiating among healthy individuals, those with PAC and those with PVC.

2 Materials and methods

In this section, we will provide an overview of the various phases, examining each one in detail. A brief overview is given by the below image:

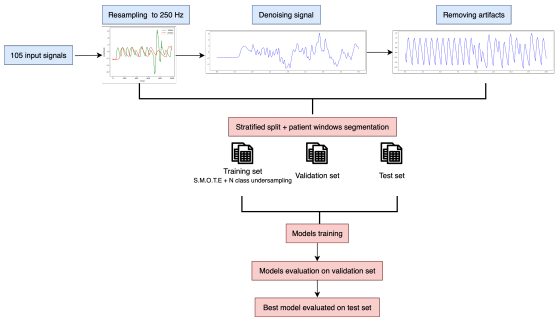

Figure 1. Project workflow

2.1 Dataset

The dataset comprises information from a total of 105 patients. The data includes the following characteristics:

- **Fs (Sampling Frequency):** The dataset consists of PPG signals collected at different sampling frequencies, with some recorded at 128 Hz and others at 250 Hz.
- **PPG Signal:** The primary data feature is the Photoplethysmography (PPG) signal, acquired non-invasively through optical sensors.

- **Systolic Peak Position:** Information on the position of the systolic peak in the PPG signal is included as a relevant parameter.
- **Labelling:** The dataset is labeled into three classes: N, S, and V. These labels correspond to different categories, potentially indicating normal (N) cases, as well as instances of abnormalities, such as premature beats classified as either S (premature systole) or V (premature ventricular contraction).

From the graph below, we can visualize the original distribution of data, predominantly residing in class N.

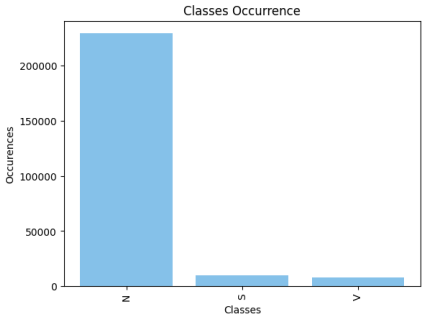

Figure 2. Data original distribution

2.2 Preprocessing

As is often the case when processing signals, especially medical signals like PPGs, the pre-processing phase is one of the most crucial stages. Various pre-processing operations are performed on the data, including:

- Frequency standardization
- Denoising
- Artifacts removal (outliers removal)
- Uninformative signals removal
- Dataset creation
- Signals normalization
- Dealing with data imbalance

2.2.1 Frequency standardization

Given that the signals were sampled at different frequencies (128Hz and 250Hz), it is advisable to resample them to a single frequency. Therefore, the signals have been resampled to 250Hz. The resampling is done with the Fourier method which, for upsampling, simply transforms the signal to the frequency domain and adds $\frac{N}{2}$ zeros at the end. Then the signal is transformed back to the time-domain. Obviously, with the signal rescaling, the peaks need to be rescaled too in order to match the right sample of the signal.

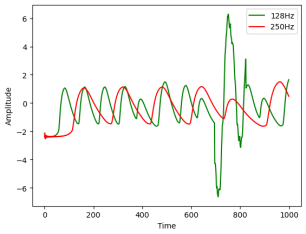

Figure 3. Resampling signal from 128 to 250 Hz

2.2.2 Denoising

Another important operation is to denoise the signals, this helps in improving the smoothness of the signal and improves the quality of the data. The possible noises we can find in a PPG signal are:

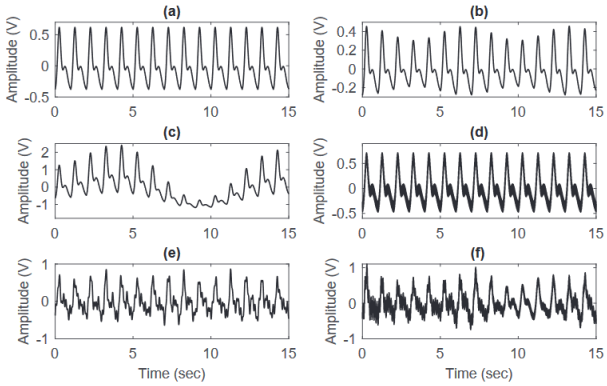

Figure 4. Idealised examples of clean and noise-corrupted PPG signals: (a) clean, (b) respiratory modulated, (c) baseline modulated, (d) power-line affected, (e) motion affected; (f) affected by all of the above

To denoise the signals we employed the wavelet transform [1]. To denoise a signal using wavelet transform, the signal is decomposed into wavelet coefficients. High-frequency coefficients, often associated with noise, are thresholded to remove unwanted components. The denoised signal is then reconstructed using the modified coefficients, effectively reducing noise while preserving signal features. Parameter choices, including wavelet type and threshold values, depend on the specific characteristics of the signal and noise.

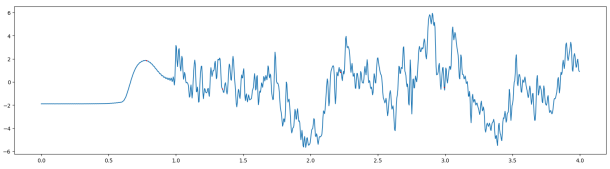

Figure 5. Original signal

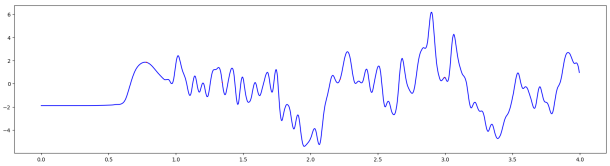

Figure 6. Denoised signal

2.2.3 Artifacts removal

After the denoising process we notice a lot of artifacts, so we decided to process the signal to remove them we applied two different thresholding methods:

- Double soft thresholding on amplitude.
- Hard thresholding on frequency spectrum.

Both methods are considered on a portion of the signal obtained with a sliding window (2 seconds, 500 samples), this because we noticed that the most of the signals are characterised by a succession of normal PPG portions and artifacts. Also, we noticed that the amplitude of the signal in the artifacts portions is much higher than the normal PPG signal.

Double soft thresholding on amplitude:
For each window we considered two thresholds, a lower and an upper one, then the percentage of the samples that exceeds one of the thresholds is thresholded again, this is done for a smoothing purpose, so that we avoid the risk of cutting out too much signal. The tuning of these thresholds has been done manually, and we have found that the optimal values are: min_amplitude = -4, max_amplitude = 4, patience_factor = 0.125.

Hard thresholding on frequency spectrum:
To further clean the signal from artifacts that do not exhibit high amplitude characteristics, an additional threshold has been applied, this time on the frequency bins obtained from the discrete Fourier transform of the window. This allows us to distinguish windows that vary rapidly (artifacts) from those containing normal PPG signals.

Finally, the windows are rejoined to obtain a clean signal for each patient. This cleaning operation naturally results in the loss of portions of the signals but, on the other hand, significantly enhances the data quality. Also, note that in this case some peaks need to be removed, and the remaining ones must be rescaled to match the correct samples of the signal.

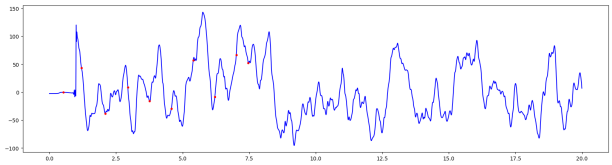

Figure 7. Denoised signal

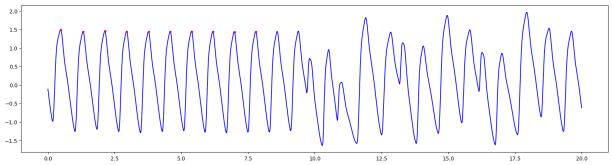

Figure 8. Denoised signal without artifact

2.2.4 Uninformative signals removal

Before proceeding with the actual dataset creation, the final operation performed on the entire signals was to remove the signals composed only of 'N' peaks, indicating signals from healthy patients. This was done because these signals are essentially non-informative, as 'N' peaks overwhelmingly dominate. Consequently, they need to be reduced in order for the models to learn sufficiently well from other types of peaks.

2.2.5 Dataset creation

For dataset creation, what has been done is to extract a window around each peak. Specifically, the window size is 1 second, equivalent to 250 samples. Additionally, some peaks, particularly those at the beginning and end of the signal, often require padding, which has been performed using zeros. An **important** aspect of dataset creation was that of **stratification**, which proved to be a non-trivial task. Indeed, we have to stratify a dataset by groups because we don't want windows extracted from the signal of the same patient to be in different datasets, as it would create correlation between the data in the training set and those in the validation or test set (we adopt a sort of stringent version with reference to the same patient beats in a set).
In addition to this, we need to stratify taking into account 3 types of labels, which makes the problem much more complex, note that there is no built-in method that allows stratifying the dataset while keeping the windows of each

patient grouped together. For this reason, an iterative method has been developed to maximize stratification of the datasets.

Iterative multi-class grouped stratified sampling:
To achieve the goal of stratification while keeping the windows grouped, we exploited the reproducibility of randomness seeds to find the best seed that, once fixed, provides the optimal dataset split. Therefore, the objective is to try many seeds until we find the best split. However, how do we determine if one split is better than another? To do this, we utilized the Kullback-Leibler Divergence between the label distributions in the two splits. The KL Divergence is defined as:

$$D_{KL}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{1}$$

So a split is considered improved with respect to the previous ones only if the KL divergence between the label distributions is lower than the previous one. The algorithm can be formalized with the following pseudocode:

Algorithm 1 Iterative multi-class grouped stratified sampling:

function STRATIFY(D, t, ids, m)
  $D_1, D_2, t_1, t_2, id_1, id_2 = Split(D, 0, t, id)$
  $KL = KLD(t_1, t_2)$
  for $i = 0$ to m **do**
    $S = Seed()$
    $D_{1_n}, D_{2_n}, t_{1_n}, t_{2_n}, id_{1_n}, id_{2_n} = Split(D, S, t, id)$
    $KL_n = KLD(t_1, t_2)$
    if $KL_n < KL$ **then**
      $KL = KL_n$
      $t_o = t_n$
      $D_1 = D_{1_n}$
      $D_2 = D_{2_n}$
      $t_1 = t_{1_n}$
      $t_2 = t_{2_n}$
      $ids_1 = ids_{1_n}$
      $ids_2 = ids_{2_n}$
  $Return\ D_1, D_2, t_1, t_2, ids_1, ids_2$

After the process we obtain a stratified distribution for the datasets.

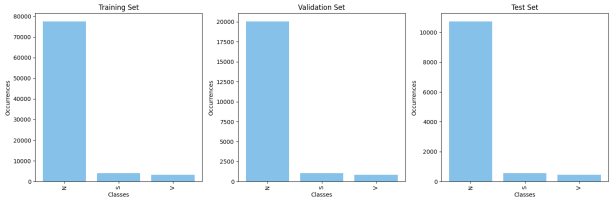

Figure 9. Three different distribution on train, validation and test set

2.2.6 Signals normalization

The windows were normalized using the z-score. For each signal, the mean (average of the average signal) was subtracted, and the signal was then divided by the standard deviation. As a standard practice, the mean and standard deviation were calculated based on the training set and then applied to the validation and test sets.

$$z = \frac{x - \mu}{\sigma} \tag{2}$$

2.2.7 Dealing with data imbalance

To address the class imbalance, primarily three techniques have been adopted, in addition to removing signals with only N peaks. These three techniques are:

- S.M.O.T.E. (Synthetic Minority Over-sampling Technique).
- Random undersampling.
- Wheighted loss.

Synthetic Minority Over-sampling Technique:
S.M.OT.E. is an oversampling technique, proposed by N.V. Chawla et. al in 2002 [2], where the synthetic samples are generated for the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together. Let x_i represent a minority instance. By identifying k nearest neighbors of x_i in the feature space, denoted as U_{x_i}, we proceed to randomly select one neighbor x_{nn}. Subsequently, a synthetic instance x_{New} is generated by combining the features of x_i and U_{x_i} according to the formula:

$$x_{New} = x_i + \lambda \cdot (x_{nn} - xi) \tag{3}$$

where λ is a randomly chosen value from the interval $[0, 1]$. This process is reiterated to produce the requisite number of synthetic instances. With S.M.O.T.E we increased by a factor 6 the minority classes S and V.

Random undersampling:
In addition to SMOTE, we applied random undersampling on windows representing N peaks to further reduce the imbalance. In this case we reduced the N by 35%.

Weighted loss:
Class weights in Keras are used to address imbalances in class frequencies within a classification problem. These weights are employed during the model training phase to give more importance to underrepresented classes compared to overrepresented ones. This is particularly

useful when dealing with imbalanced class distributions. During training, the total loss is computed by summing the losses of each sample. If w_i is the weight associated with class i, the total loss L_{Total} is calculated as:

$$L_{\text{total}} = \sum_{i=1}^{C} w_i \cdot L_i \tag{4}$$

where C and L_i is the loss associated with class i. The loss L_i depends on the specific loss function used.

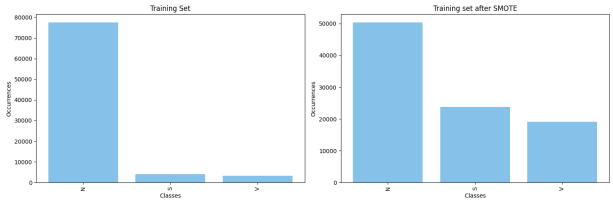

Figure 10. Training set after SMOTE & undersampling

2.3 Metrics

First of all, it is necessary to distinguish the metrics used based on the model created. However, we can confidently say in advance that the **accuracy** metric is certainly misleading due to the imbalanced nature of the dataset.

2.3.1 Binary classifier

The metrics considered are **Precision**, **Recall**, and **F1-score**, with particular attention to Recall, since we aim to minimize false negatives as much as possible.

2.3.2 Multi-class classifier

Regarding the three-class classifier, what has been decided to achieve the most realistic understanding of the model's performance is to calculate **Precision** and **Recall** for each class and subsequently the **micro F1-score** and the **macro F1-score**.
The **micro** approach gives equal importance to each observation. This means that, when the classes are imbalanced, those classes with more observations will have a larger impact on the final score, resulting in a final score which hides the performance of the minority classes and amplifies the majority.
Instead, the **macro** approach gives equal importance to each class. This means that a majority class will contribute equally along with the minority, allowing macro f1 to still return objective results on imbalanced datasets.

2.4 Deep Learning models

In this section, the models designed and utilized to solve the aforementioned task are shown below:

- VGG with Conv1D
- ResNet + CBAM with Conv1D
- EfficienNetB7 with Conv2D
- CNN + Bidirectional LSTM

Due to computational and time limit we did not perform any grid search with **EfficienNetB7** model.
For the other models, we performed a grid search in which we evaluated **batch size** and **minimum learning rate**.

2.4.1 VGG with Conv1D

The VGG-1D architecture consists of a stack of convolutional layers followed by max-pooling layers. Specifically, the network begins with an input convolutional layer with a kernel size of 3 and rectified linear unit (ReLU) activation function. Subsequent convolutional layers maintain the same kernel size and activation function, progressively increasing the number of filters to capture higher-level features. Max-pooling layers with a pool size of 2 and stride of 2 are interspersed between convolutional layers to downsample the input representation. This process is repeated to deepen the network, with additional convolutional layers and max-pooling operations. The architecture concludes with global max-pooling to aggregate spatial information and fully connected layers for classification.
On our validation set we reached:

BS	LR	P	R	F1
64	1e-08	0.65	0.83	0.72

2.4.2 ResNet + CBAM with Conv1D

ResNet, short for "Residual Network," is a type of deep neural network architecture used for image classification and other computer vision tasks. It was introduced by Kaiming He et. al in 2005 [3]. The ResNet architecture is designed to overcome the vanishing gradient problem that occurs when training very deep neural networks. This problem arises when gradients become too small as they are propagated backward through numerous layers, hindering proper weight learning in the network. The key feature of ResNet is the introduction of "residual blocks." Instead of learning the direct mapping from input to output, each residual block learns the difference (or residual) between the input and the output. This residual is then added back to the original input to obtain the block's output. More

formally, if $H(x)$ represents the expected output for a certain block, the actual output $F(x)$ of the residual block is calculated as:

$$F(x) = H(x) + x \tag{5}$$

This architecture can be easily modified with 1d convolutions to handle signals. Also, to further extend the network CBAM blocks can be added [4] this kind of blocks allow to introduce an attention mechanism. This allows the model to focus on specific parts of the signal enhancing the performances.

BS	LR	P	R	F1
512	1e-07	0.67	0.83	0.74

From this following chart we can observe how the losses were:

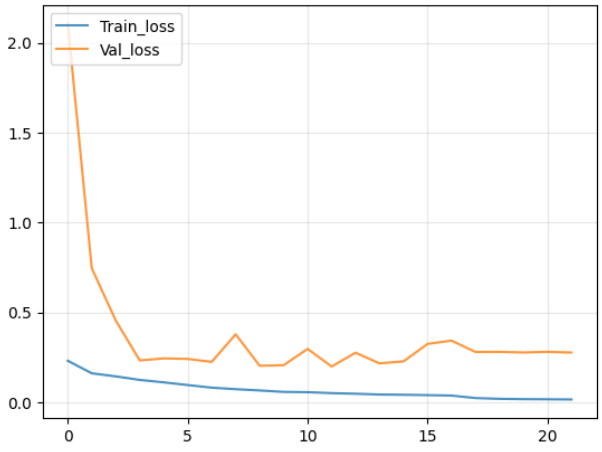

Figure 11. Train vs Validation losses

2.4.3 EfficienNetB7 with Conv2D

One of the approaches followed was to tackle the problem not as a signal classification problem but rather as an image classification problem [5]. This allows us to leverage well-established and pretrained neural networks. As one can imagine, this process involves transforming each window into an image. This transformation was carried out using three methods, each of which generates a single-channel image. Subsequently, these images are stacked depth-wise and fed into the model. The three methods used are:

- **Gramian Angular Field (GAF) conversion** [6]: This method converts a signal into a Gram matrix which is processed as an image in grayscale, the matrix definition follows this formula:

$$G(i,j) = \cos^{-1}\left(\frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|}\right) \tag{6}$$

where x_i and x_j are two samples of the signal.

- **Recurrence Plot (RP) conversion**[7]: In this case an image is constructed by exploiting the recurrence matrix. To compute the recurrence matrix for each pair of points in the signal, it is determined whether the difference between the values of the two points is less than a certain threshold. If so, the corresponding element in the recurrence matrix is set to 1; otherwise, it is set to 0. This process is repeated for all pairs of points in the signal. More formally we have:

$$\begin{cases} 1 & \text{if } \|x_i - x_j\| \leq \epsilon \quad (\text{ where } \epsilon \text{ is the threshold }) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where x_i and x_j are two samples of the signal.

- **Markov Transition Field (MTF) conversion** [8]: Let the signal be represented by a sequence of N (in our case 250) samples x_i, where $i = 1, 2, \ldots, N$, and each sample represents a discrete state. The Markov Transition Matrix P is a square matrix of size $M \times M$, where M is the number of possible discrete states in the signal. Each element $P(i,j)$ of the matrix represents the transition probability from state i to state j. More formally we define an element of P as:

$$P(i,j) = \frac{\text{Number of transitions from } i \text{ to } j}{\text{Total number of transitions from } i} \tag{8}$$

Note that in order to combine the images into a single image and feed them to a CNN, they were resized to a fixed size of 32 x 32.

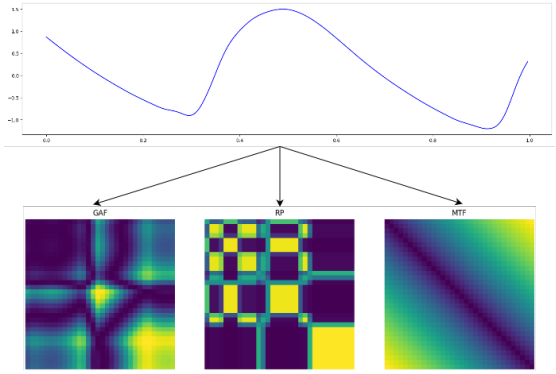

Figure 12. Three different transformations applied to the same window.

Regarding the architecture that constitutes the model, a pre-trained network, specifically EfficientNetB7 trained on ImageNet, is used. To this Feature Extractor, a Feedforward Neural Network (FFNN) with 2 layers of 512 units each is connected, and dropout is applied to mitigate overfitting. The training is done by fine-tuning the model so both the features extractor and the FFNN are trained with our data, this has been done to specialize the model on our task.
On our validation set we reached:

P	R	F1
0.59	0.76	0.66

2.4.4 CNN + Bidirectional LSTM

This architecture amalgamates 1D convolutional neural network (CNN) layers with bidirectional long short-term memory (Bi-LSTM) units to efficiently capture temporal patterns and dependencies within sequential data. The initial layers consist of a 1D CNN with 32 filters and a kernel size of 16, augmented by batch normalization and ReLU activation. These layers adeptly extract local features from the input time series data, enabling subsequent layers to discern meaningful patterns. Following the CNN layers, bidirectional LSTM units are employed to model long-range dependencies and temporal dynamics within the data. By processing sequences in both forward and backward directions, the model effectively captures contextual information, enhancing its understanding of the temporal structure.
On our validation set we reached:

BS	LR	P	R	F1
64	1e-08	0.52	0.91	0.66

The technique used consisted of finding the model among those presented above that could achieve better performance in distinguishing N patients from non-N patients. Subsequently, we began with the top-performing model to construct the three-class classifier. All the details are elaborated in depth in the following sections.

2.4.5 N vs (S-V) classifier

So, we performed a grid search in which we evaluated **batch size** and **minimum learning rate** for each model, and it emerged that ResNet + CBAM was the most performing one with the following parameters on our validation set:

BS	LR	P	R	F1
512	1e-07	0.67	0.83	0.74

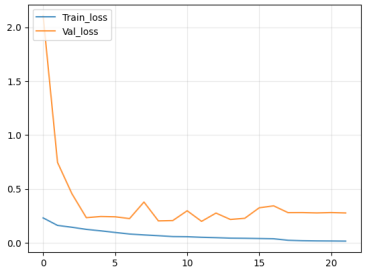

Figure 13. Train vs Validation losses

From this following chart we can observe how the losses were:
This is the confusion matrix of our two class classification model on our validation set:

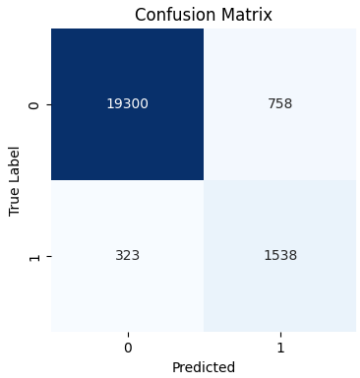

Figure 14. CM on validation set

From what we can see from the above confusion matrix, we can notice that our binary model recognizes healthy patients (N) well and moderately recognizes sick patients (S-V). This is also due to the validation set being heavily imbalanced with respect to the class of healthy individuals. Overall, the model exhibits a fairly high recall, which is positive since we want to minimize false negatives (those predicted as healthy but are actually sick).

2.4.6 N vs S vs V classifier

The next step involved designing a ternary classifier, which, this time, distinguishes all 3 types of heartbeats: N, S, and V. The chosen model architecture for this type was the best-performing one obtained in the binary task, ResNet + CBAM.
In this section, the technique used to optimize the model involved performing a manual random search of dropout parameters and network depth to try to avoid overfitting.
On our validation set we obtained:

D	N	P0	R0	P1	R1	P2	R2
0.4	6	0.98	0.95	0.44	0.63	0.49	0.46

where **D** stands for dropout, **N** for network depth, and the other are Precision and Recall for each of the three classes. As stated before, we calculated also this metrics:

F1 Macro	F1 Micro
0.65	0.92

From this following chart we can observe how the losses were:

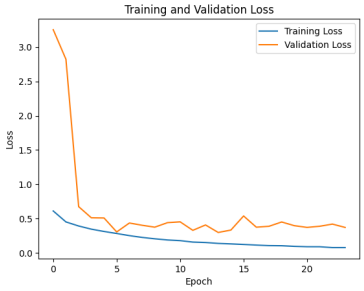

Figure 15. Train vs Validation losses

This is the confusion matrix of our three class classification model on our validation set:

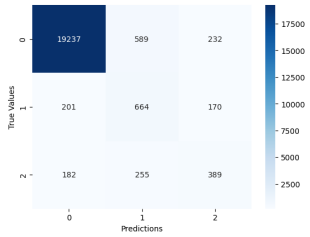

Figure 16. CM on validation set

As evidenced by the results of the metrics and the confusion matrix, the model struggles to recognize the various classes of patients, particularly class 2, which corresponds to the class of **PVC** patients.

2.5 Explainability

In the realm of signal processing, in this case for PPG signal, the quest for explainability often involves discerning which parts of the signal contribute most significantly to predictive outcomes. One prominent methodologies utilized for this purpose is **Grad-CAM++**.

2.5.1 Grad-CAM++

Grad-CAM++[9] (Gradient-weighted Class Activation Mapping++) aims to offer improved visual explanations of CNN model predictions compared to its predecessor (Grad-CAM), particularly in terms of more accurate object localization and elucidation of multiple occurrences of objects within a single image. The method is founded on a mathematical formulation that involves a weighted aggregation of positive partial derivatives extracted from the last convolutional layer feature maps concerning a specific class score.
The idea behind Grad-CAM++ is to reformulate the structure of weights w_k^c as

$$w_k^c = \sum_i \sum_j a_{ij}^{kc} relu(\frac{\partial Y^c}{\partial A_{ij}^k}) \tag{9}$$

In this way, w_k^c captures the importance of a particular activation map A^k.

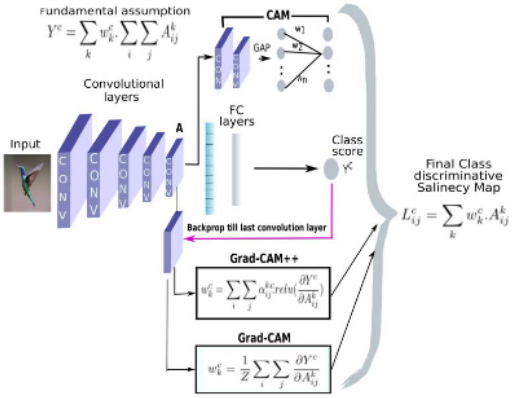

Figure 17. Grad-CAM++

The class-discriminative saliency maps for a given image, L^c is then calculated as a linear combination of the forward activation maps, followed by a relu layer. Each spatial element in the saliency map L^c is then computed as:

$$L_{ij}^c = \text{relu}\left(\sum_k w_k^c \cdot A_{ij}^k\right) \tag{10}$$

So, for this work it was employed ResNet + CBAM model for multi-class classification. It was provided 1 sample for each class N, S, V, and obtained 3 maps as output. These maps indicate, on a colored scale, the part of the signal that contributed the most to the prediction for each signal. The pictures below are the GradCam++ output for a correctly predicted sample for each class:

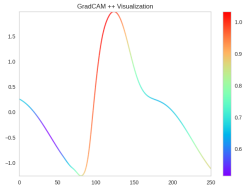

Figure 18. GradCam++ for the class N

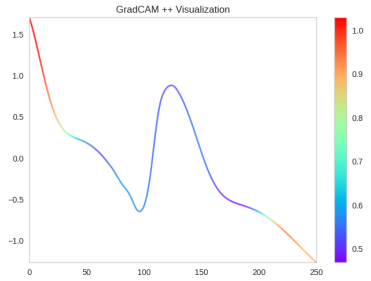

Figure 19. GradCam++ for the class S

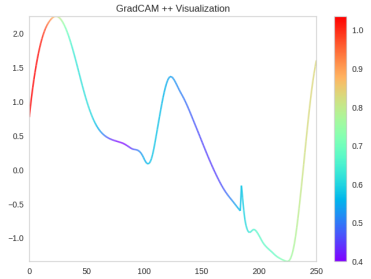

Figure 20. GradCam++ for the class V

3 Results

3.1 N vs (S-V) classifier

Regarding the binary classifier, we took ResNet + CBAM and retrained it by merging the training set and validation set with the best parameters found previously. The performances are shown below:

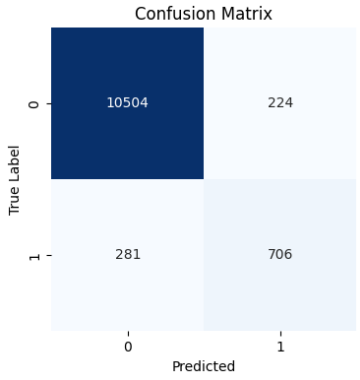

Figure 21. CM on test set for 2 class classifier

With these values as Precision, Recall and F1-score

P	R	F1
0.76	0.72	0.74

Compared to the results obtained with the validation set, the recall has decreased. This could be due to the fact that the model was trained by merging the train and validation sets, resulting in a greater imbalance than before.

3.2 N vs S vs V classifier

Regarding the ternary classifier, we took ResNet + CBAM and retrained it by merging the training set and validation set with the best parameters found previously. The performances are shown below:

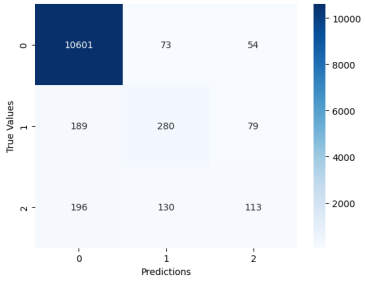

Figure 22. CM on test set for 3 class classifier

With these values of our metrics:

P0	R0	P1	R1	P2	R2
0.96	0.98	0.58	0.49	0.46	0.24

and

F1 Macro	F1 Micro
0.61	0.93

Overall these results on our test sets are more or less in line with our results obtained on validation set. The results especially on V and S class due to data imbalance and noise.

4 Discussion

The results may appear a little bit lower than expected also because the test patients have never been seen during training and may have worse splits than others. Additionally, it is important to consider that oversampling and undersampling were performed only on the training set, so as not to evaluate the model on data completely different from real-world data.

Another reason is certainly due to the fact that the signals provided exhibit a lot of noise and motion artifacts: indeed, as stated in this paper [10], PPG signal is very susceptible to noise and, in most cases, the noise in this signal is much worse than the ECG signal.

One final analysis we performed was to extract the final embeddings from ResNet + CBAM and utilize the **t-distributed stochastic neighbor embedding**[11] method to visualize them. Specifically, once the model was trained

to recognize the 3 classes, the embeddings were extracted by performing inference on the validation set. We aimed to reduce the embeddings from 512 to 2 and 3 dimensions to visualize how they were arranged in a significantly smaller dimensional space. As one might expect, the embeddings are not well separated and distinct from each other, as the model struggles to distinguish between the classes effectively.

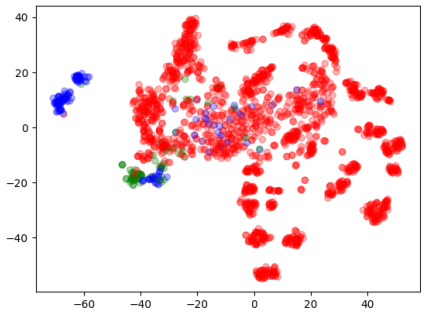

Figure 23. t-SNE in 2D

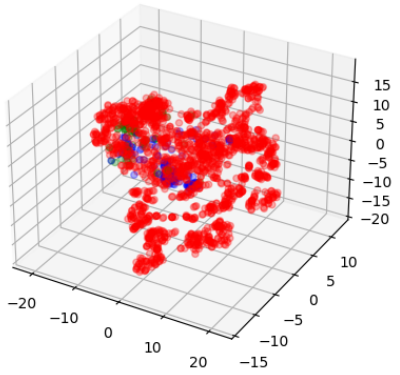

Figure 24. t-SNE in 3D

5 Conclusion

In conclusion, we are satisfied with the results obtained from training both the binary classifier for N vs. S and V and the ternary classifier. Comparing our outcomes with existing literature on similar tasks, the results demonstrate a good level of comparability.
Possible future developments related to this work could involve testing different preprocessing techniques and different models that better recognize classes of sick patients. These goals could also be achieved by analyzing other explainability techniques aimed at understanding which parts of the signal most strongly influence the network's decision.

References

[1] Elisa Mejia-Mejia, John Allen, Karthik Budidha, Chadi El-Hajj, Panicos A Kyriacou, and Peter H Charlton. Photoplethysmography signal processing and synthesis. In *Photoplethysmography*, pages 69–146. Elsevier, 2022.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Yana Luo and Zhongsheng Wang. An improved resnet algorithm based on cbam. In *2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pages 121–125. IEEE, 2021.

[5] Zeeshan Ahmad and Naimul Khan. A survey on physiological signal-based emotion recognition. *Bioengineering*, 9(11):688, 2022.

[6] Carmen Camara, Pedro Peris-Lopez, Masoumeh Safkhani, and Nasour Bagheri. Ecg identification based on the gramian angular field and tested with individuals in resting and activity states. *Sensors*, 23 (2):937, 2023.

[7] Donggeun Roh and Hangsik Shin. Recurrence plot and machine learning for signal quality assessment of photoplethysmogram in mobile environment. *Sensors*, 21(6):2188, 2021.

[8] Jian Liu, Shuaicong Hu, Ya'nan Wang, Qihan Hu, Daomiao Wang, and Cuiwei Yang. A lightweight hybrid model using multiscale markov transition field for real-time quality assessment of photoplethysmography signals. *IEEE Journal of Biomedical and Health Informatics*, pages 1–11, 2023. doi:10.1109/JBHI.2023.3331975.

[9] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[10] Aldrin Jozefan Parsaoran, Satria Mandala, and Miftah Pramudyo. Study of denoising algorithms on

photoplethysmograph (ppg) signals. In *2022 International Conference on Data Science and Its Applications (ICoDSA)*, pages 289–293. IEEE, 2022.

[11] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.