A deep–learning classifier for cardiac arrhythmias

Carla Sofia Carvalho
Hitachi Vantara
Lisbon, Portugal
carla.carvalho@hitachivantara.com

Abstract—We report on a method that classifies heart beats according to a set of 13 classes, including cardiac arrhythmias. The method localises the QRS peak complex to define each heart beat and uses a neural network to infer the patterns characteristic of each heart beat class. The best performing neural network contains six one–dimensional convolutional layers and four dense layers, with the kernel sizes being multiples of the characteristic scale of the problem, thus resulting a computationally fast and physically motivated neural network. For the same number of heart beat classes, our method yields better results with a considerably smaller neural network than previously published methods, which renders our method competitive for deployment in an internet–of–things solution.

Index Terms—Cardiac arrhythmias, electrocardiograms, convolutional neural networks.

I. INTRODUCTION

An industry domain that inherently produces data is the health domain, in particular the subdomain related to the monitoring of patients. Since cardiovascular diseases are the first cause of death worldwide, a common monitoring is that of the heart as a way to identify or prevent heart dysfunctions.

Heart dysfunctions are related to anomalies in the heart’s electrical activity, including cardiac arrhythmias, and can be diagnosed in electrocardiograms (ECG), produced in real time by portable devices. These records show the heart’s beating patterns in time as a result of differences in the electrical potential in the heart.

The interest thus lies in producing a physically motivated method that classifies heart beats from medically annotated ECG records in a fast and robust way, so that it can be deployed to generate alerts. Our suggested method encompasses the processing of ECG records, based on the location of characteristic features, and a classification model, based on a deep neural network, with the medical annotations providing the labels.

The advantage of neural networks over a heuristic model is that they search for the optimal combination of weights over different layers in sequence, which add non–linearity and can reproduce different functional forms. Neural networks have been used in the past to classify cardiac arrhythmias, using different number of heart beat types (e.g. Refs. [1]–[3]) and adopting complex architectures with difficult interpretability (e.g. Ref. [4]). In this paper, we test different architectures from previously published work [1, 6] and create new neural networks based on the scale of the problem, with view towards increasing the performance while simultaneously keeping the neural networks short and fast.

II. DATA PROCESSING

A. Data digitisation

We use publicly available data from the MIT–BIH Arrhythmia Database Directory, comprising 48 records. These records contain signals from two ECG channels (an upper signal and a lower signal) sampled at a frequency \( f_s = 1/dt = 360 \text{ Hz} \) for \( N \times dt = 30 \text{ min} \). These records also contain annotations by two cardiologists. Some records contain paced beats driven by a pacemaker or artifacts. We choose to use all 48 records, since the paced beats can work as an additional heart beat type and the artifacts can work as a noise component.
We also use the WFDB software package to read and process the file format that the records are encoded in.

B. Heart beat locations

The signals consist of readings of the heart potential in time. The heart potential contains characteristic peaks, namely the P peak, the QRS peak complex and the T peak, which correspond to the polarisation/depolarisation heart cycle. The first step consists in identifying the QRS peak complexes in time, which are usually more prominent in the upper signal.

The annotations are located at the QRS peak complex and provide the labels to the heart beats. Hence the next step consists in chopping the signals about each QRS peak complex so that each fraction contains an individual heart beat (Fig. 1). Each record has a characteristic beat length between consecutive QRS peak complexes. We choose the median characteristic beat length ($\text{len} = 256$) so that the resulting chopped signals can be concatenated into a matrix $x_{ijk}$, where $i \in \{1, ..., \text{n\_sample}\}$ is the number of resulting chopped signals, $j \in \{1, ..., \text{len}\}$ is the length in time of each chopped signal and $k \in \{\text{sign1, sign2}\}$ indicates the signal.

C. Heart beat annotations

The possible values of the annotations define the set of heart beat classes. The records in the MIT-BIH Arrhythmia Database Directory follow the annotation system such that beat annotations take the values \{N, L, R, B, a, J, A, S, j, e, n, V, F, E, /, f, Q, ?\} and non–beat annotations take all the other possible values.

We produce the distribution of heart beat classes across the records (Fig. 2, top panel). We observe that the classes \{B, n, r, ?\} are not represented in this data set. We also observe that the beat classes are not all equally represented. Hence we set an upper bound to the number of occurrences per heart beat class (here $\text{n\_row\_max} = 4000$), so that all beats from under–represented classes are included but only a fraction of the beats from over–represented classes is included. We also observe that the classes \{S, E\} contain less than six elements, which is the minimum number of elements required to balance the representation of a given class (Sec.III). Note that some values do not correspond to heart beat classes, e.g. \{Q\} corresponds to unclassifiable beats and \{/, f\} corresponds to paced beats; we choose to keep them to add robustness to the model. Hence the heart beat classes that can be classified are reduced to \{N, L, R, a, J, A, j, e, V, F, Q, /, f\}, totalling 13 classes.

We encode the annotation values into binary vectors with length equal to the number of classes, resulting in a matrix $y_{ic}$, where $i \in \{1, ..., \text{n\_sample}\}$ is the number of heart beats and $c \in \{1, ..., \text{n\_class}\}$ is the number of heart beat classes.

III. DATA ENGINEERING

A. Data resampling

Since the heart beat classes are not all equally represented, we re–sample the data by generating synthetic beats belonging to the under–represented classes, thus producing a new data set with balanced classes. We use the Synthetic Minority Oversampling Technique (SMOTE) as implemented in the imbalanced–learn package. When we resample data, the training set undergoes resampling, whereas the test set does not, so that the performance metrics refer to the original class distribution.

B. Generation of new variables

From the original signals, we can generate new variables that encode potentially useful information, e.g. the first discrete derivative of the smoothed signals $dx_{ijk}$ and the Fourier transform of the chopped smoothed signals $x_{ijk}$. The Fourier transform of a signal contains both positive–frequency and negative–frequency components; hence, instead of $x_{ik}$, we use the amplitude spectrum $|x_{ik}| = \sqrt{x_{1ik}^2 + x_{2ik}^2}$. The result of the concatenation of the original variables with the generated variables is the data matrix $z_{ijk} = \{x_{ijk}, dx_{ijk}, |x_{ijk}|\}$ such that, for $n\_\text{chann} = 2$, the data matrix $z_{ijk}$ consists of $3 \times n\_\text{chann} = 6$ variables.

C. Selection of variables

We compute the correlation between each pair of variables indexed \{k1, k2\} with values \{x_{ijk1}, x_{ijk2}\}, which we denote by $\text{Corr}_{k1,k2}$ (Fig. 3, top panel). We also compute the correlation of each variable $k1$ with the beat annotations $y_{ic}$, which we denote by $\text{Corr}_{k1,y}$ (Fig. 3, bottom panel).

---

[1] https://archive.physionet.org/physiobank/annotations.shtml
[2] https://imbalanced-learn.readthedocs.io/en/stable/index.html
[3] https://wfdb.readthedocs.io/en/latest/
By setting an upper bound (e.g. $\text{corr}_\text{max} = 0.9$), we use the correlation between each pair of variables as a measure of redundancy. Variables $\{k_1, k_2\}$ such that $|\text{Corr}_{k_1,k_2}| > \text{corr}_\text{max}$ are classified as redundant, hence one of them can be removed without loss of information. From Fig. 3 top panel, no removal is justified on the basis of redundancy.

By setting a lower bound (e.g. $\text{corr}_\text{min} = 0.1$), we use the correlation between each variable with the beat annotations as a measure of relevance of that variable in predicting annotations. A variable $k_i$ such that $|\text{Corr}_{x_i,y}| < \text{corr}_\text{min}$ is classified as irrelevant, hence it can be removed without loss of information. From Fig. 3 bottom panel, the variables $\{dx_{ij,k}=\text{sign}1, dx_{ij,k}=\text{sign}2, |X_{ij,k}=\text{sign}2|\}$ can be removed on the basis of relevance.

IV. CLASSIFICATION MODELS

A. Neural networks

Since the heart beats in the data are labelled, we look for a classification model to infer the patterns common to heart beats in the same class. Given the nature of the data, neural networks (NN) either of the recurrent type (RNN) or the convolutional type (CNN) will be adequate models.

An RNN regards each heart beat as a sequence of data points in time and combines the value at the previous instant with a transformation of earlier values. A variation of RNN is the large–short–term–memory (LSTM) NN. Each heart beat must be further divided into sublengths of the original beat length so that the different sublengths are regarded as a sequence of data points. Since each heart beat has size $\text{len}$ and we are looking for three peak–like structures, then the characteristic sublength will be $\text{len}/4 = 64$. For sublengths, we use fractions of the characteristic sublength, in particular $\text{sublen} \in \{1/8, 1/4, 1/2, 1\} \times \text{len} = \{32, 64, 128, 256\}$.

A CNN regards each heart beat as a one–dimensional image and operates one–dimensional convolutions (Conv1D) over the kernel. Since the characteristic sublength is $\text{len}/4$, then the largest scale will be the length corresponding to the Nyquist frequency, hence $\text{len}/4/2 = 32$. For kernel sizes, we use powers of two between 4 and 32; for stride step, we use $\text{stride} = 1$.

For optimizer, we use ADAM (with the default values $\text{lr} = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \text{decay} = 0$); for loss function, we use categorical cross entropy; for number of epochs, we use $\text{n\_epochs} = 25$; for batch size, we use $\text{batch\_size} = 64$; for activation function, we use the rectified linear unit (ReLU).

B. Cross–validation

We devise a cross–validation scheme so that each classification is trained on a manageable sized training set. We first divide the entire data matrix $x_{ijk}$ into $nk = 5$ subsets, each of which preserving the proportion among the different classes as $x_{ijk}$. We keep one of the $nk$ subsets as test set with the original class distribution and resample the remaining $nk – 1$ subsets, producing the resampled data. We then divide the resampled data into $nk' = 3$ subsets, one of which serving as the resampled input data. We then divide the resampled input data into $nk'' = 5$ subsets, one of which serving as resampled test set and the remaining serving as resampled training set. We rotate the resampled training data set over the $nk''$ subsets so that the fitting of the classification model is done $nk''$ times on different training sets. We then rotate the resampled input data over the $nk'$ subsets so that the fitting of the classification model is done $nk'' \times nk'$ times on different training sets.

We average the $nk'' \times nk'$ classification predictions over $nk''$, yielding $nk'$ mean predictions of the test data. We then average the $nk'$ classification predictions, yielding one confusion matrix labeled by the corresponding $nk'$ mean accuracies.

The fitting resulting from each of the $nk'' \times nk'$ subsets is applied to the test data, thus producing $nk'' \times nk'$ classification predictions for each element in the test data. The combined prediction is the mean over the $nk'' \times nk'$ classification predictions, whose resulting accuracies we include in the tables.

C. Selection of neural network architecture

In order to select an adequate NN architecture, we try different architectures for RNN and CNN [5]. We use the implementation from TENSORFLOW[6] via the application programming interface KERAS[7].

[1] https://www.tensorflow.org
[2] https://keras.io
We represent the NN architectures schematically as sequences of layers, with Input representing the input data and Output representing the predicted classification. We explore four types of architectures:

- **a) architecture with LSTM layers, named LSTM:**
  
  LSTM : Input → LSTM → Dropout → Dense → Output;  

- **b) architecture with Conv1D layers, named Conv:**
  
  Conv : Input → Conv1D → Dropout → MaxPool → Flat → Dense → Output;  

- **c) architectures that combine both LSTM and Conv1D layers, named Conv + LSTM and ConvLSTM:**
  
  Conv + LSTM : Input → Conv1D → Dropout → MaxPool → Flat → Dense → LSTM → Dropout → Dense → Output,
  
  ConvLSTM : Input → ConvLSTM → Dropout → Flat → Dense → Output.  

We first test these architectures for the minimal NN formulation and for approximately the same number of layers, setting the kernel size to kernel_size = 4 and the number of filters to n_filter = 64. As measures of performance, we use the total number of true positives (TP) and the number of predicted classes; as measure of efficiency, we use the running time (Table I).

We observe that the Conv architecture yields the best performance and the shortest running time; this prompted us to consider Conv for further study. We also observe that most heart beats belonging to the classes \{/, A, L, N, R, a, e, j\} are correctly classified by all architectures. The goal is now to increase the number of TP of the other classes, namely \{F, J, V, f\}.

### D. Selection of convolutional neural network architecture

We test the Conv architecture for different number and organisation of layers. We explore eight NNs. In the Conv1D layers, we first set kernel_size = 4 and padding = valid, and vary the number of filters within the range n_filter ∈ \{16, 32, 64\}, starting at 16 and doubling every time that the Conv1D layer is preceded by a pooling layer. As measure of performance, we use the total TP.

We start with the NN suggested in Ref. [1], since this NN was conceived to classify heart beats from the same database. We keep the architecture as shown below, named Acharya:

Acharya : Input → Conv1D (ReLU) → Dropout → MaxPool → Conv1D (ReLU) → Dropout → MaxPool → Conv1D (ReLU) → Dropout → MaxPool → Flat → Dense (ReLU) → Dense (ReLU) → Dense (Softmax) → Output.  

Comparing Acharya (Fig. 4, left panel) with the previous best performing NN, the TP of \{f\} increases significantly, whereas the TP of \{F, V\} decrease, with the total TP staying approximately the same.

We change Acharya by moving the drop–out layers from after the Conv1D layers to after the dense layers, which we name Acharya_2. Comparing Acharya_2 with Acharya, the TP of \{F\} increases, whereas the TP of \{V, f\} decrease, with the total TP decreasing, thus a worsening in performance.

We test the NN suggested in Ref. [1], since this NN was conceived to estimate cosmological parameters, which requires looking for different scales in the data. We keep the architecture as shown below, named Gupta:

Gupta : Input → Conv1D (ReLU) → AveragePool → Conv1D (ReLU) → Conv1D (ReLU) → AveragePool → Conv1D → AveragePool → AveragePool → Flat → Dense (ReLU) → Dropout → Dense (ReLU) → Dropout → Dense (ReLU) → Dropout → Dense (Softmax) → Output.  

This NN contains contiguous convolutional layers without intermediate pooling layers, forming a block of two Conv1D layers. Comparing Gupta with Acharya, the TP of \{F\} increase but the TP of \{V\} decreases, with the total TP decreasing, thus no improvement in performance.

We change Gupta by adding another set of contiguous Conv1D layers without intermediate pooling layers, as shown below, named Gupta_2:

Gupta_2 : Input → Conv1D (ReLU) → AveragePool
E. Selection of convolutional neural network hyper-parameters

We explore further the Gupta_2 NN by varying some hyper-parameters. As measures of performance, we use the average accuracies, the accuracy of the resulting accuracy and the total TP.

a) Type of pooling layer and type of padding: We test the type of pooling layer and the type of padding simultaneously. For each type of pooling layer, we vary the type of padding in the Conv1D layers, while keeping the kernel sizes per block equal to kernel_size = \{4, (4, 4), (4, 4, 4)\}. We observe that AveragePool with padding = same yield the best performance.

b) Number of dense layers: We vary the number of dense layers within the range n_dense \in \{1, 2, 3, 4, 5, 6\}, where the number of drop-out layers is n_drop = n_dense - 1, while keeping the number of filters equal to n_filter = 64. We observe that Gupta_2 yields an increase in the total TP up to n_dense = 4 and a decrease for n_dense > 5. Since the difference in performance between n_dense = 4 and n_dense = 5 is not significant, we keep the original Gupta_2 with n_dense = 4 dense layers due to its simplicity.

c) Kernel sizes of each convolutional layer: We vary the kernel sizes of each block over the range \{4, 8, 16\}, arranging these three values in combinations respectively of one, two and three values in consecutive order. With these constraints, the best performing NNs of the Gupta_2 type are for

\[
\text{kernel}_\text{size} \in \{(4, (4, 4)), (4, 8, 16)\}, \{(4, (8, 4)), (8, 16)\}, \{(8, 4, 4), (16, 16)\}, \{(16, 8, 4), (16, 8, 4)\}, (4, 4, 4)\}. \quad (8)
\]

While all these NNs yield average accuracies between 0.7 and 0.8, the NN with kernel_size = \{16, (8, 4), (4, 4, 4)\} yields the largest total TP and largest resulting accuracy. This suggests that the important features for the classification of heart beats are best captured in groups of 4, 8 and 16 data points, with the convolutions proceeding from large to small scales. These scales correspond to 4, 8, 16 × dt =
ventricular contraction”), hence the neural network is confounding between two ventricular arrhythmias. The next worse classified classes are: a) \{J\} (“Nodal (junctional) premature beat”), which is mostly classified as \{R\} (“Right bundle branch block beat”); b) \{A\} (“Atrial premature beat”), which is also classified as either \{Q,R,V,a\} (where \(a\) stands for “Aberrated atrial premature beat”); and c) \{f\} (“Fusion of paced and normal beat”), which is also classified as either \{f, Q\} (respectively “Paced beat” and “Unclassifiable beat”).

B. Comparison with results from other published work

We compare our results with those of other NNs developed in previously published work to classify cardiac arrhythmias from annotated ECG records (Table II).

On the properties of the NNs, we note that most suggested neural networks have around 10 layers, with the exception of the neural network suggested in Ref. [4] which has over 30 layers. Such a size is not justified by the authors.

On the number of arrhythmias, Refs. [1], [2] used another annotation system consisting of five classes only. References [3], [4] used the MIT-BIH adopted annotation system.

On the performance metrics, while Refs. [1], [2] used data resampling, it is not clear whether the reported results relate to a test subset of the original data or to the resampled data.

In conclusion, our results are better than those obtained in Ref. [4] for about the same number of classes and less than a third of the number of layers, albeit worse than the results obtained for about half the number of classes with about the same number of layers. This attests to the competitiveness of our method and its adequateness for a solution deployment.

REFERENCES

[1] Acharya U.R., Oh S.L., Hagiwara Y., Tan J.H., Adam M., Gertych A. and Tan R.S., “A deep convolutional neural network model to classify heartbeats,” Comput Biol Med, Oct 2017

[2] He Z., Zhang X., Cao X., Liu Z., Zhang B., Wang X., “LiteNet: Lightweight neural network for detecting arrhythmias at resource-constrained mobile devices,” Sensors 2018

[3] Jun T.J., Nguyen H.M., Kang D., Kim D., Kim Y.H., “ECG arrhythmia classification using a 2–D convolutional neural network,” 2018 (arXiv:1804.08812 [cs.CV])

[4] Rajpurkar P., Hannun A.Y., Haghpanahi, M., Bourn C. and Ng. A.Y., “Cardiologist–level arrhythmia detection with convolutional neural networks,” 2017 (arXiv:1707.01836 [cs.CV])

[5] Brownlee J., https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/, accessed 18 June 2019

[6] Gupta A., Matilla J.M.Z., Hsu D. and Haiman Z., “Non–Gaussian information from weak lensing data via deep learning,” 2018 (arXiv:1802.01212 [astro-ph.CO])

Table II

Comparison with previously published work. Column 1: Identification of the work. Column 2: Identification of the network size. Column 3: Identification of the number of classes in the data. Columns 4-6: Identification of the performance metrics for the best performing network, where accuracy is the resulting accuracy, and \(\text{(Precision)}_{w}\) and \(\text{(Recall)}_{w}\) are respectively the mean precision and mean recall over all classes, weighted by the size of each class.

| Work                        | Network size | No. classes | Accuracy | \(\text{(Precision)}_{w}\) | \(\text{(Recall)}_{w}\) |
|-----------------------------|--------------|-------------|----------|---------------------------|-------------------------|
| Acharya et al. (2017) [1]   | 3 Conv1D + 3 Dense | 5           | (0.935, 0.940) | (0.979, 0.979) | (0.960, 0.967) |
| He et al. (2018) [2]        | 9 Conv1D + 2 Dense | 5           | (0.979, 0.988) | -             | -                       |
| Rajpurkar et al. (2018) [3] | 33 Conv1D + 1 Dense | 14          | 0.990     | 0.986                    | 0.978                    |
| Carvalho (2020)             | 6 Conv1D + 4 Dense | 13          | 0.821     | 0.848                    | 0.822                    |