university of groningen

Classifying pathological heartbeats from ECG signals

Course: Machine Learning [WMAI010-05]

Authors:

Filippos Andreadis (*s4942906*)
Christodoulos Hadjichristodoulou (*s5223970*)
Chrysoula Maria Nampouri (*s4721810*)
Marios Souroulla (*s4765125*)

Course Lecturer:

Prof. Dr. H. (Herbert) Jaeger

Group 23

University of Groningen

Faculty of Science and Engineering

Groningen, the Netherlands

February 5, 2023

Abstract

Obtaining information about the electrical activity of the heart in the form of electrocardiograms (ECG) has become a standard way of monitoring patients' heart rhythm and function. It is used for diagnosing a variety of cardiac anomalies such as arrhythmia and other heart diseases. However, the interpretation of ECGs requires the expertise of trained physicians, thus bearing the need for tools that automatically classify such signals. In this study we train deep convolutional neural networks (CNNs) to perform binary classification of ECG beats to normal and abnormal. We use transfer learning in order to build models that are fine-tuned on specific patients' data, after pre-training a generic network on a set of different ECGs selected from the MIT-BIH arrhythmia database. We then compare the performance of the fine-tuned networks against that of individual networks, which are trained only on the ECG data of a single patient, in order to evaluate the overall efficacy of transfer learning on the given problem. We managed to achieve adequate results on both scenarios as the individual classifiers yielded an average of 94.6% balanced accuracy on the test set, whereas the fine-tuned models a marginally worse 93.5%.

Keywords— ECG classification, CNN, Transfer learning

Contents

1 Introduction

ECG analysis is a non-invasive and low-cost method, regularly used by healthcare professionals to detect cardiac anomalies of patients. Since cardiovascular diseases are one of the leading causes of death worldwide, ECGs have become a very important tool aiding physicians in the diagnosis of such life-threatening conditions.

Computer-based systems that facilitate the automatic interpretation of ECGs have already been introduced [1], relieving professionals from extra workload, reinforcing healthcare decisions while also reducing overall costs of treatment. Additionally, the recent prevalence of wearable smart-devices has enabled the development of effective methods of remote monitoring and diagnosis [2, 3]. Even though various approaches for the classification of ECG signals have been proposed, including Fourier transform [4], wavelet transform [5] or hidden Markov models [6], deep learning techniques currently comprise the state-of-the art [7]. However, it goes without saying that limitations to these approaches exist. Despite the abundance of recorded ECG data, the process of annotation relies entirely on the knowledge of domain experts meaning that it is a costly and time-consuming procedure. This scarcity of labelled ECGs along with the fact that deep neural networks typically require large amounts of training data, constitute an important inhibitory factor. Moreover, many abnormal cardiovascular events occur infrequently, resulting in imbalanced datasets, while the morphology of a signal greatly depends on both the recording instrument used and the different physiology of each patient.

Due to the aforementioned shortcomings, training a general-purpose classifier that would be able to directly identify the abnormal beats of any previously unseen ECG trace of a new patient is currently a very ambitious task. Hence, a more accessible challenge is to train individual models, that are tailored to single patients with the purpose to classify abnormalities within their own particular ECG [8].

In our study, we have chosen to train 1-dimensional CNNs and create separate classifiers for a set of specific patients by the means of transfer learning. The dataset used is the MIT-BIH Arrhythmia Database [9], the first widely available set of ECG material used for testing and basic research in cardiac dynamics for the last four decades. Our goal has been to pre-train a base network on a set of multiple ECGs and eventually fine-tune separate models on each one of them. We aimed to observe if the CNN was able to capture information regarding general patterns present in the aggregated data and whether it could benefit from that when tasked to recognise the abnormalities of a particular instance in the dataset. We approach this by formulating a binary classification problem and creating two separate train-test scenarios. Specifically, we aggregate all the different types of cardiovascular abnormalities together, hence labelling any given ECG window either as normal or abnormal. We then pre-train a generic network on the entire dataset. Additionally, we train individual models for each one of the patients, using only their own ECG signal for both training and testing. Finally, we fine-tune the pre-trained model on each patient, obtaining a separate network for each case that is tested on the corresponding ECG signal. This way we are able to compare the performance of the fine-tuned networks against the individual classifiers and get an insightful understanding of how much transfer learning has affected the classifying capabilities of the CNN.

2 Data

In this section, we provide an overview of the original dataset and explore its properties. Next, we present our pre-processing pipeline, following the whole process the data undergoes from its raw format to our desired representation.

2.1 Dataset

The dataset we used for our project is the MIT-BIH Arrhythmia Database [9], which has seen widespread use for the evaluation of arrhythmia detectors for decades. It consists of 48 half-hour excerpts of two-channel 24-hour electrocardiograms recorded at 360 hertz. Each one of these recordings is accompanied by a series of annotations that categorize each beat of the ECG into a range of possible classes as determined by the experts.

In order to read the database in its original format, special tools are needed. Conveniently, a version of it is also available in text format at [10], which we are using. This version of the database contains two text files for each ECG recording. The first one is a comma-separated file with the raw signal values for the two channels at each sample. The second one contains the information regarding the classification of each beat, which is done by annotating the sample that corresponds to the peak of that beat. Figure 1 displays an indicative signal for a short period along with its corresponding annotations.

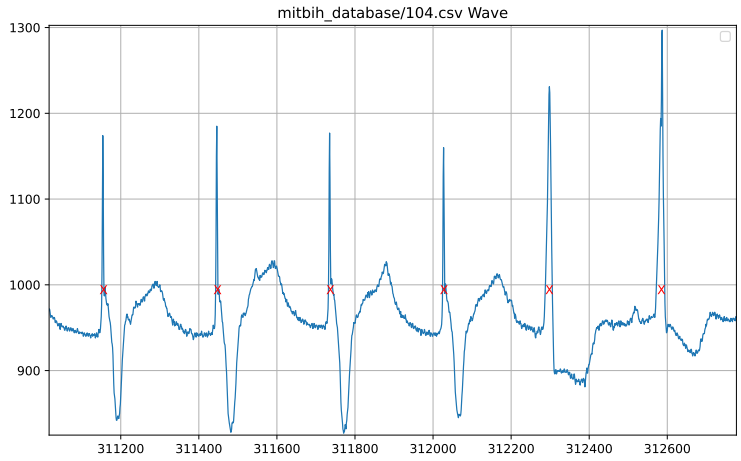

Figure 1: Short excerpt of the signal retrieved from patient 104 and the respective annotated samples, indicated with a red cross.

All the possible categories for each beat are listed and explained in Table 3 in the appendix. In Figure 2 we present the distribution of the annotations among all recordings. It immediately becomes obvious to the observer that the normal beats (denoted as *N*) vastly dominate over the rest.

2.2 Pre-processing

Before any of this data can be used, it needs to undergo a series of pre-processing steps in order to transform it into a format that is suitable for our task. The first step is keeping only one of the two channels, as in the relevant work of Hadaeghi [8], since they are highly correlated and thus, keeping both would be redundant. The raw signal we get from this channel is then normalized on a per-recording basis to values in the range $[0,1]$. In addition, ECG data are often corrupted with signals generated from the power supply of the recording instrument and have slow drift components. To combat this, all signals are passed through a Butterworth frequency filter with a low cutoff frequency of 0.4Hz and a high cutoff of 30Hz.

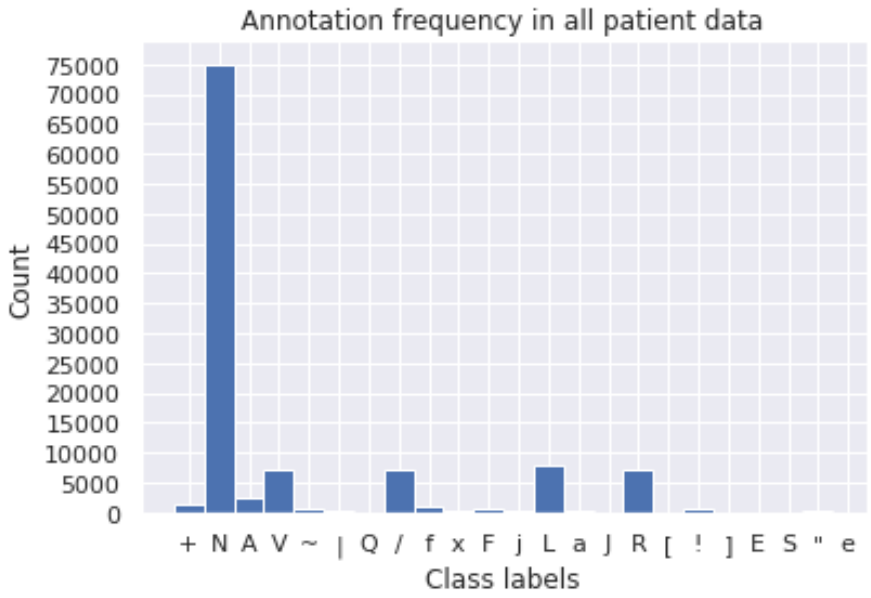

Figure 2: The distribution of the annotations for all patients.

To conform with the goals of this project, a transformation should take place from the current time-series format to fixed-size intervals, each encompassing a beat. We do this by splitting the signal into 1-second non-overlapping windows, which translates to 360 samples per window due to the 360Hz frequency of sampling. We chose this window length because it is very close to the average length of a heartbeat. Each window is then labeled with a 1 or a 0, depending on whether that window contains an abnormal beat or not. Figure 3 showcases an example of a negative class window. Windows which contain only unclassified beats are discarded from the final dataset. After this process takes place, we resample each window to 128 samples in order to lower the dimensionality of the input data.

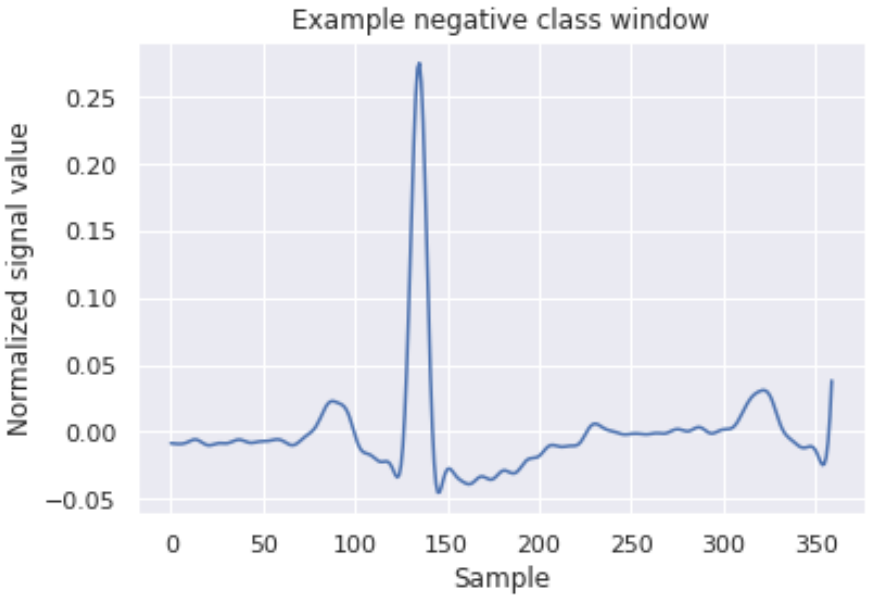

Figure 3: An example of a negative class window from patient 104 before resampling.

Applying the process described above, we observed that the distribution of positive and negative windows for some patients was extremely skewed towards one of the two sides, making them non-suitable for use as training data. In response to this, we decided to discard some patients from our experiments based on whether or not the percentage of one class of windows is less than 0.9%. This filtering results in

the following list of 29 patient recordings that we used: [’100’, ’102’, ’104’, ’105’, ’106’, ’108’, ’114’, ’116’, ’119’, ’200’, ’202’, ’201’, ’203’, ’205’,’208’, ’209’, ’212’, ’210’, ’213’, ’215’, ’217’, ’219’, ’220’, ’222’, ’221’, ’228’, ’223’, ’231’,’233’]. After the whole process, the final dataset consists of 49245 rows and 128 columns, accompanied by 49245 binary labels.

Regarding these 29 patients, in figure 4 we present the distribution of the two classes over the whole final dataset. Even after merging all the abnormalities to a single positive class, the normal beats still dominate. Figure 5 presents the rate of positive class windows for each one of the selected patients. As expected, the ratio between positive and negative windows is not the same over different patients.

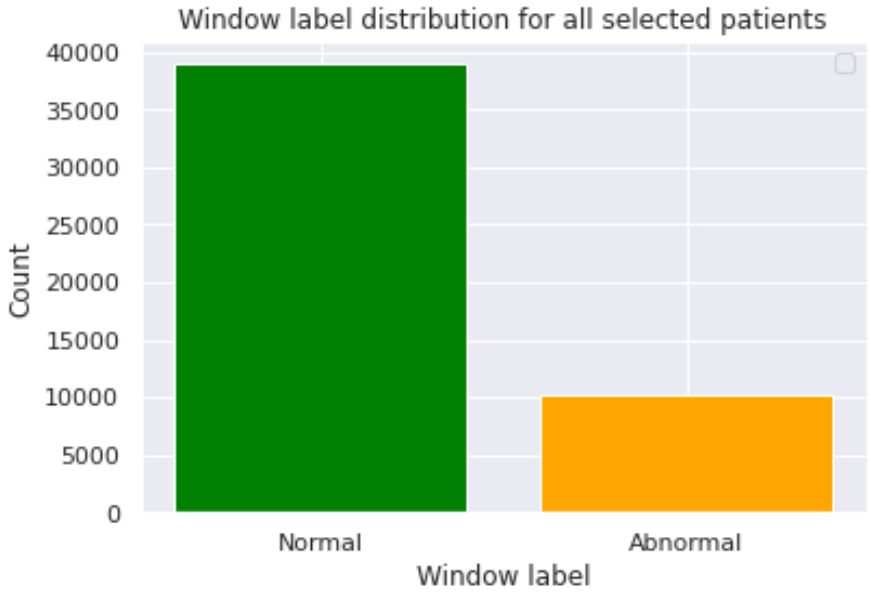

Figure 4: Distribution of positive and negative windows over the final dataset

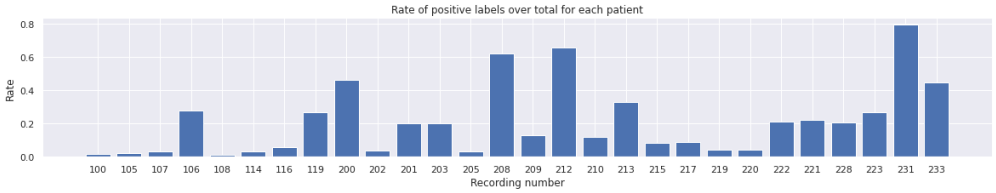

Figure 5: Rate of positive class windows for each patient in the final dataset

3 Methods

Transfer learning is a machine learning technique where a model trained on one task is re-purposed or fine-tuned to perform another similar task. It is a viable solution to overcome the challenge of limited training data, as the pre-trained model will have already learned rich representations from large amounts of data and thus, will provide a strong initialization that can help the model on the target task quickly converge to a good solution. This renders transfer learning a widely used approach in various fields such as computer vision, natural language processing, and speech recognition where collecting and annotating large amounts of data can be extremely challenging and time-consuming.

In this study, we are working on a binary classification problem of ECG data. As discussed in Section 2, the aim is to classify one-dimensional fixed-size windows of patients’ pulses as either normal (0) or abnormal (1). Our target –downstream– task treats each patient as a unique individual, taking into consideration their individual characteristics and properties. This means that we aim to provide tailored and personalized models and predictions to each patient separately, acknowledging that the notion of ‘normal’ regarding beats can vary significantly from one person to another.

To surpass the challenge of relatively limited amount of data of each patient, we employ transfer learning by first conducting pre-training on the collective data of all patients to acquire generic features of the pulse signals. Subsequently, we transfer this learned knowledge to each unique patient separately. We choose to leverage the same dataset for pre-training as for fine-tuning for the following reasons. First, the extracted pulse windows are 1-dimensional, and thus, pre-training should be performed on 1-dimensional data as well. In addition, trainable features that are suitable for one classification task, do not always perform well in other downstream tasks and especially in such niche applications. Variations are mainly attributed to the domain shift and the high discrepancy in the nature of the different tasks. In fact, several studies have indicated that the best performing encoder is the one trained on similar data [11, 12]. Over all, utilizing historical data of patients for predicting real-time outcomes would be the optimal solution in real-world scenarios.

For these reasons, our proposed methodology is task and dataset-specific and comprises the following two main steps. First, we perform supervised pre-training directly on all patients to explore the full spatial context of pulse signals and learn relevant global features that can transfer well to a similar medical setting. As a second step, we perform supervised fine-tuning on each individual patient separately to optimize even further its unique features and ultimately predict whether a pulse window is normal or abnormal.

3.1 Convolutional Neural Networks

A *convolutional neural network (CNN)* is a class of artificial neural networks (ANNs), most commonly used for computer vision-related tasks that involve the processing of pixel data. A CNN is typically composed of three types of layers or building blocks: *convolutional*, *pooling*, and *fully-connected* layers. The first two, convolutional and pooling layers, constitute the *encoder network* and perform feature extraction, whereas the third, a fully-connected layer, maps the extracted features (representations) to the desired task output.

Convolutional layer: Fundamental component of any CNN architecture that combines linear and non-linear operations, i.e., convolution operation and activation function.

- **Convolution operation:** Defines a kernel (or filter) that slides over an input image and takes the dot product between the filter and the parts of the image that fall inside.
- **Activation function:** Since the output of a convolution operation is a linear combination of features, a non-linear function is then required to add non-linearity to the network.

Pooling layer: A convolutional layer is in most cases followed by a pooling layer. This layer performs downsampling to reduce the dimensions of the convolved feature maps. In this way, the number of learnable parameters and the computational cost of the network can be reduced.

Fully-connected layer: The output feature maps of the final convolutional or pooling layer are flattened to one-dimensional array of numbers (or vector), and connected to one or more fully-connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. The final fully-connected layer (output layer) has the same number of nodes as the number of classes and maps the features to the desired output, e.g., probabilities for each class in classification tasks.

3.2 Pre-training pipeline

To learn global features of pulse signals, we adopt a 1D CNN and perform supervised pre-training directly on all patients. A 1D CNN is commonly well-suited for processing sequential data, such us time series signals including pulse signals. In this work, we investigate this type of neural network in a binary classification setting by optimizing the binary cross-entropy (BCE) loss, defined as:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1-y_i) \cdot \log(1-p(y_i)) \tag{1}$$

where y_i is the true label (1 for abnormal beats and 0 for normal beats) and $p(y_i)$ is the predicted probability of a certain pulse signal being abnormal for all N pulses.

Given a randomly sampled mini-batch of N 1D-signal windows, each 1D window is encoded via a base encoder network, a 1D CNN, to generate global features for the whole signal window. The features are then forwarded to a set of dense layers, i.e., a multi-layer perceptron (MLP), and mapped to a non-linear space, where the BCE loss is applied. The final goal is to generate optimal representations of the input signal by reducing the deviation of our predictions from the ground truth at the level of global features. Once the pre-training is complete, the base encoder is the model to be transferred, while the MLP is thrown away.

3.3 Fine-tuning pipeline

To perform the required downstream task on each individual patient from scratch, we keep only the base encoder from the pre-trained 1D CNN without the non-linear MLP. Then, given a single pulse window of a specific patient of size 1×128, we obtain 32×15 optimized feature maps using the weights of the pre-trained encoder. Each 1D feature map extracted from the encoder is flattened, resulting in a feature vector x with 160 features representing the input pulse. Another option would be to perform some kind of pooling operation, turning each feature map into a single value. This would result in a final feature vector of size 32 yet with a high loss of spatial information. Finally, given the feature vector x with the extracted features representing the input pulse signal, we want to learn one binary classifier that predicts whether this pulse is normal or abnormal. In total, we train 29 binary classifiers, one for each of the 29 unique patients separately.

4 Experimental setup

4.1 Tasks and Datasets

The resulting dataset from the pre-processing step is imbalanced in regard to the two classes. To this end, in order to split it into training, validation, and test sets we employ stratified splitting, preserving this way the ratio of the two classes among the three subsets. Specifically, we first split the dataset into training-validation and test sets with a 80-20 ratio and then, we further split the training-validation set into the training and validation sets, again with an 80-20 ratio. The random seed for the splitting operation is kept the same among all phases of the experiments in order to preserve the same examples in the sets across them. This means that the held-out test set is consistent across all experiments, since it should remain unseen by the network during the pre-training phase.

Another method we employed for dealing with the imbalanced dataset is weighted sampling, which allows for the less-represented class to be presented to the classifier in the same rate with the dominant one. This prevents the loss of every epoch to be dominated by the higher-frequency class and subsequently boosts the performance of the classifier on the other class.

4.2 CNN architecture

Our proposed CNN takes as input a 1×128 pulse window with frequency values in the range $[0,1]$. It consists of four convolutional blocks, each composed of a stack of two convolutional layers with 32 kernels of size 5×5, max pooling layer over a 5×5 kernel, and *Rectified Linear Unit (ReLU)* activation function. ReLU activation is defined as $z = \max(0,z)$ with z being the output of the second convolutional layer in each block. Our architecture also incorporates skip connections between layers via addition, inspired by the architecture of ResNet [13] for two-dimensional data. After the encoder part, the output is flattened to a 1×160 vector. On top of that, we add one dense layer of 160 units, followed by a single-unit output layer that conforms to the binary classification setting of our task. For the final layer, we use the *sigmoid* activation function that guarantees that the final output will always be between 0 and 1. As the sigmoid is a non-linear function, the output unit will be a non-linear function of the weighted sum of inputs and is derived by the following formula:

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}, \tag{2}$$

where x is the output of the final unit layer and exists in the range $(-\infty, +\infty)$. Considering that our dataset is low-dimensional and the task is binary classification, we decided to have a relatively simple architecture with a relatively small number (67169) of trainable parameters. The proposed architecture is illustrated in Figure 6.

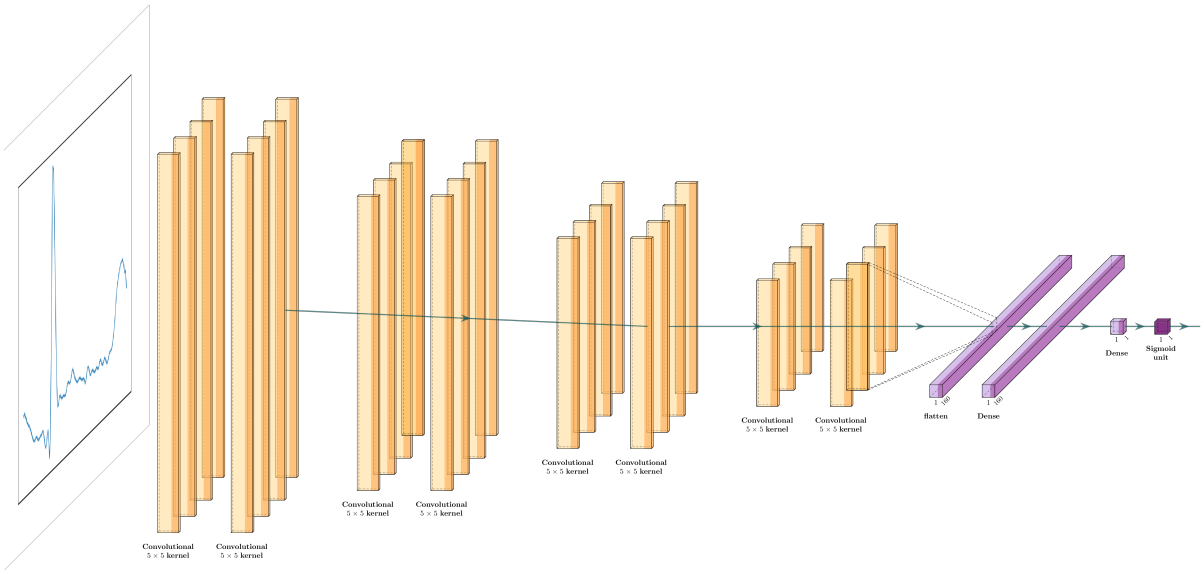

Figure 6: The architecture of our CNN.

4.3 Baseline method

For our baseline, we used classifiers that are based on the CNN architecture described in 4.2 and trained on the data of each patient individually. In order to decide on the optimal hyperparameters for each classifier, we performed a grid search over a range of values for learning rate and batch size chosen based on generally applied practices and our experiences, as well as toggling the option of weighted sampling. Those parameters along with their range are displayed in Table 1.

Parameter	Range
Learning rate	$\{0.001, 0.01, 0.1\}$
Batch size	$\{4, 16, 32\}$
Weighted sampling	$\{True, False\}$

Table 1: Parameters and their corresponding range for grid search of individual classifiers.

Our primary metric for the classification task is balanced accuracy as it can deal with imbalanced data in a classification setting. The balanced accuracy is defined as follows:

$$\text{Balanced accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2} \tag{3}$$

where

$$\text{sensitivity or recall} = \frac{TP}{TP+FN} \qquad \& \qquad \text{specificity} = \frac{TN}{TN+FP} \tag{4}$$

- TP – true positive (the correctly predicted positive class outcome of the model)

- TN – true negative (the correctly predicted negative class outcome of the model)
- FP – false positive (the incorrectly predicted positive class outcome of the model)
- FN – false negative (the incorrectly predicted negative class outcome of the model)

4.4 Pre-training protocol

For the pre-training phase, we investigate the 1D CNN as our base encoder network for learning optimal global features that are representative of the pulses windows. An MLP follows that maps the output of the encoder to a 160-dimensional embedding, which is used for binary classification. We perform supervised pre-training on the entire dataset of all patients trying to minimize the BCE loss.

We adopt Adam as the optimizer and set its weight decay and learning rate to 0.0001 and 0.001, respectively. The pre-training model is trained with a batch size of 32 and optimized with a reduce learning rate schedule that adjusts the learning rate during the training process. Using this protocol, the model is pre-trained on all patients for up to 30 epochs, which is a total of approximately $30K$ iterations. In the end, we keep the model at the epoch with the lowest validation loss for later use. All pre-training experiments are implemented in Python programming language using the PyTorch framework under the Google Colab environment. Our code supports both Central Processing Unit (CPU) and Graphics Processing Unit (GPU) programming.

4.5 Fine-tuning protocol

We train the 1D CNN model during the downstream task evaluation using the weights of the best performing pre-trained network as initialization. The end-to-end network for the fine-tuning phase is the one described in Section 4.2. At this stage, we use the same experimental setup for each patient as in the baseline method detailed in Section 4.3. Finally, we evaluate the fine-tuned model on the test set for each individual patient to report the final task performance on completely unseen data. Our primary metric for the classification task is balanced accuracy as it can deal with imbalanced data in a classification setting.

5 Experiments and Results

In this section, we evaluate the performance of transfer representation learning in binary classification. To this end, first, we explore the choice of the (pre-)training strategy for extracting features from 1D pulse windows and performing binary classification. Next, we compare the performance of transfer learning against the supervised training baseline. For each system, we provide the same performance metrics for fair comparisons.

5.1 Pre-training

During the pre-training of the network, we use the loss on the validation set as the criterion for model selection. In Figure 7 we can see the loss curves for the training and validation sets. The red vertical line denotes the epoch (16) where the validation loss is minimum.

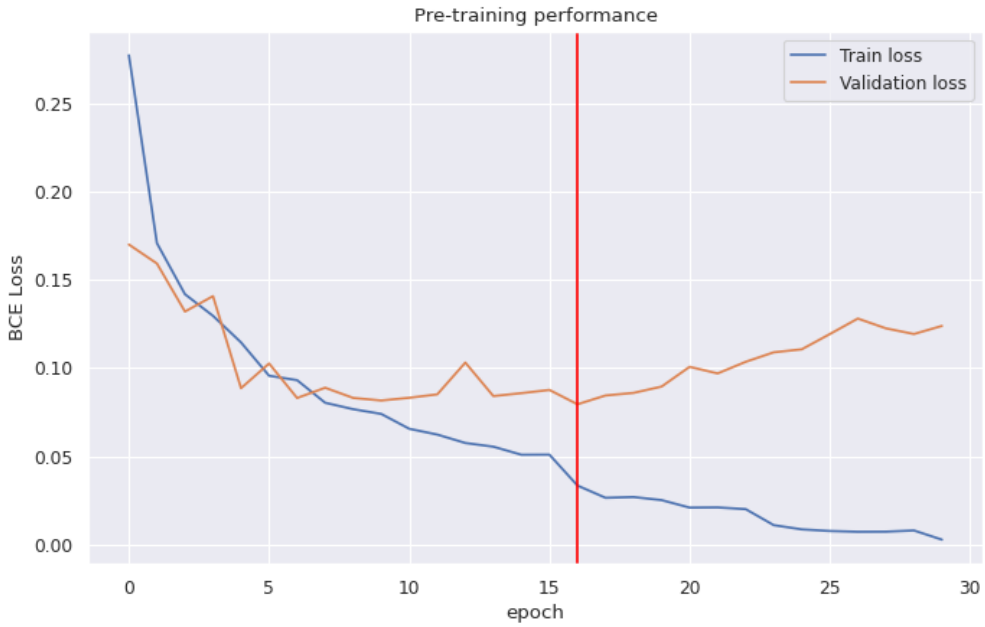

Figure 7: Training and validation loss curves over epochs for the pre-training phase.

5.2 Individual patients

To be able to judge the usefulness of the application of transfer learning in this context, we first present an evaluation of the performance of the classifiers that were trained on the data of a specific patient only as a baseline. For this evaluation, we primarily use the metric of balanced accuracy that was achieved on the test set.

The setups used for each classifier can be seen in 2 in the appendix. In Figure 8, the performance of all 29 individual classifiers is reported, collected into bins for a clearer overview. The average balanced accuracy on the test set for the individual classifiers is 0.946.

We randomly chose patient 203 to present the indicative progression of the training of these classifiers. Figure 9 shows the training and validation loss curves (left) and the evolution of the balanced accuracy over the epochs (right).

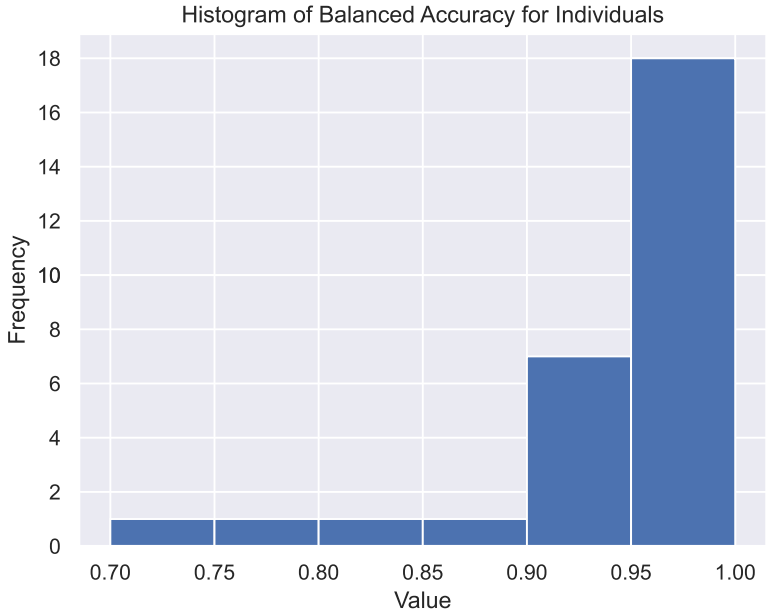

Figure 8: Histogram of the balanced accuracy of the individual classifiers on the test set.

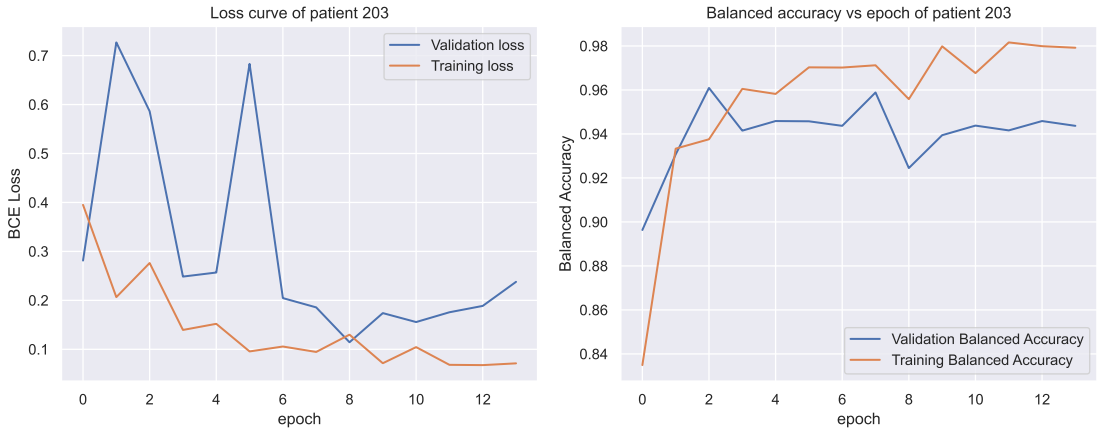

Figure 9: Loss curve (left) and balanced accuracy (right) plots over epochs for patient 203 for the individual classifier.

5.3 Transfer learning

We perform the same evaluation for the classifiers that we obtained with the method of transfer learning. In Figure 10 we present the balanced accuracy achieved on the test set for each derived classifier, again grouped into bins for visibility. The average balanced accuracy on the test set for the fine-tuned classifiers is 0.935. Figure 11 shows how the loss (left) and balanced accuracy (right) change over the epochs for the indicative patient 203.

Figure 12 makes the comparison between the achieved balanced accuracies on the test set of the classifiers trained on the specific patients and classifiers derived from the fine-tuning of the network that was pre-trained on the whole dataset.

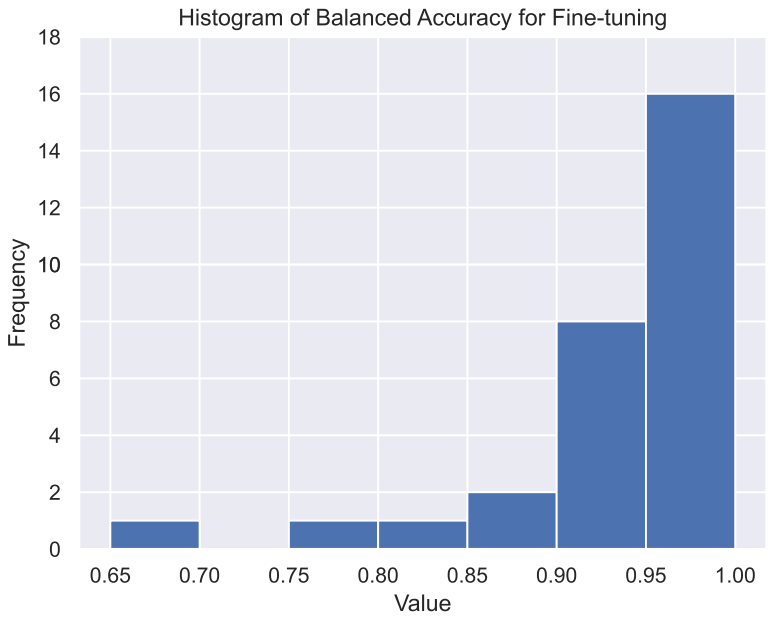

Figure 10: Histogram of the balanced accuracy of the fine-tuned classifiers on the test set.

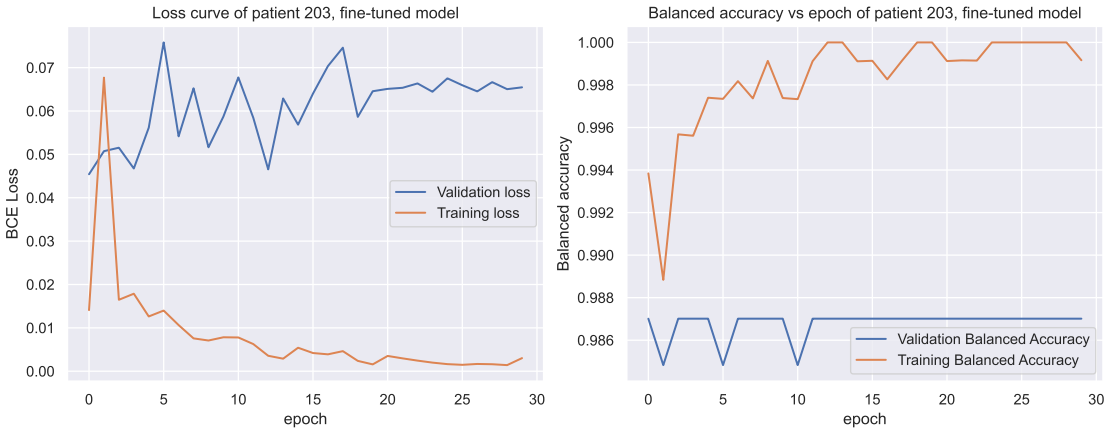

Figure 11: Loss curve (left) and balanced accuracy (right) plots over epochs for patient 203 for the fine-tuned classifier.

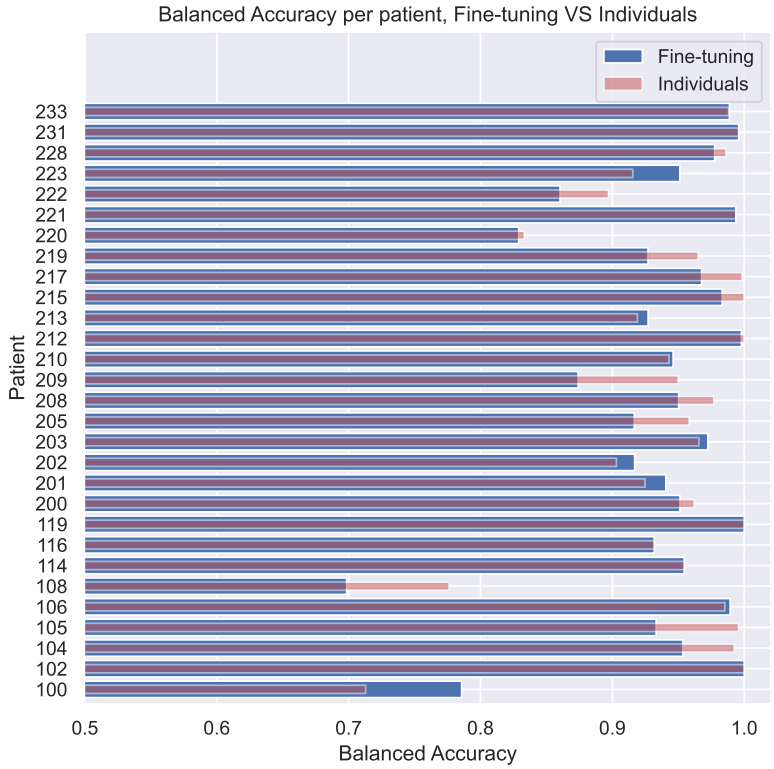

Figure 12: Bar plot of the balanced accuracy for each patient on the test set (individuals, fine-tuning)

6 Discussion

As shown in Table 2, the optimal hyperparameters vary for each patient. For example, we can see that for patient 104, the optimal batch size is 4, while for patient 200 the optimal batch size is 32. This shows the importance of the grid search using the validation set, and that it should be done separately for each patient.

Figure 7 shows the progression of loss function during pre-training. The training loss shows a steady and persistent decrease, following a monotonic trend, until the end of the training process. This indicates that the neural network is effectively learning from the training data and improving its performance over time, without strong fluctuations or plateaus in the loss reduction. However, the validation error starts increasing after epoch 16. For this reason, we choose the model at epoch 16 that has the lowest validation error.

Figure 9 (left) shows the progression of the loss function over the epochs, during the training of the individual classifier for patient 203. We can clearly see that the training loss is converging. Regarding the validation loss, it is fluctuating considerably during the first 5 epochs, reaching its lowest at 8 epochs. Figure 9 (right) shows the progression of the balanced accuracy over the epochs, during the training of the individual classifier for patient 203. We note that as the training process goes on, the balanced accuracy on the training set increases, but on the validation set it converges around the 3rd epoch, with some fluctuations.

Figure 11 (left) shows how the loss function evolves over the epochs during the fine-tuning for patient 203. The training loss converges fast. The validation loss displays some variations, and then converges. Figure 11 (right) shows the balanced accuracy for each epoch during the fine-tuning for patient 203. We can see that the training accuracy is high, as well as the validation accuracy.

Figure 8 shows the histogram of the balanced accuracy obtained from the individual classifiers on the

test set. We can see that the majority of the classifiers (18 out of 29) have achieved a balanced accuracy higher than 0.95, and 7 out of 29 were between $[0.90, 0.95]$. There is one patient with a relatively low balanced accuracy between $[0.70, 0.75]$, while for the other 3 patients, we achieved balanced accuracy between $[0.75, 0.90]$. In general, for most patients, the individual classifiers can achieve high performance. The corresponding histogram for the fine-tuning is shown in Figure 10. We can see that, fine-tuning is able to yield high performance for most of the patients (16 out of 29 between $[0.95, 1.0]$, and 8 out of 29 between $[0.90, 0.95]$. However, for some patients the performance is not that good (one patient $[0.65, 0.70]$, and 4 between $[0.75, 0.90]$).

Figure 12 shows a horizontal plot with the balanced accuracy on the test set for all patients, both for fine-tuning and individual classifiers. What is important to see here, is that for some patients the individual classifiers have better results (e.g., 222, 219), while for some others, fine-tuning was able to boost the performance (e.g., 100, 202).

When comparing the loss and balanced accuracy on the training set, the correlation between them is clear. This is an indicator that the loss function we use (BCE loss) was a proper choice. In general, we can say that while both methods achieve great results, the individual classifiers seem to get a slight edge over the fine-tuning. More specifically, the individual classifiers slightly surpasses the fine-tuned ones by a margin of 0.01 on average. This can be partially explained by the experimental setup. For the individual classifiers, we perform a grid search to find the optimal hyperparameters (e.g., batch size, learning rate), and for the second phase of fine-tuning we use those hyperparameters found for the individuals. Perhaps another grid search during the fine-tuning, or using the hyperparameters used in pre-training, would help the transfer learning achieve better performance.

7 Conclusion

In this work, we have trained various CNNs for the purpose of classifying an ECG signal into normal or abnormal. We use the MIT-BIH arrhythmia database which contains raw ECG signals, annotated by medical experts. After applying some pre-processing steps to clear the raw data, we split the resulting signal into windows of fixed length. Next, we annotate each window (0 for normal/1 for abnormal) according to the annotation provided by the medical expert. Then, we use those windows as input patterns into a CNN, and as targets, we use the 0/1 label set.

Our goal was to create a system that could classify the input patterns as normal or abnormal. To achieve this, we use individual classifiers as a baseline, and transfer learning. For the individual classifiers, we train a separate CNN for each patient, using only data from that patient. We also perform a grid search to find the optimal hyperparameters for each patient, using a dedicated validation set. For transfer learning, we use data from all the patients to train a generic network, in hope of learning the important features. Then we use that generic network, and fine-tune it on each patient.

Our results show that the individual classifiers are performing quite well already, and the transfer learning while it increases the performance for some cases, on average it achieves marginally worse results.

7.1 Limitations and Future Work

The biggest limitation of our work is that we use data of a patient to train a system for that same patient. In a practical application, it means that in order to create such system that classifies a patient's ECG, you would first need to collect and annotate data from him/her. The holy grail would be to train a system based on some patients, and then use it to make predictions on a new patient. However, this proves to be too difficult most of the times.

As a future work, there are some things we could further explore and expand on. Firstly, we can experiment with other setups, such as more layers, different loss functions, dropout between layers, and learning rates. Secondly, trying other options in the pre-processing stage, for example, data augmentation (e.g., through added noise), or a moving average filter. Moreover, we could experiment with different window sizes, or segment the original signal based on some beat detection. This means that instead of splitting the signal into consecutive windows of fixed length, we could use as input a whole beat. While this looks promising to increase the accuracy of the prediction, it strongly relies on the ability to

confidently detect beats. Lastly, we can explore the effects of combining the dataset we use with other datasets of the same nature to further increase the available data.

References

[1] J. Schläpfer and H. J. Wellens, "Computer-interpreted electrocardiograms: Benefits and limitations," *Journal of the American College of Cardiology*, vol. 70, no. 9, pp. 1183–1192, 2017.

[2] S. R. Steinhubl, J. Waalen, A. M. Edwards, L. M. Ariniello, R. R. Mehta, G. S. Ebner, C. Carter, K. Baca-Motes, E. Felicione, T. Sarich, and E. J. Topol, "Effect of a Home-Based Wearable Continuous ECG Monitoring Patch on Detection of Undiagnosed Atrial Fibrillation: The mSToPS Randomized Clinical Trial," *JAMA*, vol. 320, pp. 146–155, 07 2018.

[3] J.-L. Ma and M.-C. Dong, "R&d of versatile distributed e-home healthcare system for cardiovascular disease monitoring and diagnosis," in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 444–447, 2014.

[4] K. Minami, H. Nakajima, and T. Toyoshima, "Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network," *IEEE Transactions on Biomedical Engineering*, vol. 46, no. 2, pp. 179–185, 1999.

[5] O. T. Inan, L. Giovangrandi, and G. T. A. Kovacs, "Robust neural-network-based classification of premature ventricular contractions using wavelet transform and timing interval features," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 12, pp. 2507–2515, 2006.

[6] D. Coast, R. Stern, G. Cano, and S. Briller, "An approach to cardiac arrhythmia analysis using hidden markov models," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 9, pp. 826–836, 1990.

[7] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A review on deep learning methods for ecg arrhythmia classification," *Expert Systems with Applications: X*, vol. 7, p. 100033, 2020.

[8] F. Hadaeghi, "Reservoir computing models for patient-adaptable ecg monitoring in wearable devices," 2019.

[9] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[10] V. Mondejar, "Mit-bih database," Apr 2018.

[11] S. Azizi, B. Mustafa, F. Ryan, Z. Beaver, J. Freyberg, J. Deaton, A. Loh, A. Karthikesalingam, S. Kornblith, T. Chen, *et al.*, "Big self-supervised models advance medical image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3478–3488, 2021.

[12] K. Kotar, G. Ilharco, L. Schmidt, K. Ehsani, and R. Mottaghi, "Contrasting contrastive self-supervised representation learning pipelines," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9949–9959, 2021.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Appendices

A Optimal settings for each patient

patient	batch size	learning rate	weighted sampling	balanced accuracy
100	4	0.001	True	0.713
102	4	0.001	True	1.00
104	4	0.001	False	0.993
105	4	0.001	False	0.996
106	16	0.001	True	0.985
108	32	0.01	True	0.776
114	4	0.001	False	0.955
116	16	0.001	True	0.932
119	4	0.01	True	1.00
200	32	0.01	True	0.962
202	16	0.001	True	0.903
201	32	0.001	True	0.925
203	4	0.001	True	0.966
205	32	0.001	True	0.958
208	4	0.01	False	0.977
209	4	0.001	False	0.950
212	16	0.001	True	1.00
210	16	0.001	True	0.943
213	4	0.001	True	0.919
215	4	0.001	True	100
217	4	0.001	True	0.998
219	16	0.01	True	0.965
220	16	0.01	True	0.955
222	32	0.001	False	0.897
221	32	0.001	False	0.994
228	16	0.001	True	0.986
223	16	0.001	True	0.916
231	4	0.001	True	0.996
233	32	0.001	False	0.988

Table 2: This Table provides the optimal hyperparameters for each patient, found through grid search based on the balanced accuracy on the validation set.

B Beat Annotations

Code	Description
N	Normal beat (displayed as "·" by the PhysioBank ATM, LightWAVE, pschart, and psfd)
L	Left bundle branch block beat
R	Right bundle branch block beat
B	Bundle branch block beat (unspecified)
A	Atrial premature beat
a	Aberrated atrial premature beat
J	Nodal (junctional) premature beat
S	Supraventricular premature or ectopic beat (atrial or nodal)
V	Premature ventricular contraction
r	R-on-T premature ventricular contraction
F	Fusion of ventricular and normal beat
e	Atrial escape beat
j	Nodal (junctional) escape beat
n	Supraventricular escape beat (atrial or nodal)
E	Ventricular escape beat
/	Paced beat
f	Fusion of paced and normal beat
Q / ?	Unclassifiable beat
[	Start of ventricular flutter/fibrillation
!	Ventricular flutter wave
]	End of ventricular flutter/fibrillation
x	Non-conducted P-wave (blocked APC)
(	Waveform onset
)	Waveform end
p	Peak of P-wave
t	Peak of T-wave
u	Peak of U-wave
‘	PQ junction
’	J-point
^	(Non-captured) pacemaker artifact
|	Isolated QRS-like artifact
∼	Change in signal quality
+	Rhythm change
s	ST segment change
T	T-wave change
*	Systole
D	Diastole
=	Measurement annotation

Table 3: All beat annotations and their explanations