ML4HC Project 1: ECG time series

Rhea Sukthanker
srhea@student.ethz.ch
18-949-628

Rishabh Singh
risingh@student.ethz.ch
19-953-793

Rahul Bera
rahbera@ethz.ch
19-900-687

1 INTRODUCTION

Electrocardiogram (ECG) is widely used by the doctors and medical practitioners to monitor cardiac health. An irregular heartbeat might be proven fatal in many cases [1], which can be avoided by accurately identifying arrhythmic heartbeat signals in ECG. However, detecting irregularities and classifying different waveforms in a long time-series ECG data is not only tedious and time-consuming for any practitioner, but also subjected to individual human bias. A machine-learning model, trained using large ECG dataset, can alleviate the shortcomings of a manual analysis and can provide higher accuracy in detecting arrhythmic heartbeats.

2 SAMPLES VISUALIZATION AND CLUSTERING

2.1 Sample Visualization

Figure 1 and 2 shows the visualized samples from the PTB Disgnostic dataset [3] and MIT-BIH Arrhythmia dataset [2] respectively. Our key observation is that some of the dataset classes can be distinguished clearly by visual inspection using timeseries properties like amplitude, while other classes are overlapped and harder to be distinguished. For example, classes 0 and 4 from MIT-BIH Arrythmia dataset can be distinguished in Figure 2, while classes 0 and 1 are not.

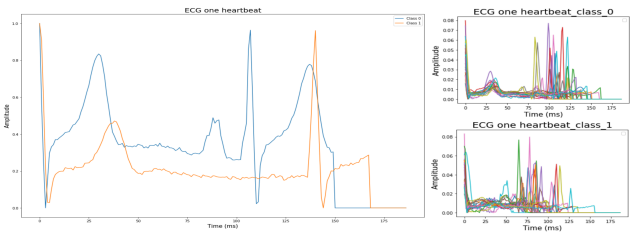

Figure 1: Visualization of class-wise time series samples of PTB Diagonostic dataset

2.2 Visualization in 2D using PCA

The figure above depicts the 2-D representation of the time series (obtained using Principal Component Analysis). We observe that in the case of the MIT-BH dataset the five classes are clearly distinguishable. On the other hand the two classes in the case of the PTB dataset are quite hard to distinguish.

2.3 Visualisation using k-means clustering

The figure above depicts the results using k-means clustering with 5 and 2 clusters for the MIT-BH and the PTB datasets respectively. We observe that the clusters are well segregated in the 2-D space.

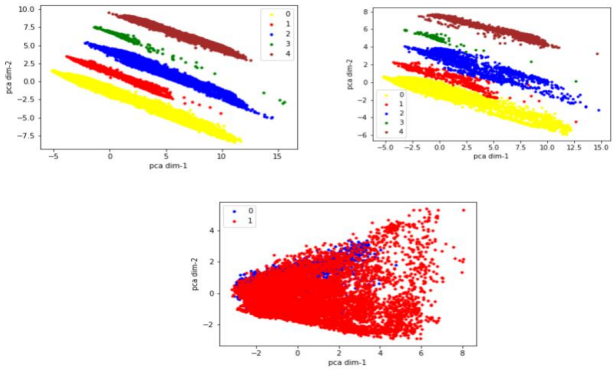

Figure 3: PCA on MIT-BIH Arrhythmia dataset (train and test) and the PTB dataset

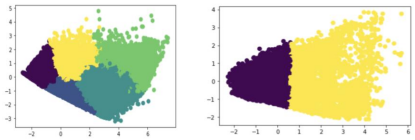

Figure 4: Visualization of k-means clustering on MIT dataset (left)and the PTB dataset (right)

3 BASELINES

We use the model proposed by Kachuee et al. [4](without residual connections) as the baseline to classify the ECG signals taken from two datasets: the MIT-BIH Arrhythmia database [2] and the PTB Diagnostic ECG database [3]. The architecture of the model is illustrated in Figure 5. The model is based on 1D convolutions but without residual blocks.

For the MIT-BIH Arrhythmia dataset, the training of the model ended early in 32 epochs, achieving an F1 score of 91.2178% and accuracy of 98.5383%. For the PTB dataset, the training ended in only 25 epochs, achieving an F1 score of 98.6698% and an accuracy of 98.0763%.

4 NEURAL NETWORK ARCHITECTURES IMPLEMENTED

We implemented five different architectures for the task (a CNN with residual connections, a bidirectional LSTM, a RNN, A GRU). We describe the architectures in detail in the Appendices.

5 ENSEMBLE MODELS

We follow two approaches to realize an ensemble of methods mentioned in Table 1. The first approach is based on majority voting,

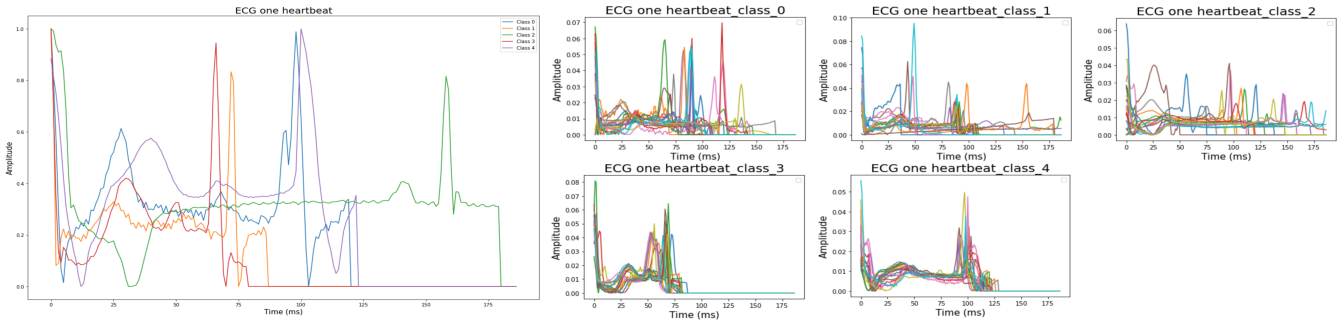

Figure 2: Visualization of class-wise time series samples from MIT-BIH Arrhythmia dataset

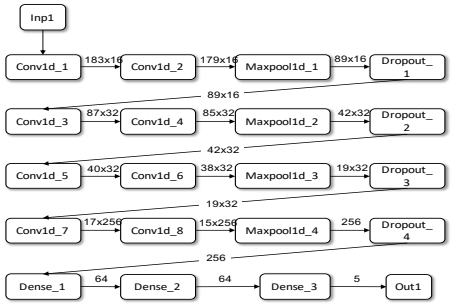

Figure 5: The architecture of the baseline network

where every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes. In the second approach, we first obtain a d-dimensional vector of logits from each model. We then calculate the average value of these logits across the models. We then train a logistic regression model on this feature vector to predict the test labels.

6 TRANSFER LEARNING

We investigated the effect of transfer learning between the two datasets by training on one and testing on another. To do so, we first train our LSTM model on MIT dataset. We then remove the output layer from the trained model using model.pop() functionality provided by keras. We then add a new output layer and train the whole model with the PTB dataset. We finally tested the performance on the test set of PTB dataset. We obtained an accuracy of **72.2%** and F1-score of **42.1%**. We evaluated the same method on GRU based models and got an an accuracy of **92.3%** and F1-score of **94.2%**.

7 DISCUSSION AND RESULTS

We worked on two datasets for ECG time series classification in this project. The MIT-BIH is a multi-class dataset where PCA shows that individual classes are easily separable. This is concurred by the observation that all the classification methods in table 1 achieved an accuracy of more than 95% for prediction. However, PTB dataset has comparatively lesser inter-class separability,i.e. the classes highly overlap with each other as shown in 3. As such, classification was more challenging and the performance of various models varied widely. However, the ensembling with logistic regression performed the best. We tried CNNs, RNNs, ensemble and transfer learning as different approaches for classification. CNN+Residual was consistently better across the datasets, both in accuracy and F1-score. Among the recurrent networks, GRU was consistently better on both datasets. Both ensemble (in case of MITBH) and transfer learning didn't provide significant boost-up.

Table 1: Accuracy and F1-score of different models in MIT-BIH Arrhythmia and PTB dataset

Models	MITBH		PTB	
	Accuracy	**F1**	**Accuracy**	**F1**
Baseline	98.54%	91.21%	98.08%	98.67%
CNN+Residual	**98.73%**	**92.61%**	**99.48%**	**99.36%**
CNN+Residual+Transfer	98.69%	92.30%	99.24%	99.06
Simple RNN	82.76%	18.11 %	72.11%	42.25%
LSTM	97.09%	84.67%	71.62%	61.50%
GRU	97.99%	88.94%	93.20%	91.03%
Bi-dir LSTM	98.41%	91.46%	88.05%	85.19%
Ensemble (majority voting)	95.73%	78.53%	88.11%	83.28%
Ensemble (logistic reg.)	94.13%	60.13%	**99.48%**	**99.35%**

Table 2: AUROC & AUPRC of models in PTB dataset

Models	AUROC	AUPRC
Simple RNN	0.59	0.57
LSTM	0.76	0.70
Bi-dir LSTM	0.90	0.87
GRU	0.84	0.82
CNN + Residual	**0.99**	**0.99**

REFERENCES

[1] "Heart Diseases Disorders," https://www.hrsonline.org/patient-resources/heart-diseases-disorders.
[2] "MIT-BIH Arrhythmia Database," https://www.physionet.org/content/mitdb/1.0.0/.
[3] "PTB Diagnostic ECG Database," https://www.physionet.org/content/ptbdb/1.0.0/.
[4] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2018, pp. 443–444.

Appendices

Appendix A

A MODELS

A.1 CNN with Residual Connections

The baselines provided do not use residual blocks in the CNN architecture as proposed by Kachuee et al. [4]. We observe that adding residual blocks improves the performance on both the datasets. Residual connections basically are skip connections from the input as illustrated in the 6. The motivation behind the use of residual connections is to allow a direct(skip) connection from the input to the (intermediate) output.

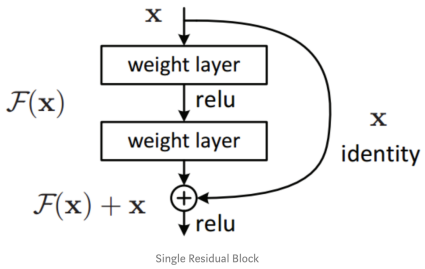

Figure 6: A single residual block

A.2 RNN Based Models (Simple RNN, LSTM and GRU

A.2.1 ***RNN.*** Time series are essentially sequences (like words in a natural language), thus it is reasonable to use sequence based models (eg: hidden markov models) to process time series. The Deep Learning based generalization of hidden markov models is an RNN. Precisely an RNN is a sequence model in which every state depends on the previous state and the current input x (depicted in the figure 7). The following equations constitute an RNN. However an RNN has some disadvantages. Firstly it suffers greatly from the "vanishing gradient problem" and secondly it tends to "forget" information it has seen too long ago. GRUs and LSTMs were proposed to handle precisely this task. In the figure below Xt denotes the observation(ecg measurement) at time t and Yt denotes the prediction for a particular time step t. Thus given a history of time steps the RNN gives a prediction of the measurement at the next time step.

$$h_t = f(Wh_{t-1} + Vx_t + b), \text{ where } f \in \{\text{sigm}, \tanh\}$$

In the equation above x_t is input at time step t and h_t is hidden state at time t

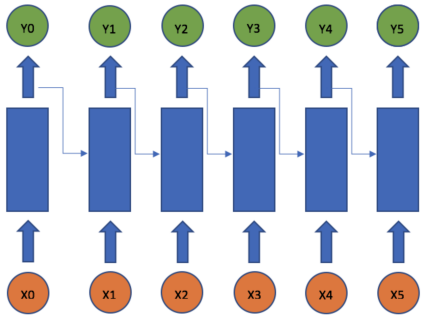

Figure 7: Unrolled RNN

A.2.2 ***Long Short-Term Memory: LSTM.*** An LSTM is a variant of the RNN which has capabilities to retain relevant information for longer periods of time (refer figure 8. It used the "input" and the "forget " gate to selectively allow the memory of the past to propagate to the further output (using the output gate). Though LSTMs suffer from the vanishing gradient problem too they are influenced by it to a smaller extent. The equations of the RNN/LSTM is as depicted by the figure below.

$$\begin{aligned} f_t &= \sigma(W_f * [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i * [h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o * [h_{t-1}, x_t] + b_o) \\ \tilde{C}_t &= tanh(W_c * [h_{t-1}, x_t] + b_c) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ h_t &= o_t * tanh(C_t) \end{aligned}$$

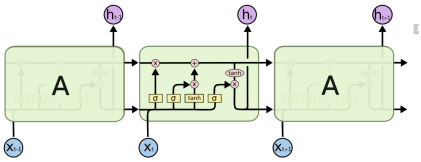

Figure 8: LSTM block

A.2.3 ***Bi-LSTM.*** The bi-directional LSTM is a modification on top of the general LSTM architecture. Generally the LSTM is unidirectional i.e we proceed in a certain direction (usually left to right). A bidirectional-LSTM allows propagation in both forward and reverse directions. The architecture of the Bi-LSTM is as depicted in the figure 9:

Models	Inp. layer size	Rec. layer hidden state size	Dropout rate	Number of layers	Optimizer	Learning Rate	Recurrent gate
Simple RNN	(187,1)	64,128 (2 RNN layers)	RNN layer 0.2	2 RNN 3 dense total 5	Adam	0.001	RNN hidden units
LSTM	(187,1)	32,64,128 (3 LSTM layers)	LSTM layer 0.2	3 RNN 3 dense total 6	Adam	0.001	LSTM cell
Bi-dir LSTM	(187,1)	64,128 (2 LSTM layers)	LSTM layer 0.2	2 RNN 4 dense total 6	Adam	0.001	LSTM cell
GRU	(187,1)	64,128 (2 GRU layers)	GRUlayer 0.2	2 GRU 3 dense total 5	Adam	0.001	GRU cell
CNN + Residual	(187,1)	-	GRUlayer 0.2	1 conv1d+5 residual blocks+3 dense =9 layers	Adam	0.001	-

Table 3: Best hyperparameters for the validation set

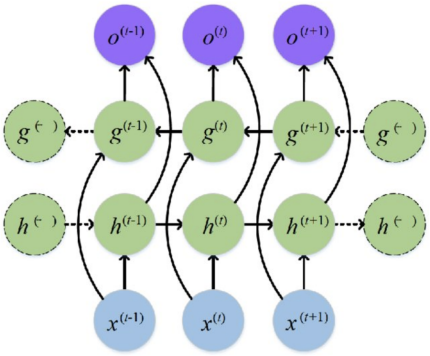

Figure 9: Bi-RNN

A.2.4 ***Gated Recurrent Unit: GRU.*** GRU is another variant of RNN which like LSTM aims at retaining relevant information longer with an additional advantage of using fewer parameters than the LSTM. Precisely it replaces the weighted combination of the input and the forget gate by a convex combination of a single gate (hence reducing the number of parameters to be learnt). A GRU is depicted by the equations below.

$$z_t = \sigma(W_z * [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r * [h_{t-1}, x_t])$$
$$\tilde{h_t} = tanh(W * [r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t$$

A.3 Hyper-parameters

We refer the reader to table 3 for best hyperparameters obtained on the validation set for each of the models used in the above study.