CS236781 – Deep Learning on Computational Accelerators

Detection of Atrial Fibrillation using Deep Learning models

Michael Mendelson-Mints [1]

Dan Kalifa [2]

September, 2020

[1] michael.me@campus.technion.ac.il, ID - 207193210
[2] kalifadan@campus.technion.ac.il, ID - 206063497

1 Abstract

Atrial Fibrillation (AF) is the most common dysrhythmia encountered in the United States. Symptoms may be similar to those of other cardiac conditions, which can delay the timely detection, diagnosis, and management of AF [Hickey and Riga, 2019]. This study provides a well-motivated deep learning architecture for detection of AF (Atrial Fibrillation) from ECG signals.

Our work is based on the paper "Detection of Paroxysmal Atrial Fibrillation using Attention-based Bidirectional Recurrent Neural Networks"[Shashikumar et al., 2018]. The proposed method in the article is a combination of time-frequency representation (Wavelet Transform), translation-invariance of CNN, temporal features (BRNN), and a soft attention mechanism that is capable of detecting important features independent of their actual position within a sequence of windows.

We implement a baseline model which draws inspiration from the paper, and show that it performs well even on a small benchmark data set (the MIT-BIH AF data set). It was shown that TCNs (Temporal Convolutional Networks) are able to learn inherent patterns in sequential data automatically and preform better on similar tasks as our task. Thus, we replaced the BRNN layer with a Temporal Convolutional Network (TCN).

We show that for the task of AF detection from ECG, a TCN-based model substantially outperforms the baseline model, by presenting a more modern and robust architectural approach to ECG-based disrhythmia detection. The TCN-based model achieves an accuracy of 0.88 compared to the baseline's 0.82, and an AUC of 0.87 compared to the baseline's 0.83. Future work might test transferability of the new model to other types of heartbeat signals (such as PPG) and to set-level detection of different rhythm types.

Full code for this project and instructions on how to run the experiments are available on GitHub [3].

[3] https://github.com/kalifadan/Final_project_cs236781

2 Introduction

Atrial Fibrillation (AF or A-fib) is an abnormal heart rhythm (disrhythmia) characterized by the rapid and irregular beating of the atrial chambers of the heart (prevalence of 2% in the adult population). It often begins as short periods of abnormal beating, which become longer or continuous over time [Zoni-Berisso et al., 2014], and is usually diagnosed by the absence of P-waves in the ECG reading. Paroxysmal AF (PAF) is a form of AF that occurs occasionally, and has a higher probability of being undetected.
Our work is based on the paper "Detection of Paroxysmal Atrial Fibrillation using Attention-based Bidirectional Recurrent Neural Networks"[Shashikumar et al., 2018]. The paper [Shashikumar et al., 2018] discusses two main concepts:

- Detection of Paroxysmal AF (PAF) using a deep learning architecture with an attention mechanism [Bahdanau et al., 2014].
- Transferring knowledge from the source domain (ECG readings), to detect AF from pulsatile photoplethysmogram (PPG) readings (the target domain).

The paper shows that the proposed model bypasses baseline models with an AUC of 0.94 on the testing set, and is able to detect AF with a low false-alarm rate. It also discusses the potential of domain transfer to PPG readings obtained from wearable devices to act as a long-term monitoring system which can detect AF in real time.

2.1 Paper's data

The datasets used in the article consists of 2 datasets:

- Holter ECG dataset.
- Smart Watch PPG dataset.

The first dataset consists of 24 hour Holter ECG recordings collected from 2850 patients after been confirmed by a clinical adjudicator and divided into 10-minute segments (after excluding less than 2% segments with a low signal quality index - SQI).
The other dataset consists of pulsatile photoplethysmogram (PPG) recordings from 97 subjects, taken from smartwatches - 44 positive cases and 53 negatives (with other rhythms) for approximately 5-10 minutes.

2.2 Paper's methods

Figure 1 presents an overview of the paper's proposed deep learning architecture for the detection of Paroxysmal Atrial Fibrillation (PAF).

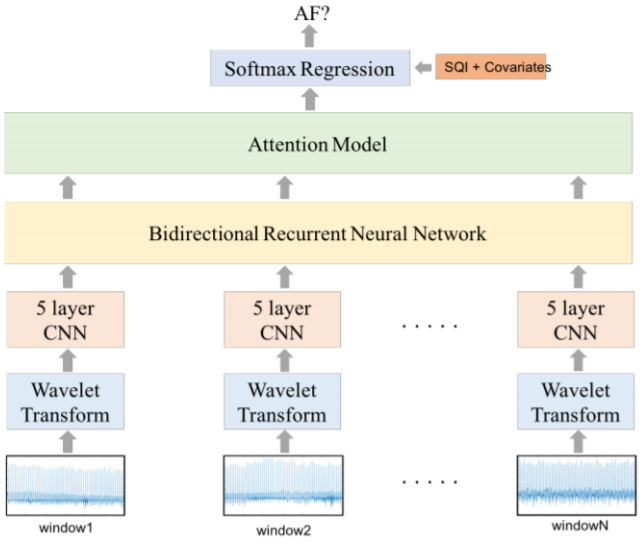

Figure 1: Schematic diagram of the AF detection algorithm

The deep learning architecture that shown includes five main parts:

- WAVELET DECOMPOSITION – The wavelet transform is capable of extracting time-frequency information from signals of a non-stationary nature, due to its optimal time-frequency resolution trade-offs [Daubechies, 1990]. The main purpose of applying it is for extraction features from the frequency domain. In the paper's algorithm, the 10-minute segment was split into non-overlapping 30-second windows, and the wavelet transform [Shashikumar et al., 2017] was applied to each of the windows, resulting in a wavelet spectrogram matrix of size 20 by 300.

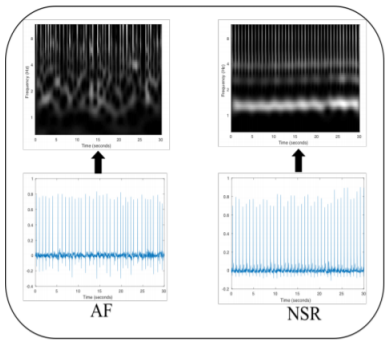

Figure 2: Wavelet power spectrum for 30 second ECG sample. The left panel shows an example of AF segment and the right panel represents a Normal Sinus Rhythm segment

- Feature extraction – A 5-layer CNN receives as input an image of a wavelet spectrogram (for a 30 second window of an ECG reading), which is then fed into 2 successive convolutional layers, a max pooling layer, 2 more convolutional layers and a fully-connected layer. CNNs were shown to achieve good results in extracting important features from images which benefit from invariance of translation and locality of visual features.

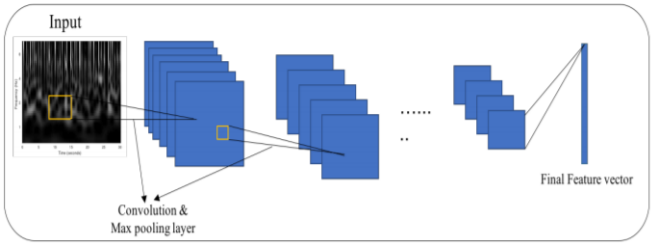

Figure 3: Schematic diagram of the deep CNN layer

- Temporality – The result of all parallel CNN layers (each receiving the next window in a 10-minute ECG reading) is fed into a Bidirectional Recurrent Neural Network, to capture temporal features, and leverage the information flow in both directions of time in the reading.

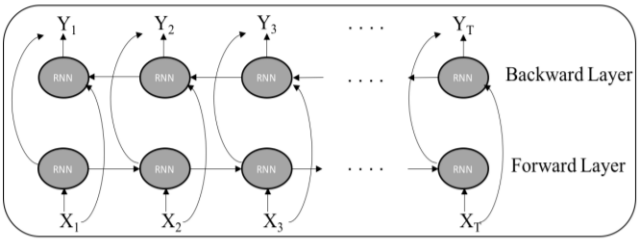

Figure 4: Schematic diagram of the bidirectional recurrent neural network

- Attention – The output of the BRNN is then fed into an attention layer, to capture the most important windows of the input (relying on their temporal and visual features as an indication to the presence of AF).
- Classification – Finally, the attended features are fed into a fully-connected layer and a decision is made based on a Softmax regression, which is also given as input external features for the ECG reading (time series covariates), which are made to capture the quality of the reading (beat-to-beat sample entropy and autocorrelation of the signal).

2.3 Paper's results

The data samples were divided into groups with a different AF burden. an AF burden is the percentage of time from a 10-minute segment spent in AF. It was shown that for an AF burden of $> 5\%$, the model was able to achieve an AUC of 0.94. The model was then compared to a baseline model, and two more models, one without covariates information, and another algorithm proposed by Carrara et al. [Carrara et al., 2015]. The results of the model performances are detailed in Figure 5.

	Testing set				Training set			
% AF burden	*AUC*	AUC_{pr}	*SPC*	ACC	*AUC*	AUC_{pr}	*SPC*	ACC
>5	0.94	0.84	0.95	0.94	0.96	0.93	0.96	0.96
>25	0.93	0.82	0.92	0.92	0.96	0.93	0.95	0.95
>50	0.95	0.82	0.93	0.95	0.97	0.91	0.95	0.94
>75	0.97	0.84	0.96	0.96	0.98	0.96	0.98	0.96

Figure 5: Summary of classifier performance for different % of AF burden. The Area Under the Curve (AUC), Area under the precision recall curve (AUCpr), Specificity (SPC) and Accuracy (ACC) are reported for both training set and testing set.

The model outperformed all others when compared with the AUC, the Area Under the Precision-Recall curve (AUC_{pr}), the specificity (SPC) and accuracy (ACC).
The model was then used to transfer knowledge from the domain of ECG reading, to a new target domain of PPG readings taken from smart watches. The pre-trained model outperformed a baseline model which was only trained on PPG samples, showing that the domain transfer is plausible, and has applications in real-time AF detection and monitoring using wearable devices.

2.4 Paper's conclusion

In conclusion, the paper [Shashikumar et al., 2018] provides a well-motivated deep learning architecture for detection of paroxysmal AF, and demonstrates clinically acceptable AF detection accuracies across different recording modalities. Furthermore, the major finding of this study is that combining spectral representation of cardiac pulsatile recordings with traditional indices of heart rhythm irregularity in a deep neural network framework results in better AF classification. This approach facilitates transferring of learned model parameters across recording modalities such as ECG and PPG, thus enabling accurate AF classification in settings with limited access to large patient cohorts for model training purposes. Moreover, the results indicate that the hierarchical architecture of a deep neural network with one or more image-based feature extraction layers, a sequential layer capable of passing temporal information, and an attention mechanism allows for accurate classification of paroxysmal AF.

2.5 Drawbacks

We believe that the recent work on AF detection suffers from some limitations[Shashikumar et al., 2018]. One major drawback in medical applications of machine learning algorithms is the lack of acceptable clinical gold standards for AF detection. The data used in this study - the ECG dataset labels were obtained from automatic bedside Holter software, with a coarse overview by a single adjudicator for potential mislabeled cases of AF. Therefore, the ECG-based AF detection results provided in this work should be taken with some caution. Furthermore, the Holter and Smart Watch data sets were not made public, so it may be hard to reason about the generalization of the model and its performance on previously unseen data. In addition, without large data sets, it's hard to use in such a complex model, so a few changes were needed for adjusting it to our available data.

3 Methods

In this section, we provide an overview of the original approach and our ideas for modifications and improvements, and overview the data which we use to train our models.

3.1 Baseline - Original approach

The baseline model is well-described above, in section 2.2. In summery, the original approach of the problem leveraged the wavelet transform as a basis for a convolutional, sequence based architecture that extracts both spatial and temporal values from the processed signal. We use this model as a baseline for our comparison. After extensive surveying, we found the *"MIT-BIH Atrial Fibrillation" database*, which includes 25 long-term ECG recordings of human subjects with atrial fibrillation (mostly paroxysmal). We chose this database because it was relatively simple to work with, and matched our requirements for detecting AF segments, most of them paroxysmal. Our model thus does not directly measure the AF burden of each segment, but looks as AF segments as those which have at least a single AF incarnation.

3.1.1 Data Description

We use the *"MIT-BIH Atrial Fibrillation Database"* [Amaral et al., 2000, GB and RG., 1983] [4]. This database includes 25 long-term ECG recordings of human subjects with atrial fibrillation (mostly paroxysmal). Of these, 23 records include the two ECG signals. The individual recordings are each 10 hours in duration, and contain two ECG signals, each sampled at 250 samples per second with 12-bit resolution over a range of 10 millivolts. The original analog recordings were made at Boston's Beth Israel Hospital using ambulatory ECG recorders with a typical recording bandwidth of approximately 0.1 Hz to 40 Hz. The rhythm annotation files were prepared manually, these contain rhythm annotations of types `(AFIB` (atrial fibrillation), `(AFL` (atrial flutter), `(J` (`AV` junctional rhythm), and `(N` (used to indicate all other rhythms). In our experiments we've labelled all these rhythms as containing AF. In the original data set, junctional rhythm and and atrial flutter appeared in no more than 10 windows.

[4] Dataset is available at `https://physionet.org/content/afdb/1.0.0/`

3.2 Advanced Model - *TCN-based model*

Our main idea for model improvement is to replace the BRNN layer with a TCN-based approach. This, in our thoughts, will lead to better performances and results, and will gain higher performances. The theoretical motivation for this modification is described below.

3.2.1 Temporal Convolutional Networks - TCN

In this section, a theoretical background on Temporal Convolutional Networks (TCN) will be presented. The work of Lea et al.[Lea et al., 2016] first proposed Temporal Convolutional Networks (TCNs) for video-based action segmentation.
The conventional process of time-series analysis includes extraction of low-level features using a CNN encoder, and then using feeding those features into a recurrent neural network which captures high-level temporal information. Therefore, TCN provides a unified approach to capture all two levels of information hierarchically[Or, 2020].

A TCN is designed following two basic principles:

- The convolutions are causal, meaning that no information leaks from future to past. To achieve this, the TCN uses causal convolutions
- The architecture can take a sequence of any length and map it to an output sequence of the same length (like a RNN). The TCN uses a 1D fully-convolutional network architecture, where each hidden layer is the same length as the input layer[Richter, 2020]. An outline of the architecture is provided in Figure 6.

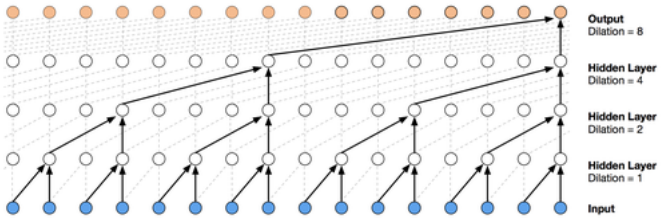

Figure 6: Visualization of a stack of dilated causal convolutional layers (Wavenet, 2016)

Every block in the network is a residual block, which stacks two dilated causal convolution layers together, and the results from the final convolution are added back to the inputs to obtain the outputs of the block. If the width (number of channels) of the inputs and the width(number of filters) of the second dilated causal convolution layers differs, we'll have to apply an 1D convolution to the inputs before the adding the convolution outputs to match the widths.

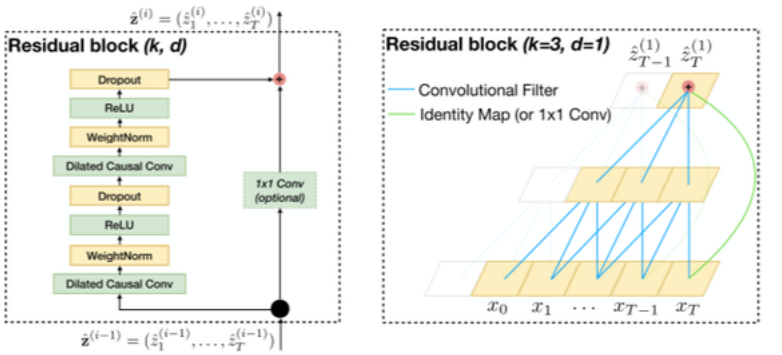

Figure 7: A general TCN residual block (left) and a visualization of the block for $k = 3$, $d = 1$ (right). A 1x1 convolution is added when residual input and output have different dimensions.

It was shown that the TCN architecture performs better on certain tasks and achieves **higher accuracies** [Lea et al., 2016], and furthermore gains some advantage from the following properties [Bai et al., 2018, Lea et al., 2016, N. et al., 2016]:

- **Capturing local information**: Using a convolutional architecture as part of the sequential model helps capture local and translation-invariant features along with temporal features.
- **Parallelism**: Sequence training cannot be parallelized in RNN since RNNs are built as recurrent memory units which receive as input a part of a sequence and the previous state. In contrast, convolutions can be done in parallel since the same filter is used in each layer. Therefore, in both training and evaluation, a long input sequence can be processed as a whole in TCN, instead of sequentially as in RNN.
- **Flexibility**: Much like a 2D CNN, a TCN can change its receptive field size in multiple ways. For instance, stacking more dilated (causal) convolutional layers, using larger dilation factors, or increasing the filter size. TCNs thus affords better control of the model's memory size, and are easy to adapt to different domains.

3.2.2 TCN-based model for AF detection

We propose a new model, similar to the baseline model. We replace the BRNN layer with a TCN layer, which we will denote as `TemporalConvNet`. The new model architecture is visualized in Figure 8.

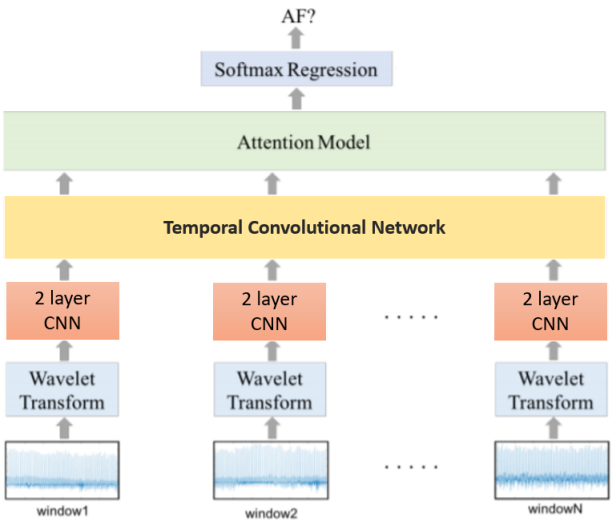

Figure 8: `TemporalConvNet` model. The Temporal Convolutional Network replaces the sequential BRNN model as a temporal feature extraction layer.

The motivation for this change stems from the fact that TCNs are generally very good at capturing information in time-series based inputs. He and Zhao [2019] present a new methodology of applying TCN for anomaly detection in time series. They train the TCN on normal sequences and use it to predict trends in a number of time steps.

In the article, it is shown that TCNs are able to learn inherent patterns in sequential data automatically and so we believe it is a feasible model to learn various time-series behaviors, and can be used for anomaly detection. In addition, the TCN-based approach presented in the article works well on three publicly available, real-world data sets.

This work gives further motivation for using TCNs in time-series-based rhythm detection, and specifically our task of AF detection. The remainder of the architecture is kept the same as the baseline model, since they work well in other time-series based tasks, each fulfilling an important role in the success of such a system.

4 Implementation and experiments

The article's authors didn't publish code or data of the article. **Therefore, we needed to find matching data, and to implement the models without using pre-existing code** [5].

4.1 Data Preprocessing

Loading our dataset resulted in 7009 segments of 2-minute ECG signals [In the article this was 10-min segments], annotated with AF labels. To balance the data we take all the positive examples (with AF) and randomly chose the examples without AF to create, like the article, 70% of negative examples (without AF), and %30 positive examples (with AF). This left us with a total 1360 segments of 2-minute ECG signals. We split each segment into 4 windows, 30 seconds each [In the article they split to 20 segments, but in our model 4 segments gave better results, due to the complexity of the model]. The dataset was sampled with 250 samples per second, which gave a one-dimensional signal of 7,500 samples for each 30-second window. The signal was then downsampled by a factor of 20, and noise above 40Hz was removed using a lowpass butterworth filter. Each signal was transformed using the wavelet transform (a morlet with $\omega_0 = 6$) and a scale of 92 octaves, with a 1/12 scale between octaves and minimum scale - $S0 = 2$. The final data-set was 1460 samples, given as tensors of dimension $(4, 92, 375)$, each representing an array of 4 wavelet power spectra. Implementation of the wavelet transform was made using `PyCWT` [6].

4.1.1 Wavelet Transform

Here is an example for the data preprocessing that we did, and especially the wavelet transform. In compare to the Wavelet Transform that has been shown in the article, we can see that our results are similar and very well. Here is a Wavelet power spectrum for 30 second ECG sample from out data-set after the Wavelet Transform:

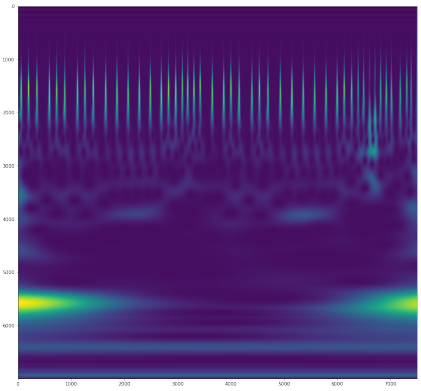

Figure 9: AF segment

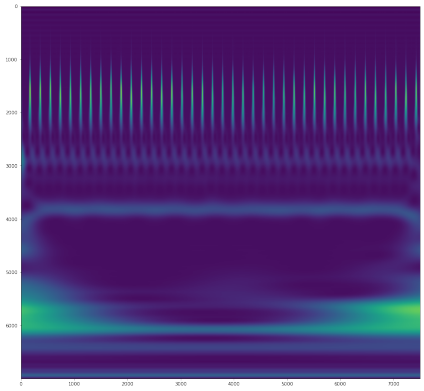

Figure 10: Normal Sinus Rhythm segment

[5] Our full code is available at `https://github.com/kalifadan/Final_project_cs236781`

[6] The Python module is available at `http://regeirk.github.io/pycwt/pycwt.html`

4.2 Baseline implementation

4.2.1 Architecture

We implemented the baseline architecture from scratch using PyTorch [Paszke et al., 2019]. First, a `Conv2d` layer was implemented and duplicated 4 times (corresponding to 30-second windows of a 2-minute ECG sample). The output is then fed into a `BRNN`, comprised of 4 RNN units stacked together, with the initial state, h_0, set to 0. Afterward, a soft attention layer outputs the attended output of the sequence output from the BRNN. the output of the attention layer is fed into a fully-connected linear layer and a Softmax regression. For the data preparation, a `Dataset` was written, which loads the data, splits the ECG windows, and maps their signals to the corresponding wavelet spectra, as described above.

Our baseline model is smaller than [Shashikumar et al., 2018], to account for the smaller amount of data that we worked on. We've experimented with larger sequence lengths and more complex architectures with worse or as-good results compared to our implementation, which suggests that the complexity of the model is indeed better suited for larger datasets such as the Holter ECG dataset.

Signal Quality Index Signal Quality Indices are metrics used across different scientific areas to measure the quality of signals. [Shashikumar et al., 2018] has shown that using external features, and specifically time-series covariates such as the standard deviation of the processed signal, aids in model performance when fed into the softmax regression. The idea behind this is that not all signals are of the same quality, and this model decisions should also be weighed against a the quality of the input it receives. Since we've used a relatively small benchmark data set for our model, which was carefully tailored and manually checked for errors, we've decided to test our baseline without using signal quality indices, and relying solely on the quality of the data. With that, we were still able to achieve fairly good results.

Batch normalization In our CNN implementation, we've add batch normalization layers, to account for deviations the scales of each power spectrum in a batch. We've also experimented in training without batch normalization and found it to be worse than our results.

4.2.2 Training process & Hyperparamters

The BRNN layer was set with a hidden size of 100, and an input size of 50 (according to the 50-feature vector output of each CNN layer). The regression head contains a single fully-connected layer of 2 outputs (for a binary classification). The model was trained end-to-end for a total of 50 epochs. RMSProp was used as the optimizer with a L2 regularization of $\lambda = 0.01$. The learning rate was set at 0.001. Moreover, the loss function used is cross entropy. All the hyperparameters of the model are described below.

- **CNN layer** : Our CNN architecture, comprised of two 2 successive convolutional layers (each layer having a kernel size of 32x64) and a max-pooling layer. Each convolutional layer in the architecture was followed by a layer of activation with Rectified Linear Unit (ReLU) nonlinearity and a Batch-Normalization for the first layer [That we added to improve the model results]. The pooling layer had a pooling region of size 2x2 with a stride of 2 along with both the directions. The number of filters used for each of the convolutional layers was 8. Finally, the output was flattened and fed into a fully connected layer, having a total of 50 outputs. The output of the CNN was a 50-dimensional feature vector that summarized the entire wavelet power spectrum.
- **Bidirectional recurrent neural network (BRNN) layer** : In our model, the size of the hidden state in the forward and the backward layer was 50, with the output at each time step being a 100-dimensional vector.
- **Attention layer & Classification layer** : The Attention layer is as presented in the article, and can be briefly described as receiving a sequence of outputs $Y = [y_1, \ldots, y_N]$, and applying with following transformation to receive an output z:

$$\alpha = \text{softmax}\left(w_{\text{att}}^T Y\right) \tag{1}$$

$$z = Y\alpha^T \tag{2}$$

The idea behind this is that the model learns to attend to certain areas of the sequence. Given inputs Y, the model assigns probabilities to the entire sequence of data using a Softmax function.

4.3 Advanced Model implementation

The data preprocessing and the training process for the advanced model was as same as the baseline model, and we have changed only the model's architecture.

4.3.1 Architecture

This advanced architecture is similar to the baseline architecture except for the `BRNN` component in the model. This component has been replaced by `TCN` - Temporal Convolutional Network. The TCN replaces the bi-directional sequential RNN model as a temporal feature extraction layer.

4.3.2 Training process & Hyperparamters

The model was trained, similar to the baseline model end-to-end for a total of 30 epochs. RMSProp was used as the optimizer and L2 regularizer with regularization parameter $\lambda = 0.01$ was used. The learning rate was set at 0.0005. Cross entropy was used as the loss function. All the hyperparameters for the unchanged layers of the model remained identical to the baseline model.

Temporal Convolutional layer For the Temporal Convolutional layer, a single temporal block with 16 channels was used, with a dropout of 0, and a kernel size of 8. A temporal block consists of 2 `Conv1d` layers, with a ReLU activation, dropout and then a 1D batch normalization layer after each layer.

4.4 Experiments

Our data set after the preprocessing includes 1360 segments of 2-minute ECG signals as described above in the Data preprocessing section, with 70% NSR segments and 30% AF segments. Each segment has a corresponding binary label - AF (1) or no AF (0), and divides into four 30-second wavelet power spectra that are then fed to the CNN layers.
Of the 1360 segments, 80% of them were used for developing the model (training set) and the remaining 20% of the patients were used as the hold out test set.
The training set consisted of a total of 952 segments of which 30% contained at least one episode of AF, and the testing set consisted of a total of 408 segments of which 30% contained at least one episode of AF. The batch size was fixed at 30 samples with data randomly sampled from our data set. The baseline model and advanced model (TCN) were each trained with the above setup. For each model, Specificity, Sensitivity, Area under the ROC curve (AUC), area under the precision-recall curce (AUC_{pr}), speficicity (SPC), F1 score (F_1) and the accuracy (ACC) were calculated for the train and test phases.

4.4.1 Evaluation Metrics

We chose to evaluate and compare the baseline and advanced model using the following metrics, as described in the article (for fairness comparison):

- **AUC**: Area under the receiver operating characteristic (ROC) curve.
- **AUCpr**: Area under the precision recall curve.
- **Accuracy**: Proportion of correct predictions (both true positives and true negatives) among the total number of cases examined.
- **Specificity**: Proportion of negatives that are correctly identified.
- **F1 score**: The harmonic mean of the precision and recall.
- **Sensitivity**: The proportion of positives that are correctly identified (the recall).

All these metrics were calculated for both the training and the test sets. Furthermore, these first four metrics were chosen since in the article they presented these metrics, so for fair comparison we decided to compare these metrics. In addition, we chose to add the F1 score metric - this is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric. Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as in the above case. For this reason, we chose to add the F1 score metric as well (since our data is not perfectly balanced).

5 Results - AF detection performance

After training and testing both of the model as described above, we calculated the chosen metrics and the results are as shown below. As described below, we compare out results to the results from the paper for %5 AF burden (or more) only, since our data don't consist enough samples with bigger AF burden present.

5.1 Original results

Provided here are the original results with %5 AF burden as taken from [Shashikumar et al., 2018]:

	AUC	AUC_{pr}	ACC	SPC
Testing Set	0.94	0 84	0.94	0.95
Training Set	0.96	0 93	0.96	0.96

5.2 Baseline results

We include the results for the baseline model, as well as the ROC curve of the model.

	AUC	AUC_{pr}	ACC	SPC	F_1	SEN
Testing Set	0.83	0 57	0.82	0.80	0.79	0 85
Training Set	0.88	0 70	0.87	0.85	0.86	0 92

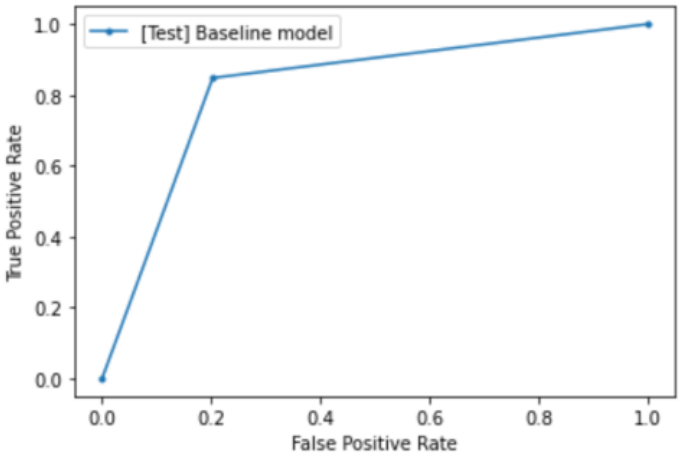

Figure 11: ROC curve for baseline model on testing set

5.3 Advanced model results

We include the results for the TCNNet model, as well as the ROC curve of the model.

	AUC	AUC_{pr}	ACC	SPC	F_1	SEN
Testing Set	0.87	0 72	0.88	0.90	0.87	0 83
Training Set	0.94	0 83	0.94	0.94	0.93	0 94

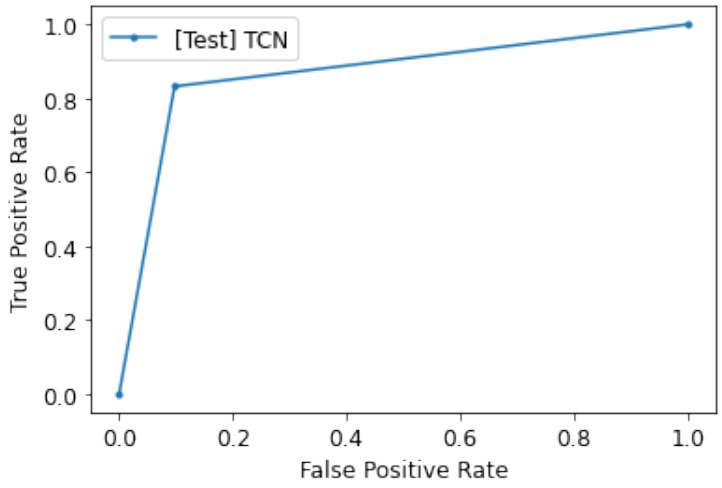

Figure 12: ROC curve for advanced model on testing set

5.4 Discussion and Conclusion

In the baseline model we got AUC of **0.88** on training set, and **0.83** on testing set. Due to the requirement of training the model on data with ratio of 7:3 between the negative and positive samples (from the paper), that led us to get small amount of training data in compare to the article, this results are high. We've shown that even for a small data set, the proposed architecture and baseline perform relatively well, and even with a small amount of (mostly) AF episodes, the baseline model is able to generalize well for novel data. Some over-fitting is inspected when testing the model, which can be attributed to insufficient noise cleaning during preprocessing and sub-optimal parameters for the model. Furthermore, we got accuracy of **0.82** on test set, specificity of **0.8** and F_1 score of **0.79**. In addition, the sensitivity (recall) of the model is **0.85**, thus, we conclude that the model is capable of detecting most of the AF samples successfully (85% of them).

Since our data is imbalanced (having less positive examples to work with), we look at the F_1 score of each model and as we see a substantial improvement over the baseline, we conclude that the model learns to predict better results for positive and negative examples, regardless of data balancing. Furthermore, imporvement in AUC_pr suggests that the model performs well on both negative and positive examples, making better predictions for each class

Our contribution to previous work, as described in this project, is the TCN-based model (the advanced model). As we see in the results, the Temporal Convolutional Network performs better for extracting the temporal features from the sequence, and has better preformances than our baseline model. This improvement was very significant and using the new model we got an AUC of **0.94** on training set, and **0.87** on testing set. Moreover, we got accuracy of **0.94** on training set and **0.88** on test set. On testing set the specificity was **0.9** and F_1 score of **0.87**, compared to a specificity of **0.94** and F_1 score of **0.93** on the training set. In addition, the sensitivity (recall) of the model is **0.83** on testing set and **0.94** on training set, suggesting that the model is indeed capable of detecting most of the AF samples successfully without many prediction errors, and similar to the article results.

In conclusion, with our limitations in mind, the baseline model successfully achieved good results, and the TCN-based model achieved significantly better results.

References

A. Amaral, L. Glass, L. Hausdorff, J. Ivanov, and P.C. Mark. *PhysioNet: Components of a new research resource for complex physiologic signals.* 2000.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* 2014.

S. Bai, Kolter J. Z., and V. Koltun. *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.* 2018.

Marta Carrara, Luca Carozzi, Travis J Moss, Marco de Pasquale, Sergio Cerutti, Manuela Ferrario, Douglas E Lake, and J Randall Moorman. *Heart Rate Dynamics Distinguish Among Atrial Fibrillation, Normal Sinus Rhythm and Sinus Rhythm with Frequent Ectopy.* 2015.

Ingrid Daubechies. *The Wavelet Transform, Time-frequency Localization and Signal Analysis.* 1990.

Moody GB and Mark RG. *A new method for detecting atrial fibrillation using R-R intervals.* 1983.

Yangdong He and Jiabao Zhao. Temporal convolutional networks for anomaly detection in time series. 2019.

Kathleen T. Hickey and C. Riga. *Detection and management of atrial fibrillation using remote monitoring.* 2019.

Colin Lea, Rene Vidal, Austin Reiter, and Gregory D. Hager. *Temporal convolutional networks: A unified approach to action segmentation.* 2016.

Kalchbrenner N., Espeholt L., Simonyan K., Oord A. V. D., Graves A., and Kavukcuoglu K. *Neural machine translation in linear time.* 2016.

Barak Or. *Temporal Convolutional Networks, The Next Revolution for Time-Series?* 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Julius Richter. *Temporal convolutional networks for sequence modeling.* 2020.

Supreeth P. Shashikumar, Amit Shah, Gari D. Clifford, and Shamim Nemati. *Detection of Paroxysmal Atrial Fibrillation using Attention-based Bidirectional Recurrent Neural Networks*. 2018.

Supreeth Prajwal Shashikumar, Amit J Shah, Qiao Li, Gari D Clifford, and Shamim Nemati. *A Deep Learning Approach to Monitoring and Detecting Atrial Fibrillation Using Wearable Technology*. 2017.

Massimo Zoni-Berisso, Fabrizio Lercari, Tiziana Carazza, and Stefano Domenicucci. *Epidemiology of atrial fibrillation: European perspective*. 2014.