# Analysis of Statistical Models for Fast Time Series ECG Classifications

Ramkumar Rathi, Niraj Yagnik, Soham Tiwari, Chethan Sharma*

*Abstract*—**This paper studies and compares several different ways of classifying time series data using machine learning instead of popular deep learning models that need large amounts of data and high computational requirements to train. The models are compared over the MIT-BIH Arrhythmia database, which contains electrocardiogram (ECG) signal data used to monitor a patient's heartbeat, and classifies the data into different types of diseases. The various methods explored to classify this data are Time Series Forests (TSF), Random Interval Spectral Ensemble (RISE), Word extraction for time series classification (WEASEL), and K-Nearest Neighbours with Dynamic Time Warping (DTW). The models are then compared on the basis of the amount of data needed to train, accuracy, precision, recall, time to train, and time to predict per sample. This research aims to compare the performance of these unconventional dictionaries, frequency, and interval-based time series classification models and identify the fastest and most robust algorithm. Time Series Forests emerge to be the fastest ML-based time series classifier, making it suitable for many potential smart devices which desire to perform on-device time series classification.**

*Index Terms*—**Classification, DTW, ECG, KNNs, RISE, Time series analysis, Time Series Forests, WEASEL.**

## I. Introduction

TIME series analysis (TSA) is the detailed analysis of chronological data to extract meaningful insights from historical trends and to be able to make suitable prognostications about the future. Numerous methods have been developed, ranging from mathematical and statistical methods to advanced machine learning and deep learning algorithms. These methods excel at finding causal relationships and patterns in longitudinal data. TSA can help recognise the different trends present and the seasonality in the data set and then utilise the learned trends and seasonality to make appropriate predictions. Hence time series analysis is extensively used in numerous applications which have chronological data such as:

- financial analysis and the prediction of stock prices;
- analysis of the trends in the weather of a region and predict future weather conditions; however, the most useful of them all,
- health analysis, i.e. monitoring and analysing the patients' vitals.

TSA has many potential applications on portable devices. For example, one such application would include deploying a smart weather balloon, which can monitor the atmospheric variables and even perform computations on devices or in the field of healthcare, such as in smartwatches that have ECG sensors and can monitor the heart rate. The usefulness of these devices can improve if they can perform TSA on the time series data they collect. It would be useful to predict weather conditions using the collected weather data or classify the health and condition of the heart by analysing the Electrocardiogram (ECG) signals. However, one significant limitation is that portable smart devices usually have very low compute power. Hence, expecting these devices to run state of the heart TSA algorithms would be very difficult. Any deep learning algorithm would require a lot of compute power to make real-time predictions. On the other hand, ML algorithms are relatively more efficient and run on low compute devices.

Hence, this paper compares the performance of different dictionary, frequency, and interval-based time series classification algorithms. We have chosen to compare the algorithms on the ECG dataset (1). The motivation behind using the ECG dataset is that it belongs to the domain of healthcare which we are pretty motivated about. We try to identify the fastest and most robust algorithm by comparing them on several metrics such as accuracy, precision, recall, dataset size and time to train and predict and various other model parameters. In doing so, we find the most efficient algorithm of the ones we test. This algorithm can also be a candidate to be deployed on low compute devices to make rudimentary and preliminary assessments of the wearer's heart conditions with high accuracy and speed. This might also be extended to other such applications with appropriate research.

Electrocardiogram (ECG) signals are used to monitor the patient's heartbeats. ECG's record the electrical activity of the heart muscles, as electrical pulses stimulate the heart muscles to contract and pump blood throughout a body. As a result, ECG's can also record and detect any anomaly in a person's heartbeat, technically known as arrhythmia. Hence, ECG recordings are instrumental in monitoring the heart's condition and help determine diseases that might ail one's heart as an anomaly or irregularity can be immediately recorded in the ECG readings. Hence ECG analysis immensely benefits from various TSA algorithms. Given an ECG reading of a heart, these algorithms can yield insights about the heart's condition and quickly detect abnormalities that afflict the heart. Every cardiac cycle is typically defined

Manuscript received August 13, 2021; revised March 7, 2022.

Ramkumar Rathi is a final year student of the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India. (email:ryrathi@gmail.com)

Niraj Yagnik is a final year student of the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India. (email: nirajyagnik80@gmail.com)

Soham Tiwari is a final year student of the Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India.(email: soham.tiwari800@gmail.com)

Chethan Sharma is a Senior Assistant Professor in Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India. (Corresponding Author, e-mail: chethan.sharma@manipal.edu)..

by a sequence of wave shapes known as a P wave, QRS complex wave, and T wave - as showin in Figure 1. Intervals between different patients' wave forms, as well as their shapes and orientation, show physiological activities taking place in the heart and the autonomous nervous system.

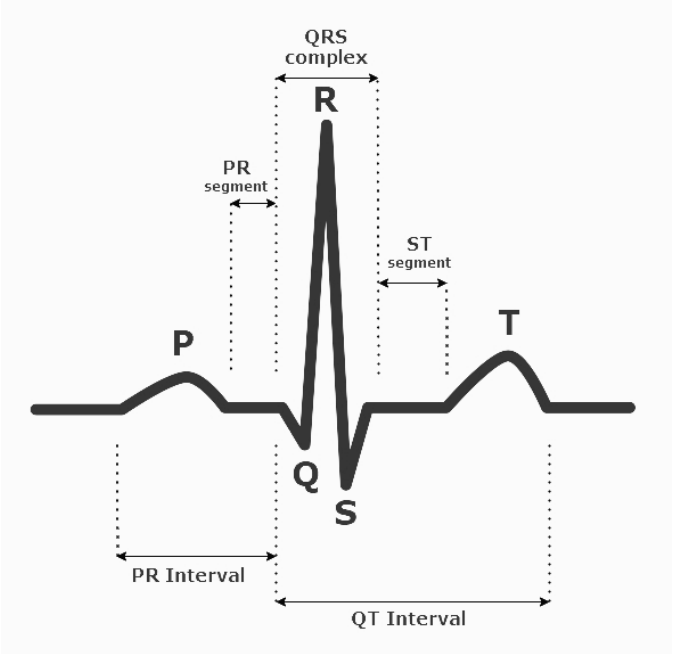


Fig. 1. ECG Diagram with P-QRS-T Waves

The remainder of this paper is structured as follows:

- In Section II, we briefly look at the state-of-the-art, deep-learning-based TSA algorithms and also look at ML-based time-series algorithms.
- In Section III we look at the research methodology, explaining the steps involved in running the different ML-based time series classification algorithms (TSC) on our ECG Data.
- Section IV contains the analysis of the results of our different algorithms on the EEG dataset.
- Section V compares the different TSC algorithms based on the results. These results are further compared with various other models as well.
- Section VI we state our limitations and constraints and possible future work.
- Section VII we state our conclusion.

## II. Literature Review

This paper looks at the various machine learning methods used for time series classifications for ECG analysis. Time series classifications are a popular domain used over various problem statements, and the most common methods used pertain to deep learning. Some popular methods of recognition of various problems in ECG include the use of support vector machines (SVM) (2), Recurrent neural networks (RNNs) (3), Convolutional Neural Networks (CNNs) (4), Multilayer perceptron's (MLP) (5) and Markov Models (6). Convolutional Neural Network models have shown high accuracy scores reaching up to 97.2% as mentioned (4), but require a large amount of data, as well as computational time to train and make these predictions. 'Long short term memory networks', which are a more popular choice for time series data, do not work particularly well with ECG classifications and are restricted to an accuracy of 88.1%. Normal RNNs further degrade the accuracy to 85.4%, and GRU's prove to provide the least accuracy at 82.5% (3). The same data and computational power problems persist for these methods as well. For our research, we have focused on Time Series Forests (7), Random Interval Spectral Ensemble (RISE) (8), Word ExtrAction for time SEries classification (WEASEL) (9) and K-Nearest Neighbours models (10).

*Time Series Forests (TSF)* (7) is an efficient and accurate time series classifier. It consists of a tree ensemble algorithm that uses simple statistical features that summarise each time series and then use the summary to train its algorithm. It also uses a novel metric as a splitting criterion, resulting in TSF outperforming other baseline algorithms such as nearest neighbours with dynamic time warping (NNDTW).

*Random Interval Spectral Ensemble (RISE)* (8) has proved helpful in the problems involving speech processing and long time-series data, such as in audio processing, where the discriminatory features of the dataset are found in the frequency domain rather than the time domain. It is a tree-based, ensemble time series algorithm, The class probability for each sample is then computed based on the majority vote of all the trees.

*Word ExtrAction for time SEries cLassification (WEASEL)* (9) follows the bag-of-patterns approach. Hence, it can achieve high classification accuracy with incredible speeds, making it suitable for application in domains with high run time and quality constraints. The model's robustness also makes it highly suitable for application in smart-grid systems (9).

*K-Nearest Neighbours (with Dynamic Time Warping)* (10) is a distance-based algorithm. It is a simple and robust algorithm that does not require a high amount of hyper-parameter tuning. It has become a commonly used benchmark for evaluating the performance of several time-series classification algorithms.

Regarding literature on the classification of arrhythmia using the MIT BIH database, one of the initial studies on how the MIT BIH dataset impacted and shaped the development of arrhythmia detection was shown by Geroge et al. (11). It gave rise to numerous works and has become the most popular database for arrhythmia classification and detection.

Markos et al. (12) published an influential study on this where they used time-domain analysis to produce features, which were then fed into 63 different types of neural networks in various distinct combinations. The outputs of these neural networks were then fed into a decision tree, where the diagnosis of arrhythmia was made. Similar to this study, short-time Fourier transform and other time-frequency distributions were used in the time-frequency analysis.

Karimifard (13) used the modelling of signals and the Hermitian basis function to get a feature vector. This was then sent to a KNN classifier to classify the different types of arrhythmia. An important conclusion in this study showed how the feature vector's size affected training times.

Hamid et al. (14) compared the use of different time series transformation methods such as CWT (Continuous wavelet transform), DCT (Discrete Cosine Transform) and DWT (Discrete Wavelet Transform) to improve the classifications

of ECG datasets when using Multi-Layer Perceptron's (MLP) and Support Vector Machines (SVM). They used the feature extraction methods separately on the ECG signal and formed 4 structures using MLP and 4 others using SVMs. Then, they compared the efficiency of these feature extraction methods on time taken for the model to train.

Oscar (15) segmented the ECG signals from the MIT BIH dataset and used three methods – fuzzy KNN's, Multi-Layer Perceptron with Gradient Descent and momentum Backpropagation, and Multi-Layer Perceptron with Scaled Conjugate Gradient Backpropagation to classify them into 5 classes. They finally achieved the highest accuracy of 98% by combining the outputs of all three methods in a Mamdani type fuzzy inference system.

Roland et al. (16) used Fast Fourier Transform as the method of pre-processing for the signals and then passed this to a neural network. They achieved an accuracy of 98.6% and concluded that the diagnosis of heart abnormalities and early detection of debilitating medical conditions would improve when the medical community would adopt neural networks to analyze test data samples further.

Shirin et al. (17) used a block-based neural network which consisted of a 2D array of blocks connected. The structure of each block was dependent on the number of incoming and outgoing signals. The inputs in this method consisted of temporal features and the Hermit function coefficient. The network structure and weights were then optimized using Particle Swarm Optimization (PSO). This provided a unique method to overcome the possible change of ECG from person to person as the BBNN would have a unique structure for every person. They achieved an accuracy of 97%.

Saroj et al. (18) proposed the use of yet another deep learning-based method called the Restricted Boltzmann Machine (RBM) model. After the initial normalization and segmentation of signals, the RBM is used to extract the essential features, which are then passed to a SoftMax activation function. They classified the heartbeats into 4 distinct types achieving an accuracy of 98.61%. We mention the drawback of using specialized hardware and large datasets for this method to be useful, which our paper tries to solve.

Nirmala et al. (19) used Dual-Tree Complex Wavelet Transform (DT-CWT) to extract features and then further passed them to a neural attention mechanism to improve on the capturing of temporal patterns from the extracted features. The attention-based model was trained using 8 different arrhythmia classes and achieved an accuracy of 98.5%.

Convolutional neural networks are well-researched methods for arrhythmia classification. Bambang Et al. (20) showed how a 1 dimensional CNN could be used to detect atrial fibrillation detection with an accuracy of 96.36%. This study consisted of 3 normal, AF and non-AF and was trained on 8232 records based on 8 different datasets. Mohammad et al. (21) similarly used a CNN to classify the MIT BIH dataset into 5 different classes instead, obtaining an overall accuracy of 95.2%. Özal et al. (22) further extended the number of classes to 17 and obtained an accuracy of 91.33% using a 1D convolutional neural network. The study's objective also extended to the time taken per classification, which averaged at 0.015s per sample. They thus proposed using this model for mobile devices and cloud computing. Mengze et al. (23) use a 12 layer deep one dimensional CNN to classify the signals into 5 distinct classes. They achieve an accuracy of 97.2%, which is higher than other CNN, random forest and SVM based methods.

## III. Research Methodology

### A. Data Analysis & Pre-processing

The dataset being used in this study is the popular MIT-BIH Arrhythmia database (11) created by teams at Beth Israel Deaconess medical centre and MIT between the years 1975 and 1979. This database contains two-channel ambulatory ECG recordings samples of 47 patients resulting in 48 hours' worth of data. There are a total of 5 output classes representing different types of human heartbeat patterns, which are:

- Non-Ectopic Beats (N) - These represent normal healthy human heartbeats.
- Supraventricular Ectopic Beats (S)
- Ventricular Ectopic Beats (V)
- Fusion Beats (F)
- Unknown Beats (Q)

A graphical representation of each pattern is demonstrated in Figure 3.The distribution of the different classes in the dataset has been provided in Figure 4.

For the analysis and the application of various machine learning methods on this dataset, the data was cropped into smaller subsets resulting in a new dataset. This dataset has a total of 109,446 samples. Each ECG sample has been clipped and down-sampled to a 188 point dimension, i.e., each sample has a length of 188. In the scenario where there are fewer points available, the samples are padded with zeroes to maintain uniform sample lengths. The dataset samples are representative of the electrocardiogram signals of the heartbeat, ranging from normal heartbeats to those afflicted by different arrhythmia's and myocardial infarction, commonly known as a heart attack.

It is evident from the distribution chart (**Figure 2**) that the dataset is largely imbalanced and should ideally not be used in this state for any analysis to be conducted on it. Hence, the data is re-sampled to provide an equal distribution of data across both test and training datasets. Each sample after this step accounts for 20% of the distribution. Next, Gaussian noise is added to each sample to increase the generality of the algorithms during the training phase.

### B. Training

We have applied four different time series models on the given ECG data. **Figure 1** illustrates a general pipeline of the proposed work.:

*1) Time Series Forests:* Classical random forest models are not popularly known to be used on time series data but can be modified to do the same. They can be considered robust and efficient models to classify time series data.
Our application is a multi-class classification problem that classifies the ECG data into five classes. The forest consists of various time series trees (7), which summarise data intervals based on a few features. The time-series data is broken up into intervals of random length. However, the

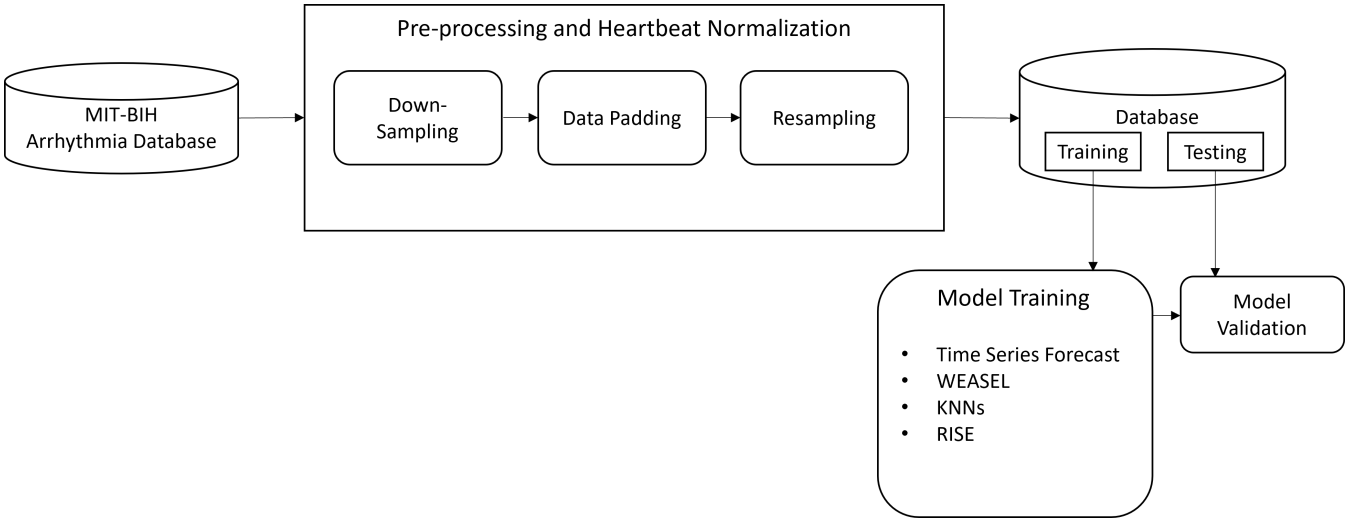


Fig. 2. Illustration of the architecture of the proposed work.

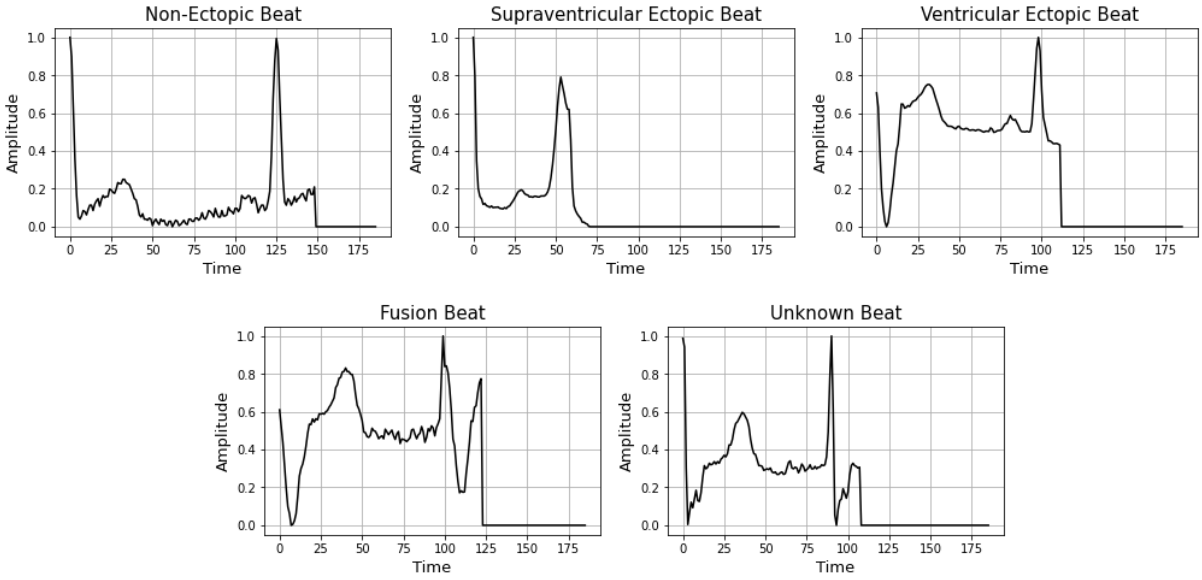


Fig. 3. Graphical Representation of each class in the database.

TSF algorithm (7) suggests using interval sizes which are the square root of M to reduce the size of the feature space from

$$O(M^2)$$

to

$$O(M),$$

where *M* is the number of time dimension points in each sample. In our dataset, M has been set to 188 dimension points.

Next, for each time interval from every sample, three features are computed, which are:

- *Mean* – Mean of the values of each dimension point in the selected interval
- *Standard deviation* – measuring how much the value of each dimension point varies or deviates from the previously computed mean of the selected time interval
- *Slope* – Slope of the regression line which best fits the values of the dimension points for the selected time interval.

The trees in the time series forest are constructed using these three features. The split criteria at each node of a tree are decided using a new metric called *Entrance*. The paper on TSF (7) explains that Entrance is the combination of the entropy gain and an additional measure called *Margin* to choose the best split criteria for the node. Hence, the TSF is a collection of multiple time series trees and classifying a sample into a particular class based on the majority of trees present in the forest.

Hence, time series forest can be considered a modification of the random forest algorithm and can be summarised by the following steps:

1) Splitting the series into multiple intervals.
2) Extracting features (mean, standard deviation and slope) from each interval.
3) Training a decision tree on the extracted features.
4) Ensemble steps 1 – 3.

*2) Random Interval Spectral Ensemble (RISE):* RISE models draw inspiration from tree-based ensemble algorithms, for example, time-series forests (TSF). Like the TSF

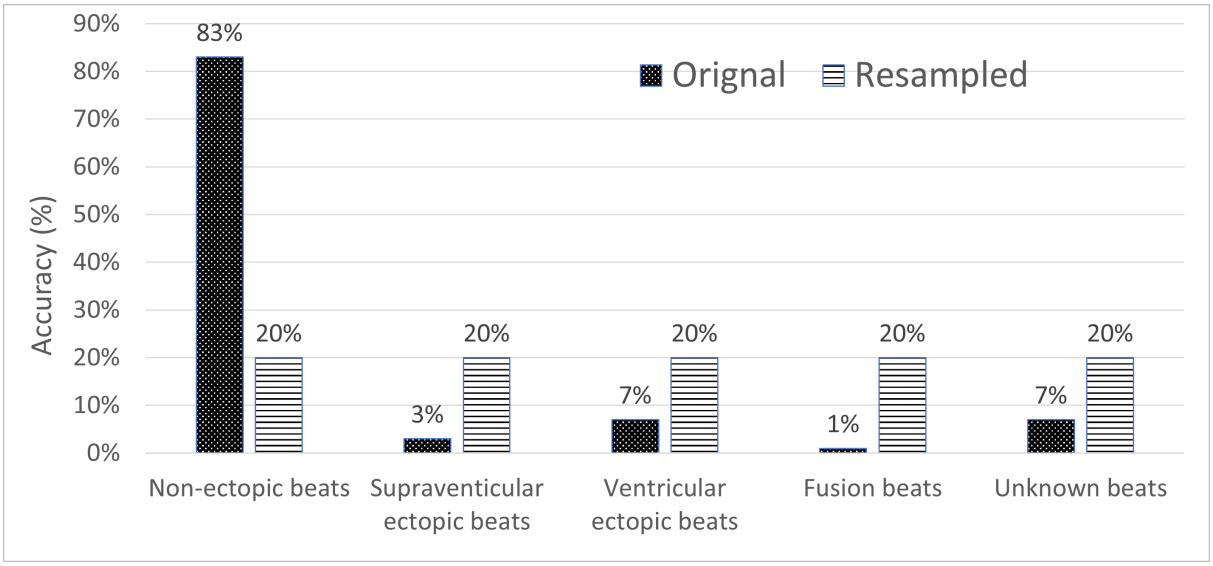


Fig. 4. Class distribution before and after resampling.

**Algorithm 1:** Build time series forest, **T = (X,y)**

Let $K$ represent the number of trees, $p$ the minimum interval length, $M$ the length of the series
Let $F \leftarrow [F_1, F_2, ..., F_K]$ be the different trees in a forest
$R \leftarrow \sqrt{M}$
$i \leftarrow 1$
**while** $i < K$ **do**
  $s$ a matrix with $r$ rows and $3r$ columns
  **for** $j \leftarrow 1$ to $r$ **do**
    $b \leftarrow$ randomBetween$(1, m - p)$
    $e \leftarrow$ randomBetween$(b + p, m)$
    **for** $t \leftarrow 1$ to $R$ **do**
      $s_{t,3(j-1)+1} \leftarrow$ mean$(x_t, b, e)$
      $s_{t,3(j-1)+2} \leftarrow$ standardDeviation$(x_t, b, e)$
      $s_{t,3(j-1)+3} \leftarrow$ slope$(x_t, b, e)$
    **end for**
  **end for**
  buildTree$(s, y)$
  $i \leftarrow i + 1$
**end while**

model, the algorithm constructs a random forest classifier using random intervals from data. TSF computes the time domain statistical summary features by calculating each interval's mean, variance, and slope. On the other hand, RISE only creates a single interval for each tree instead of multiple and computes spectral features for this interval instead of a statistical summary. The length of this interval is chosen as a power of 2. The first tree is a special case in which the interval is of the same length as the time series, i.e. the whole series is taken as the first tree interval(24).

Power Spectrum (PS) features, Auto Correlation-based Features (ACF), or a combination of the two can be used here. Each interval is transformed using the Fast Fourier Transform (FFT) and ACF. This help make RISE faster as compared to the Partial Auto Correlation Function (PACF) and Auto-Regressive (AR) model features used in the original RISE algorithm(25)(26)(24).

This paper uses a combination of both PS and ACF features to train the model. RISE with a combination of PS and ACF is known to produce significantly better results than other spectral ensembles in terms of speed and do not significantly affect the accuracy. RISE also tends to control the run time by creating adaptive models of the time required to build each tree, proving helpful when large intervals are present (8).

**Algorithm 2:** Build RISE, **T = (X, y)**

Let $M$ be the length of training series, $K$ the number of trees, and $p$ the minimum interval length
Let $F \leftarrow [F_1, F_2, ..., F_K]$ be the different trees in a forest in the forest
$i \leftarrow 1$
**while** $i < K$ **do**
  **if** $i = 1$ **then**
    $r \leftarrow M$
  **else**
    $r \leftarrow$ powerOfTwoInterval$(p, MaxInterval)$
  **end if**
  $T' \leftarrow T[b : b + r]$
  $s \leftarrow$ retrieveSpectralFeatures$(T')$
  buildRandomTreeClassifier$(s, y)$
  $i \leftarrow i + 1$
**end while**

*3) WEASEL:* WEASEL uses several novel ideas to achieve a Time Series Classification (TSC) model, which is invariant to noise or distortion present in the time series data and can be easily scaled to classify a large number of time series samples of varying lengths.

WEASEL first uses windows of multiple lengths for a single time series to generate sub-time series of varying lengths. Each such window is then transformed using Fourier transforms and selects discriminative features that help differentiate the different time series data classes using the Analysis of Variance (ANOVA) F-test. An ANOVA(27) test

in simple words helps to understand the difference between the independent variables, or in this case, the features, and how does it affect the dependent response variable. Also, to retain the temporal order in a time series, WEASEL includes a bi-gram model, which helps encode the order relationship of the time series data in the model. Finally, to reduce the large number of features generated, WEASEL uses a Chi-squared statistical test to determine relevant features, eliminating insignificant features whose absence does not affect the model's classification accuracy. As a result, the feature-set obtained is highly effective in differentiating between the different time series classes, then used in a fast logistic regression model to make predictions. The logistic regression model proves to be much quicker than other complex classification algorithms due to its simplicity and provides good classification accuracy, a merit of the feature set obtained.

**Algorithm 3:** Build WEASEL, **T = (sample, I)**

Let $I$ represent the number of Fourier values to keep
$b \leftarrow$ empty BagOfPattern
**for all** $w$ in $AllWindowLengths$ **do**
  $AllWindows \leftarrow$ slidingWindow($sample, w$)
  normalize($AllWindows$)
  **for all** ($prevWindow, window$) **in** $AllWindows$ **do**
    $word \leftarrow$ quantTransform($window, I$)
    $bag[w + word]$.increaseCount()
    $prevWord \leftarrow$ quantTransform($prevWindow, I$)
    $bag[w + prevWord + word]$.increaseCount()
  **end for**
**end for**
**return** ChiFeatureSelect($bag$)

*4) KNNs:* KNN is a simple classification algorithm that is often used as a baseline for various classification problems, as it classifies samples simply by comparing them with their neighbours. The sample is labelled as the class with the most frequency in the first k closest neighbours, which are considered the most similar to the sample.

The standard KNN algorithm, however, makes use of the Euclidean Distance measure to determine the similarity of the input sample with all the other samples present in the dataset. While this metric is suitable for cross-sectional data, it is not ideal for time series data classification. Two time-series samples can have a similar structure but have different phases, i.e., the two waves are displaced along time. Therefore, the Euclidean distance metric will compute the difference between values corresponding to the same instance and hence might not determine whether the two phase-shifted time series are similar.

The use of dynamic time warping (DTW) makes the k-nearest neighbours' algorithm suitable for application to time series data. The distance between any two samples is measured using DTW, which measures the similarity between sequences that may be displaced relative to each other along time or may have different speeds or lengths.

KNNs with DTW is used to establish a baseline accuracy quickly and easily for the time series classification problem. However, this algorithm is computationally very expensive as it requires a lot of time to compare each time sequence with all other time sequences and requires a lot of memory in doing so. Moreover, this algorithm fails to explain why the sequence was labelled a particular class.

The algorithm is also susceptible to noise in the dataset, which can severely distort the shape of a time sequence resulting in it suffering a deviation from its valid class label.

**Algorithm 4:** K Nearest Neighbors, **T = (X, y)**

Let $N$ be a list of neighbors
$DistMatrix \leftarrow$ calculateDistancesUsingDWT($X, y$)
$N \leftarrow$ identifyKNN($DistMatrix$)
$Label \leftarrow$ findMode($N$)
**return** $Label$

**Algorithm 5:** Dynamic Time Warping, **Input: (X, y)**

Let $S$ be a two dimensional matrix with $N \times M$ dimensions
$S[0][0] \leftarrow 0$
**for** $i \leftarrow 1$ **to** $M$ **do**
  $S[0, i] \leftarrow \inf$
**end for**
**for** $j \leftarrow 1$ **to** $N$ **do**
  $S[j, 0] \leftarrow \inf$
**end for**
**for** $i \leftarrow 1$ **to** $N$ **do**
  **for** $j \leftarrow 1$ **to** $M$ **do**
    $cost \leftarrow$ dist($X[i[, y[j]$)
    $S[i, j] \leftarrow cost+$ $\min(S[i, j-1], S[i-1, j], S[i-1, j-1]))$
  **end for**
**end for**
**return** $S[N, M]$

## IV. Analysis and Results

Each of the mentioned four models has been tested using different values for the respective hyper-parameters. Plots for the model's accuracy vs the different hyper-parameter values have also been shown. The results for each run have been gathered and tabulated. The various columns in the tabulated results are:

- The *Estimators* column will only be found in the results of the tree-based algorithms. Estimators refer to the number of trees used in forest algorithms.
- The *Number of samples* column indicates the number of samples over which the model was trained on.
- The *Micro precision* is micro averaged precision. Micro averaged precision helps capture the performance of different models when faced with datasets with class imbalance. They can account for the imbalance in the dataset of a multi-class classification problem, hence providing more insights about the model's precision and classifying classes with low samples.

$$Micro - precision = \frac{\sum_1^5 TP_i}{\sum_1^5 TP_i + FP_i}$$

- The *Macro Precision* column is macro-averaged precision. Macro averaged precision reports the generalised precision values over the entire dataset. The metric fails to account for any dataset imbalance in a multi-class classification problem. It will report a good precision score even when the poorly performing classes witness poor precision scores. This is because the metric is the simple arithmetic mean of the precision scores of each class. As a result, the low scores will be compensated by the high-performing classes' high scores. Hence the macro-precision might provide misleading inferences about imbalanced datasets.

$$Macro - precision = \frac{\sum_1^5 Precision_i}{5}$$

- The *Micro Recall* column is the score computed by micro-averaging the recall values for all the classes. This is a better metric to analyse the recall of the model, classifying an imbalanced dataset into multiple classes.

$$Micro - recall = \frac{\sum_1^5 TP_i}{\sum_1^5 TP_i + FN_i}$$

- The *Macro Recall* is computed by computing a macro-average of all the recall scores, i.e. finding the arithmetic mean of the individual recall scores for the various classes. Like all other macro-averaged metrics, it fails to account for any class imbalance and is prone to reporting misleading scores.

$$Macro - recall = \frac{\sum_1^5 Recall_i}{5}$$

- The *Accuracy* column reports accuracy scores of different models for different configurations of the various hyper-parameters involved. It tells us how many have been correctly classified out of all the classified samples. Another point to note is that the same values are reported for the metrics of micro-precision, micro-recall, and accuracy due to the nature of micro averages.
- The *Time to train* column signifies the amount of time (in seconds) for the model to fit the entire dataset.
- The *Time per sample* column, on the other hand, provides the average time taken for the model to predict a class for an input sample to the model.

Keeping in mind the definitions of the various columns, below are the results provided for the various models run over the dataset.

### A. Time Series Forests

The hyper-parameters being varied here are – *Estimators* and *Number of Samples*.

They are the two main factors that determine the accuracy of the TSF model. The number of estimators refers to the number of trees used in the time series forest, and the number of samples refers to the number of data points in our training dataset. Regardless of the sample size, the characteristics of the training data remain the same, all being of length 188 points and running at 125Hz. The class distribution of the data is also unchanged over the various sample sizes.

A brief analysis of the results in Table I would show that the micro-precision reports higher values than the macro-precision, contrary to the general trend of micro-precision being less than or almost equal to the macro-precision. The same is observed in the case of micro-recall and macro-recall.

Looking at Table I, one can see that the sample size of data is more critical to the accuracy than the number of estimators. Larger sample size with a sparse number of estimators consistently outperforms a small sample size with a large number of estimators.

The time required to predict a single sample is highly correlated with the number of estimators and does not depend on the sample size. The time to train depends on the sample size and the number of estimators.

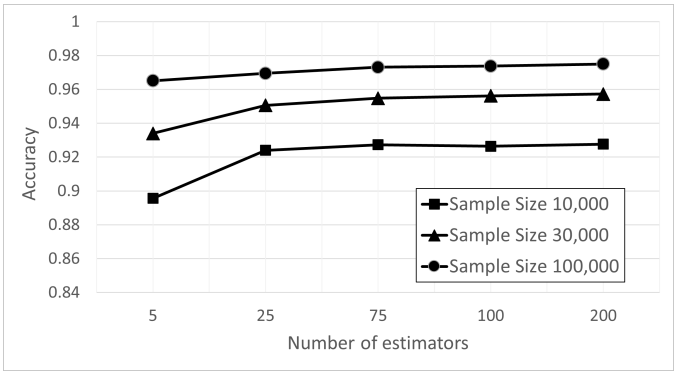


Fig. 5. TSF results - Accuracy vs. Number of estimators

### B. Random Interval Spectral Ensemble

We test this model over a varying number of estimators. It can be seen in Table II that RISE, like TSF, also reports higher micro-averages than macro-averages for both the precision and recall metrics.

RISE classifiers, compared to TSFs, take a much longer time to train. The experiment in Table II was run on eight concurrent processes, unlike the TSF experiment, which was run only on 2. The RISE classifier still takes a lot more time to train and is slower while predicting samples (Time per sample).

In comparison, a TSF model would be much more efficient to run when compared to RISE classifiers. This paper does not attempt to train RISE classifiers with more than 200 estimators as we do not have the required computational power.

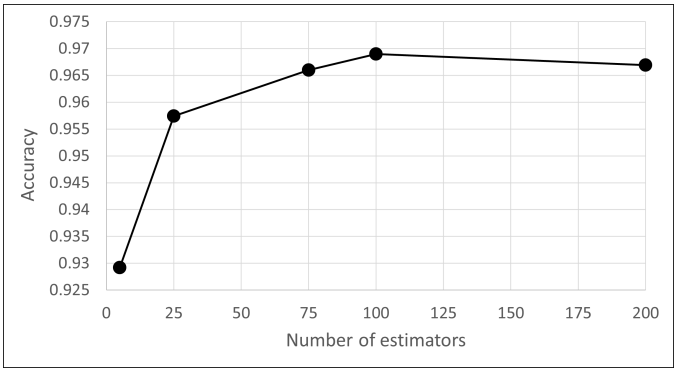


Fig. 6. RISE results - Accuracy vs. Estimators

TABLE I
TSF RESULTS

| Time Series Forest | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Estimators** | **Number of samples** | **Accuracy** | **Precision** | | **Recall** | | **Time to train** | **Time per sample** |
| | | | **Micro** | **Macro** | **Micro** | **Macro** | | |
| 5 | 10000 | 0.89562 | 0.89562 | 0.63011 | 0.89562 | 0.85634 | 17.57 | 0.00081 |
| 25 | 10000 | 0.92403 | 0.92403 | 0.67994 | 0.92403 | 0.89051 | 84.99 | 0.00388 |
| 75 | 10000 | 0.92731 | 0.92731 | 0.68503 | 0.92731 | 0.88728 | 162 | 0.01175 |
| 100 | 10000 | 0.92636 | 0.92636 | 0.68384 | 0.92636 | 0.88752 | 204 | 0.01567 |
| 200 | 10000 | 0.92764 | 0.92764 | 0.68631 | 0.92764 | 0.88818 | 408 | 0.03171 |
| 5 | 30000 | 0.93399 | 0.93399 | 0.71601 | 0.93399 | 0.85385 | 31.86 | 0.00078 |
| 25 | 30000 | 0.950575 | 0.950575 | 0.76586 | 0.950575 | 0.87486 | 150 | 0.00394 |
| 75 | 30000 | 0.95477 | 0.95477 | 0.77821 | 0.95477 | 0.88441 | 462 | 0.01176 |
| 100 | 30000 | 0.95619 | 0.95619 | 0.78238 | 0.95619 | 0.88696 | 588 | 0.01571 |
| 200 | 30000 | 0.95724 | 0.95724 | 0.78702 | 0.95724 | 0.886205 | 1182 | 0.03123 |
| 5 | 100000 | 0.96505 | 0.96505 | 0.86382 | 0.96505 | 0.84041 | 102 | 0.00079 |
| 25 | 100000 | 0.96948 | 0.96948 | 0.88571 | 0.96948 | 0.84103 | 474 | 0.00452 |
| 75 | 100000 | 0.97309 | 0.97309 | 0.90377 | 0.97309 | 0.84973 | 1452 | 0.01183 |
| 100 | 100000 | 0.97374 | 0.97374 | 0.90293 | 0.97374 | 0.85416 | 1998 | 0.01589 |
| **200** | **100000** | **0.97501** | **0.97501** | **0.91257** | **0.97374** | **0.8567** | **3888** | **0.03204** |

TABLE II
RISE RESULTS

| RISE Classifier | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Estimators** | **Sample Size** | **Accuracy** | **Precision** | | **Recall** | | **Time to train** | **Time per sample** |
| | | | **Micro** | **Macro** | **Micro** | **Macro** | | |
| 5 | 100000 | 0.92921 | 0.92921 | 0.70977 | 0.92921 | 0.76124 | 1327 | 0.02821 |
| 25 | 100000 | 0.95742 | 0.95742 | 0.80425 | 0.95742 | 0.82071 | 7227 | 0.03714 |
| 75 | 100000 | 0.96596 | 0.96596 | 0.85157 | 0.96596 | 0.82987 | 10287 | 0.09561 |
| 100 | 100000 | 0.96898 | 0.96898 | 0.88352 | 0.96898 | 0.82747 | 13417 | 0.12636 |
| **200** | **100000** | **0.96692** | **0.96692** | **0.86028** | **0.96692** | **0.82503** | **27414** | **0.26895** |

### C. WEASEL

The number of samples is the hyper-parameter being varied for different test runs of the WEASEL model. We can observe that an increase in the number of samples in the dataset does not elicit a proportional increase in the model's accuracy. The model accuracy peaks at 10,000 samples and then observe a decline in the model accuracy with a further increase in the number of samples in the dataset.

On the other hand, an increase in the number of samples results in an expected increase in the time required to train the model. The model has training times similar to RISE, much larger than TSF. The best performing configuration, which witnessed an accuracy of 94.779%, falls short of the best 96.692% accuracy of the RISE model. However, the best configuration of WEASEL requires one-sixth of the time needed to train that of RISE.

Furthermore, WEASEL is the only model that reports higher or similar macro averages compared to micro averages for precision and recall.

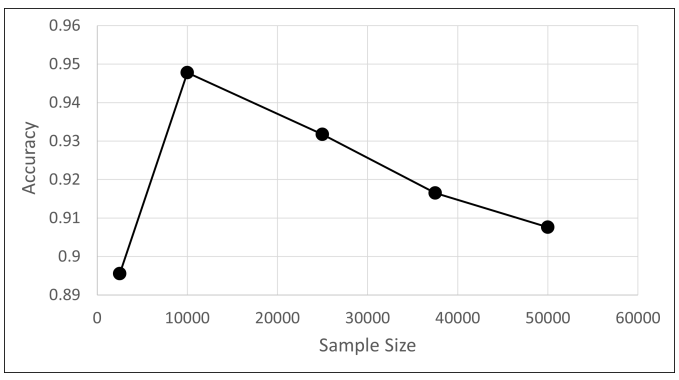


Fig. 7. WEASEL results - Accuracy vs. Sample Size

### D. KNNs

The KNN algorithm was tested using different values of the number of samples in the dataset used and the number of neighbours.

The maximum accuracy achieved for the number of neighbours set to 100 was approximately 89%. However, the time to train per sample of the dataset is much larger than all the other models, even for the base number of 5000 samples in the dataset. Another observation is that the time to train per sample is majorly dependent on the number of samples in the dataset, with very little dependence on the number of neighbours the algorithm queries to determine the class of a particular sample.

One can also observe higher accuracy when using ten neighbours instead of 100 neighbours in the algorithm for the corresponding number of dataset samples.

Finally, KNNs with DTW also report higher micro averages than macro averages.

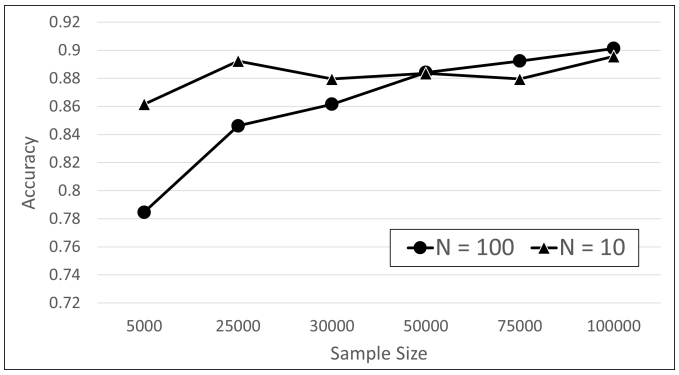


Fig. 8. KNN results - Accuracy vs. No. of Samples

TABLE III
WEASEL RESULTS

| WEASEL | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sample Size | Accuracy | Precision | | Recall | | Time per sample | Time to Train (s) |
| | | Micro | Macro | Micro | Macro | | |
| 2500 | 0.89558 | 0.89558 | 0.89907 | 0.89558 | 0.89706 | 0.03696 | 839 |
| **10000** | **0.94779** | **0.94779** | **0.95199** | **0.94779** | **0.94859** | **0.03771** | **4525** |
| 25000 | 0.93172 | 0.93172 | 0.94004 | 0.93172 | 0.93282 | 0.03997 | 12935 |
| 37500 | 0.91646 | 0.91646 | 0.92618 | 0.91646 | 0.91322 | 0.04235 | 21195 |
| 50000 | 0.90763 | 0.90763 | 0.92379 | 0.90763 | 0.90763 | 0.04612 | 30393 |

TABLE IV
KNN RESULTS

| K-Nearest Neighbors | | | | | | | |
|---|---|---|---|---|---|---|---|
| Sample Size | Accuracy | Precision | | Recall | | Time per sample | Neighbors |
| | | Micro | Macro | Micro | Macro | | |
| 5000 | 0.86153 | 0.86153 | 0.7516 | 0.86153 | 0.85964 | 0.13077 | 10 |
| 25000 | 0.89231 | 0.89231 | 0.77485 | 0.89231 | 0.88188 | 0.65698 | 10 |
| 30000 | 0.879518072 | 0.879518072 | 0.885011933 | 0.879518072 | 0.881456095 | 0.796532 | 10 |
| 50000 | 0.883534137 | 0.883534137 | 0.893715061 | 0.883534137 | 0.885612958 | 1.336411468 | 10 |
| 75000 | 0.879518072 | 0.879518072 | 0.892954956 | 0.879518072 | 0.881265132 | 1.999505015 | 10 |
| 100000 | 0.895582329 | 0.895582329 | 0.909117694 | 0.895582329 | 0.897456095 | 2.726337341 | 10 |
| 5000 | 0.78461 | 0.78461 | 0.66766 | 0.78461 | 0.840388 | 0.13389 | 100 |
| 25000 | 0.84615 | 0.84615 | 0.72854 | 0.84615 | 0.8513 | 0.65904 | 100 |
| 30000 | 0.86153 | 0.86153 | 0.75205 | 0.86153 | 0.86601 | 0.77946 | 100 |
| 50000 | 0.884165 | 0.884165 | 0.76455 | 0.884165 | 0.8781 | 1.33083 | 100 |
| 75000 | 0.89231 | 0.89231 | 0.77495 | 0.89231 | 0.88227 | 2.00243 | 100 |
| **100000** | **0.90131** | **0.90131** | **0.798857** | **0.90131** | **0.88644** | **2.63226** | **100** |

## V. MODEL COMPARISON

In Figure 9, the highest possible accuracy's of different models have been plotted against the corresponding time take taken for each sample by the model. One can infer from Figure 9 that KNNs not only take the most time per sample but also report the least accuracy. RISE reports a reasonably high accuracy with respectable times per sample. However, it is not as good as the other two models. Although WEASEL takes a comparable amount of time per sample as TSF, TSF performs better, as it is able to report higher classification accuracy.

Consequently, TSF is the best candidate TSC algorithm for deploying on low-compute portable devices. It reports the highest classification accuracy of the four ML TSC models and takes the least time to predict a class for each input sample. As a result, one may expect the TSF algorithm to run smoothly on low-compute devices. However, further performance testing with actual hardware is needed to be carried out to confirm the same.

Table V showcases the performances of various state-of-the-art deep learning-based models and compares them to the proposed work. They compare the accuracy of prediction over all classes, the type of model use and the number of classes that the data is classified into. The four machine learning algorithms presented by the proposed research work provides comparable accuracy results when compared to the accuracy scores of the corresponding models.

TSF outperforms most deep learning models, thus verifying its efficacy despite utilising considerably lower hardware resources and compute for training. RISE and WEASEL also provide comparable results with several research works, further validating the consistency and reliability of the proposed work.

## VI. LIMITATIONS & SCOPE FOR FUTURE WORK

The proposed work has been limited to comparing four different machine learning (ML) algorithms. Our goal is to experiment with lightweight algorithms that can potentially be deployed on portable devices. They should be capable of providing accurate results in the least amount of time with a smaller dataset. That is why we aimed to use machine learning instead of deep learning models such as recurrent neural high computational requirements and large datasets, leading them to be less than ideal for healthcare purposes on portable systems.

However, deep learning models such as recurrent neural networks (RNN) and convolutional neural networks (CNN) have been found to give a higher degree of accuracy for our requirement in some cases. Furthermore, genetic algorithms have also been used in the past to classify time series data, which can also be experimented with for ECG classification. Scope for future work here would include increasing the accuracy of these machine learning models while honouring the same time and data constraints specified. Apart from that, there is a further need to look into different neural network models which are faster in their approach and may not need a large amount of data, considering medical data is not as largely available as other forms.

## VII. CONCLUSION

Time series classification is a famous area of research, with a wide array of algorithms already available and new, faster, and better algorithms appearing every year. For our application of ECG classification, we wanted an algorithm that is robust and light enough to be potentially deployed on portable devices. We explored four algorithms – TSF, RISE, WEASEL and KNNs with DTW. TSF outshone the rest of the three algorithms tested in this paper. To summarise this comparison:

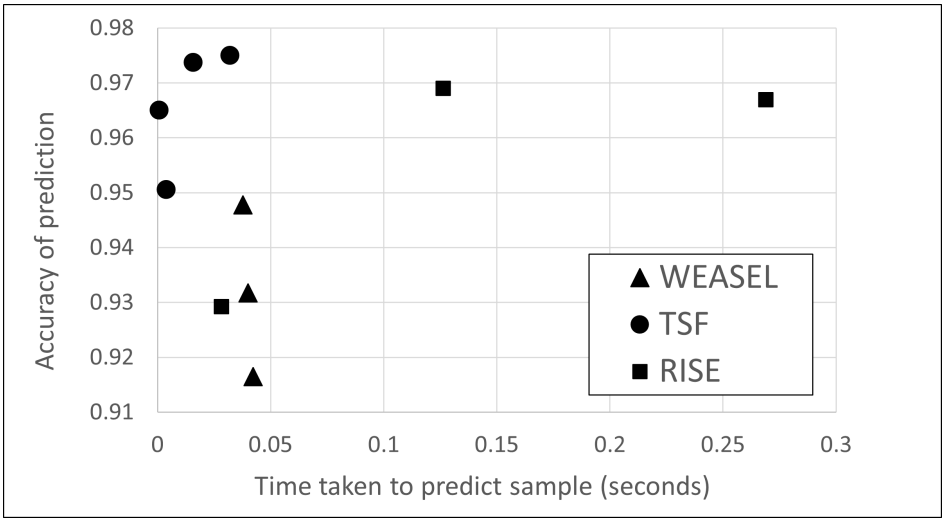


Fig. 9. Different models' performance comparison - Accuracy vs. Time per sample

TABLE V
RESULT COMPARISON WITH OTHER MODELS

| Author/Paper | Method | Classes | Accuracy |
|---|---|---|---|
| Pandey et al.(28) | CNN (SMOTE) | 5 | 98.3 |
| Acharya et al.(29) | CNN (11 Layers) | 4 | 94.3 |
| Chi et al.(30) | LDA | 5 | 96.23 |
| Oscar. (15) | NN | 5 | 98 |
| Roland et al.(16) | FFT + NN | 6 | 98.6 |
| Yeh et al.(31) | Cluster analysis | 5 | 94.3 |
| Manu et al.(32) | MLP | 5 | 94.64 |
| Ping. (33) | K-mean & SVM | 12 | 98.92 |
| Ali et al. (34) | Conventional ANN | 2 | 92.4 |
| Vasileios. (35) | SVM | 3 | 95.35 |
| Mehrdad. (36) | Negative corelation learning | 2 | 96.02 |
| Saroj et al. (18) | Stacked RBM | 5 | 98.61 |
| Shririn et al. (17) | Block based NN + PSO | 5 | 97 |
| Nimmala et al. (19) | Attention based NN | 8 | 98.5 |
| Bambang et al. (20) | AFibNet (CNN) | 3 | 96.36 |
| Ozal et al. (22) | 1D CNN | 17 | 91.33 |
| Mohammad et al. (21) | CNN | 5 | 95.2 |
| Mengze. (23) | CNN (12 Layers) | 5 | 97.2 |
| **Proposed Work - KNN** | **KNN** | **5** | **90.13** |
| **Proposed Work - WEASEL** | **WEASEL** | **5** | **94.79** |
| **Proposed Work - RISE** | **RISE** | **5** | **96.69** |
| **Proposed Work - TSF** | **Time series forest** | **5** | **97.5** |

- Using TSF, we get an impressive accuracy of 97.5%. TSF reports the highest classification accuracy amongst all four ML-based TSC algorithms and reports the least amount of time taken to predict a class for each input sample.
- WEASEL takes a comparable amount of time as TSF to predict a class for each input sample. However, it reports lower classification accuracies than TSF.
- RISE reports classification accuracies comparable to TSF; however, it takes more time to predict a sample for each input class.
- KNNs with DTW report the lowest classification accuracy and take significantly more time to predict a class for each input sample than other algorithms.

The results highlight the potential of TSF to be used in various TSC applications, such as in healthcare devices where the devices can continuously monitor the ECG readings of the heart, and the model can detect any ailments by classifying the ECG readings. We also hope that our project also ignites further research in the application of time series analysis, especially in the domain of healthcare, with the hope that ML can help advance existing healthcare technologies and not only help increase the longevity of human life but also help in understanding the unexplained aspects of human anatomy.

## REFERENCES

[1] G. Moody and R. Mark, "Mit-bih arrhythmia database," Feb 2005. [Online]. Available: https://physionet.org/content/mitdb/1.0.0/

[2] A. Kampouraki, G. Manis, and C. Nikou, "Heartbeat time series classification with support vector machines," *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 13, pp. 512–8, 09 2008.

[3] S. Singh, S. K. Pandey, U. Pawar, and R. R. Janghel, "Classification of ecg arrhythmia using recurrent neural networks," *Procedia Computer Science*, vol. 132, pp. 1290–1297, 2018, international Conference on Computational Intelligence and Data Science.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918307774

[4] M. Wu, Y. Lu, W. Yang, and S. Y. Wong, "A study on arrhythmia via ecg signal classification using the convolutional neural network," *Frontiers in Computational Neuroscience*, vol. 14, p. 106, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2020.564015

[5] A. Das, F. Catthoor, and S. Schaafsma, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ecg," in *Proceedings of the 2018 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*, ser. CHASE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 69–74. [Online]. Available: https://doi.org/10.1145/3278576.3278598

[6] R. Andreão, B. Dorizzi, and J. Boudy, "Ecg signal analysis through hidden markov models," *IEEE transactions on bio-medical engineering*, vol. 53, pp. 1541–9, 09 2006.

[7] H. Deng, G. C. Runger, E. Tuv, and V. Martyanov, "A time series forest for classification and feature extraction," *CoRR*, vol. abs/1302.2277, 2013. [Online]. Available: http://arxiv.org/abs/1302.2277

[8] M. Flynn, J. Large, and T. Bagnall, "The contract random interval spectral ensemble (c-rise): The effect of contracting a classifier on accuracy," in *Hybrid Artificial Intelligent Systems*, H. Pérez García, L. Sánchez González, M. Castejón Limas, H. Quintián Pardo, and E. Corchado Rodríguez, Eds. Cham: Springer International Publishing, 2019, pp. 381–392.

[9] P. Schäfer and U. Leser, "Fast and accurate time series classification with WEASEL," *CoRR*, vol. abs/1701.07681, 2017. [Online]. Available: http://arxiv.org/abs/1701.07681

[10] A. Abouyahya and S. El Fkihi, "An optimization of the k-nearest neighbor using dynamic time warping as a measurement similarity for facial expressions recognition," in *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*, ser. LOPAL '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3230905.3230921

[11] 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.

[12] M. Tsipouras and D. Fotiadis, "Automatic arrhythmia detection based on time and time–frequency analysis of heart rate variability," *Computer methods and programs in biomedicine*, vol. 74, pp. 95–108, 06 2004.

[13] S. Karimifard, A. Ahmadian, M. Khoshnevisan, and M. Nambakhsh, "Morphological heart arrhythmia detection using hermitian basis functions and knn classifier," vol. 1, 02 2006, pp. 1367–70.

[14] H. Khorrami and M. Moavenian, "A comparative study of dwt, cwt and dct transformations in ecg arrhythmias classification," *Expert Syst. Appl.*, vol. 37, pp. 5751–5757, 08 2010.

[15] O. Castillo, P. Melin, E. Ramírez, and J. Soria, "Hybrid intelligent system for cardiac arrhythmia classification with fuzzy k-nearest neighbors and neural networks combined with a fuzzy system," *Expert Syst. Appl.*, vol. 39, pp. 2947–2955, 02 2012.

[16] E. Adams and A. Choi, "Using neural networks to predict cardiac arrhythmias," 10 2012, pp. 402–407.

[17] S. Shadmand and B. Mashoufi, "A new personalized ecg signal classification algorithm using block-based neural network and particle swarm optimization," *Biomedical Signal Processing and Control*, vol. 25, pp. 12–23, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809415001743

[18] S. K. Pandey, R. R. Janghel, A. V. Dev, and P. K. Mishra, "Automated arrhythmia detection from electrocardiogram signal using stacked restricted boltzmann machine model," 2021.

[19] V. J. K. S. B. M. B. G. S. M. B. L. B. Nimmala Mangathayaru, Padmaja Rani, "An attention based neural architecture for arrhythmia detection and classification from ecg signals," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 2425–2443, 2021. [Online]. Available: http://www.techscience.com/cmc/v69n2/43850

[20] B. Tutuko, S. Nurmaini, A. E. Tondas, M. Naufal Rachmatullah, A. Darmawahyuni, R. Esafri, and A. Sapitri, "Afibnet: An implementation of atrial fibrillation detection with convolutional neural network," 02 2021.

[21] M. M. Rahman Khan, M. A. Bakr Siddique, S. Sakib, A. Aziz, A. K. Tanzeem, and Z. Hossain, "Electrocardiogram heartbeat classification using convolutional neural networks for the detection of cardiac arrhythmia," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2020, pp. 915–920.

[22] Özal Yıldırım, P. Pławiak, R.-S. Tan, and U. R. Acharya, "Arrhythmia detection using deep convolutional neural network with long duration ecg signals," *Computers in Biology and Medicine*, vol. 102, pp. 411–420, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482518302713

[23] M. Wu, Y. Lu, W. Yang, and S. Y. Wong, "A study on arrhythmia via ecg signal classification using the convolutional neural network," *Frontiers in Computational Neuroscience*, vol. 14, p. 106, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2020.564015

[24] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst, "A tale of two toolkits, report the third: on the usage and performance of hive-cote v1.0," 2020.

[25] "Autoregression: Model, autocorrelation and python implementation," Mar 2021. [Online]. Available: https://blog.quantinsti.com/autoregression/

[26] J. Brownlee, "A gentle introduction to autocorrelation and partial autocorrelation," Aug 2020. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/

[27] S. K. Gajawada, "Anova for feature selection in machine learning," Oct 2019. [Online].

Available: https://towardsdatascience.com/anova-for-feature-selection-in-machine-learning-d9305e228476

[28] S. K. Pandey and R. R. Janghel, "Automatic detection of arrhythmia from imbalanced ecg database using cnn model with smote," *Australasian Physical & Engineering Sciences in Medicine*, vol. 42, pp. 1129 – 1139, 2019.

[29] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ecg segments with convolutional neural network," *Information Sciences*, vol. 405, pp. 81–90, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025517306539

[30] Y.-C. Yeh, W.-J. Wang, and C. W. Chiou, "Cardiac arrhythmia diagnosis method using linear discriminant analysis on ecg signals," *Measurement*, vol. 42, no. 5, pp. 778–789, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224109000050

[31] Y.-C. Yeh, C. W. Chiou, and H.-J. Lin, "Analyzing ecg for cardiac arrhythmia using cluster analysis," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1000–1010, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417411010633

[32] M. Thomas, M. K. Das, and S. Ari, "Automatic ecg arrhythmia classification using dual tree complex wavelet based features," *AEU - International Journal of Electronics and Communications*, vol. 69, no. 4, pp. 715–721, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1434841114003641

[33] C.-P. Shen, W.-C. Kao, Y.-Y. Yang, M.-C. Hsu, Y.-T. Wu, and F. Lai, "Detection of cardiac arrhythmia in electrocardiograms using adaptive feature extraction and modified support vector machines," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7845–7852, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417412001066

[34] A. Isin and S. Ozdalili, "Cardiac arrhythmia detection using deep learning," *Procedia Computer Science*, vol. 120, pp. 268–275, 2017, 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091732450X

[35] V. Tsoutsouras, D. Azariadi, S. Xydis, and D. Soudris, "Effective learning and filtering of faulty heart-beats for advanced ecg arrhythmia detection using mit-bih database," *EAI Endorsed Transactions on Pervasive Health and Technology*, vol. 2, no. 8, 12 2015.

[36] M. Javadi, S. A. A. A. Arani, A. Sajedin, and R. Ebrahimpour, "Classification of ecg arrhythmia by a modular neural network based on mixture of experts and negatively correlated learning," *Biomedical Signal Processing and Control*, vol. 8, no. 3, pp. 289–296, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1746809412001127