2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI) | 979-8-3503-4891-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDSAAI59313.2023.10452629

Time Series Classification using Attention-Based LSTM and CNN

Vaibhava Lakshmi R
Department of Computer Technology,
MIT campus, Anna University,
Chennai, India
vaibhavi18092002@gmail.com

Radha S
School of Advanced Sciences,
Vellore Institute of Technology,
Chennai, India
radha.s@vit.ac.in

***Abstract*—Time series classification has many real-world applications, including medical diagnosis, financial forecasting, and environmental monitoring. Accurately classifying time series data can provide valuable insights and help make informed decisions in various fields. This paper proposes the attention-based LSTM - CNN framework for classifying time series data. Our framework incorporates joint extraction of spatio-temporal features from a convolutional neural network and attention-based long- and short-term memory. The extracted features are fed into a Random Forest classifier to obtain the final output classes. We have used the Wafer dataset from the UCR repository for experimentation. An accuracy of 99.86 % was achieved for the proposed model, hence transcending the performance of the other baseline models.**

***Index Terms*—Attention-based LSTM, Time Series Classification, Random Forest classifier, Convolutional Neural Network**

I. Introduction

Time series is a succession of temporal data points collected over a definite period of time and is ever-present. In short, a time series is used for observing factors that influence some variables from time to time and can track changes over milliseconds, days, or even years. Time series data is produced daily, like weather readings, financial recordings, application performance monitoring, network data, Web3 and blockchain data, biological signals, or IoT and sensor data. Time series can be either univariate or multivariate. Time series can also be categorized as Continuous and Discrete data. A) Continuous Time Series: This type of time series data is measured continuously over time without any gaps or interruptions in the data collection process. For example, continuous time series data could be the daily temperature measurements a weather station collects over a year. B) Discrete Time Series: This type of time series data is measured at specific intervals over time, and there may be gaps or missing data in the collection process. For example, discrete time series data could be the number of daily visitors to a website over a year, where some days might not have any visitors due to holidays or other events.

Analyzing a collection of time series data points obtained over a period of time is called Time Series Analysis. It has a range of applications in statistics, sales, economics, medical and weather-related fields and constitutes a multitude of tasks like forecasting, classification, pattern recognition, segmentation, curve fitting, descriptive, and explanative analysis. Classification of time series refers to categorizing its data points into different classes or labels based on their patterns, features, or characteristics. Time series classification is a focal task in Machine Learning that might involve financial, medical, sensor, or speech and activity data. Classification of time series addresses many real-world problems.

Some specific reasons why classification of time series is necessary are: 1) Prediction: Classification of time series can help recognize trends and patterns in the data that can be used to predict future values of the time series. For example, classifying stock prices as bullish or bearish can help investors predict the stock market's future direction.2) Anomaly detection: Classification of time series can help to identify abnormal or unexpected behavior in the data, which can indicate potential problems or issues. For example, classifying network traffic as normal or abnormal can help detect potential security threats. 3) Signal processing: Time series classification can also be used to analyze signals in various audio, image, and video applications. For example, classifying audio signals as speech or music can help identify the audio type being processed.

The current research trends in time series classification include improving the accuracy and efficiency of classification algorithms, handling large and complex datasets, dealing with missing data and outliers, and developing interpretable models for decision-making. Overall, time series classification is expected to remain a major research field with a significant impact on various industries and applications in the future.

In this work, we proffer an ensemble of CNN and attention-based LSTM models to extract the spatial and temporal features and concatenate them into a Random Forest classifier to obtain the final output classes.

II. Background

A. Traditional methods

A multitude of traditional time series classification algorithms have been put forth in the past years and can be spanned to enhance the performance of classification over six focal.

categories: 'Distance- based', 'Feature-based', 'Model-based', 'Shapelet-based', 'Interval-based' and 'Dictionary-based' techniques. Distance-based techniques are those in which, between two series, similarity measure is defined. Distance-based methods like k – Nearest Neighbors (k - NNs) along with Dynamic Time Warping is very efficacious in classifying multivariate time series. But, owing to the temporal nature of time series data, noise, high dimensionality and the differing lengths of the series, finding a suitable distance measure can be really a difficult task [1]. The Feature- based classification algorithms are also used which rely majorly on extracting features from the time series data. Two such effective distance-based approaches are 'Hidden State Conditional Random Field (HCRF)' and 'Hidden Unit Logistic Model (HULM)'. However, these algorithms are onerous to be implemented owing to the difficulty of obtaining the intrinsic features of time series
data [2].

The Model-based classifiers assume that all the time series in a class are furnished by the same basal model, and thus a new series is given with the class of the model that best fits. These can be Hidden Markov models or statistical methods, which do not perform at par with the deep learning models [3]. Small sub-shapes or subsequences of time series which are indicative of a class are called Shapeless. Shapeless with discriminatory power are searched for in the Shapelet-based techniques. The Shapelet Transform classifier is one such method. The backdrops of using this technique are that the searched shapelet lacks generalization and also very extensive computation is required [4].

The Interval-based methods divide the time series into multiple intervals and further each interval is used to train a 'Machine Learning model'. Time Series Forest is one such algorithm. This approach brings about a group of classifiers which acts on its own subsequence. The final classification is allocated based on the most common class generated by the individual classifiers. This technique can lead to a long computation time for huge datasets as the runtime increases with the length of the time series sample, the number of samples, and the number of features per interval [5]. Another familiar Time Series Classification approach is 'Dictionary-based', in which the frequency of occurrences of patterns is recorded. But, the implementation of these techniques is highly time consuming [6].

There is also ongoing research in developing efficient and scalable time series classification algorithms for large-scale datasets. One approach is to use dimensionality reduction techniques such as Principal Component Analysis (PCA) or matrix factorization to reduce the size of the data. Another approach is to use ensemble methods such as bagging and boosting to algorithms [14].

B. *Deep Learning for time series classification*

Traditional machine learning methods for time series classification relied on manually engineered features, which can be time-consuming and difficult to generalize across different datasets. In contrast, deep learning models can automatically extract relevant features from raw time series data, without the need for feature engineering. Deep learning models can handle noisy time series data and learn to ignore irrelevant features. They can capture complex temporal dependencies and patterns in the data that may be difficult to model with traditional machine learning methods and they can be trained completely, meaning that the feature extraction and classification steps are learned simultaneously, resulting in improved performance. In recent years, there has been a flow of interest in 'deep learning-based' approaches for time series classification [7]–[9].

Convolutional neural networks (CNNs), which are commonly used in image classification, have also been applied to time series classification by treating the time series as one-dimensional signals. These networks use filters to extract features from the time series, and the resulting feature maps are used for classification. Other deep learning approaches, such as autoencoders and deep belief networks, have also been used for time series classification. Autoencoders are neural networks that learn to encode and decode data in an unsupervised manner. They can be used for time series classification by training the autoencoder on a set of time series, and then using the learned encoding to classify new time series [8]–[11].

Recurrent neural networks (RNNs), which can handle sequential data and capture temporal dependencies, have been applied to time series classification with promising results. In particular, long short-term memory (LSTM) networks, which are a type of RNN with memory cells that can retain information over long time periods, have been shown to be effective for time series classification [12]. Another important research direction in time series classification is the development of interpretability methods. Deep learning models are often considered black boxes, making it difficult to understand how they make their predictions. However, interpretability methods such as attention mechanisms and saliency maps can provide insight into which parts of the input signal are significant for the classification decision [13].

Overall, time series classification is a mature and active research field that continues to make important contributions to various applications. The use of deep learning-based approaches has shown great promise, and ongoing research in interpretability and efficiency will further advance the field.

III. LITERATURE SURVEY

Wang et. al. [7] have examined three Deep Neural Network models for classification of Time Series data, namely: 1) the Multilayer Perceptron constituting three fully connected layers with each 500 neurons stacked together with a dropout used at each layer's input, 2) the Fully Convolution Networks constituting of three stacked convolution blocks with sizes of filters 128, 256 and 128 respectively, followed by the feeding of the features extracted being fed into a global average pooling layer to reduce the number of weights and the final class is furnished by the SoftMax layer. 3) Residual blocks are built using Convolution blocks. Three Residual blocks are stacked together for the final ResNet followed by the global average pooling and SoftMax layers. The proposed neural networks were tested on the University of California, Riverside's Time Series repository and assessed using the Mean Per-Class Error (MPCE) measure.

Karim et. al. [8] come up with a multivariate time series classification model by transformation of the LSTM-FCN and Attention-based LSTM-FCN and augmenting the fully convolutional block using a squeeze-and-excitation block to ameliorate the accuracy along with very less pre-processing. The proffered framework outperformed all the baseline models. Fawaz et. al. [9] have put forth Inception Time — a hybrid of 5 different Inception models. Each prediction is assigned an equal weightage. The proposed model was evaluated on 85 disparate datasets from the UCR archive. It was observed that Inception Time's complexity improves linearly with an increase in the length of the time series, unlike 'HIVE-COTE' (baseline), whose execution is almost two times slower. Also, the results on overall accuracy of several hyperparameters of the CNN architecture are analyzed. Du et. al. [10] have put forth an ensemble model comprising of an attention-based LSTM for capturing temporal dependencies and CNN for recording the spatial sparsity from the time series datasets. Experimentally it was proven that the proposed method outperforms the other traditional techniques used for TSC.

Tang et. al. [11] have put forth naive 1D-CNN block called the OS-block. The focal idea behind their proposal is to cover every RF size using prime number design in an effective manner. Owing to the ability to capture scale effectively, the best performance is achieved compared to several TSC state-of-the-art techniques. Malhotra et. al. [12] have come up with TimeNet, a multi-layered 'recurrent neural network' (RNN) trained in an unsupervised manner to extract features from time series. Based on the sequence-to-sequence models that modify differing length time series to fixed dimensional vector representations, the encoder network of an autoencoder is what is TimeNet. The TimeNet was trained on 24 different datasets from sundry domains from the UCR archive and then evaluation was done on 30 other datasets not used for training. Turbe et. al. [13] have proffered a technique with quantitative metrics to evaluate the performance of extant post-hoc interpretability techniques, specially in classification of time series. Drawbacks such as the reliance on 'human judgement', 'retraining' and the 'alteration in the data distribution' while obstructing samples, are addressed in this framework. They have designed a synthetic dataset with known differentiable features and alterable complexity. The proffered methodology and 'quantitative metrics' can be used to comprehend the reliability of interpretability techniques' results furnished in practical applications. In turn, they can be embedded within operational workflows in significant fields that need precise interpretability results. Six interpretability methods were taken into consideration which record a wide range of available techniques, while keeping the problem computationally tractable. They are: (1) 'DeepLift'
(2) 'GradShap' (3) 'Integrated Gradients' (4) 'KernelShap'
(5) 'DeepLiftShap' and (6) 'Shapley'. Relevance identification and Relevance attribution were the two aspects considered for the performance evaluation of these methods.

IV. Proposed framework

The proffered attention-based LSTM-CNN is composed of an attention-based 'Long Short Term Memory network' and a 'Convolutional Neural Network'. The architecture is depicted in Fig. 1. The components of the proposed model are described as follows:

A. Feature extraction

1) Attention-based LSTM: Attention mechanisms in LSTM (Long Short-Term Memory) networks are often used for time series classification tasks because they allow the network to selectively focus on various parts of the input sequence deppon their relevance to the task at hand. Traditional LSTM networks process the input sequence and generate a fixed- length representation of the sequence, which is then fed into a classifier. However, this fixed-length representation may not capture all the relevant information in the sequence, and different parts of the sequence may be more important for different input instances. Attention mechanisms in LSTM networks allow the network to dynamically weight the importance of each element in the input sequence based on its relevance to the task. This can be particularly useful for time series classification tasks where different parts of the sequence may be more relevant for different input instances. For the time series given as $x_1, x_2, x_3, \ldots\ldots, x_T$, where x_i R^n, the LSTM unit is computed iteratively as follows:

$$h_i = \alpha(W[h_{i-1}; x_i] + b_s) \quad (1)$$

where W $[h_{i-1};\ x_i] \in \mathrm{R}^{m+n}$ is a concatenation of input x_i at time step i and previous hidden state h_{i-1}. $W \in R^{m\times(m+n)}$ is a weight matrix. The weighted feature of the attention mechanism at time i is given as:

$$I_i = \sum_{i-T_x}^{i} \alpha_j h_j \qquad (2)$$

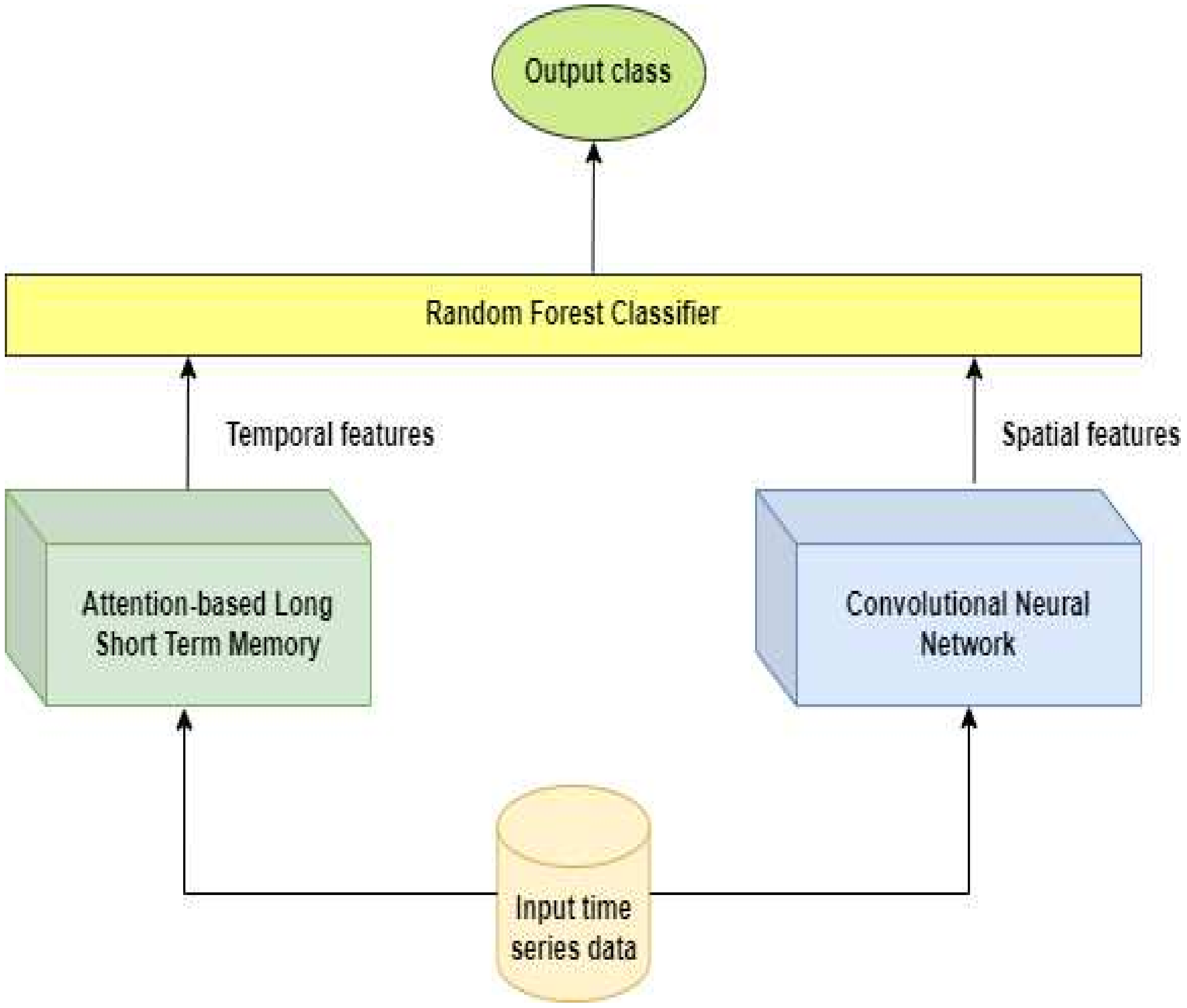

Fig. 1. Architecture of the proposed framework

where T_x is the window size of the attention weight calculation and α_j refers to the attention weight which is computed as:

$$\propto_j = \frac{exp\left(h_j^T q_s\right)}{\sum_{i-T_x}^{i} exp(h_k^T q_s)} \quad (3)$$

where the vector $q_s \in R^m$ is randomly initialized and will be optimized while training happens. It helps to select pertinent hidden states across every time step.

Convolutional Neural Network: Convolutional layers in a CNN extract local features from a sequence of data, and the use of pooling layers allows the network to focus on the most relevant features while also reducing the dimensionality of the input. Additionally, CNNs learn hierarchical representations of the data, which can record intricate patterns at different levels of abstraction. This ability to learn complex representations of the input data is particularly useful for time series classification tasks, where the patterns in the data may be nonlinear and difficult to capture using traditional machine learning techniques. The

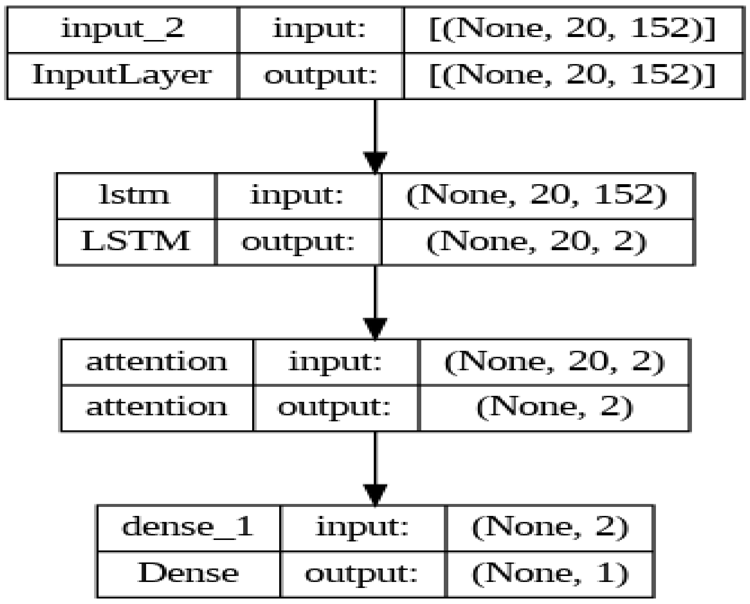

Fig. 2. Architecture of the attention-based LSTM model

CNN model is composed of 1-Dimensional Convolution layers and finally ensued by Global Average Pooling layer. We input the time series data $x_{i-T_x:i}$ into the CNN network and obtain the features $c_i \in R^s$ at time step i.

B. Classification of the extracted features

For the final classification of the concatenated features $[l_i, c_i]$, where $[I_i, C_i] \in R^{m+s}$ the prediction class is obtained using the Random Forest classifier. Random Forest is a machine learning algorithm and it is generally used for classification tasks. It is an ensemble learning technique that incorporates multiple decision trees to make a prediction. In a Random Forest classifier, a bootstrap sample of the input dataset is created by randomly selecting n samples from the original dataset with replacement. This produces a new dataset of size n that contains some duplicates and some missing samples. And then, several decision trees are grown independently on various random subsets of the data. Every decision tree in the Random Forest is trained with a different subset of the data and features, so they all make independent predictions.

To make a prediction, each tree in the 'Random Forest' independently predicts the class of the input data point, and the class with the most votes among all the trees is selected as the final prediction. At each node of the Decision tree, a random subset of m features is selected from the original set of p features. The number m is typically much smaller than p, and it is specified by the user as a hyperparameter. The equation for the splitting criteria *Information Gain*, used in Random Forest is:

$$IG(S, F) = H(S) - H(S|F) \tag{4}$$

where:
S is the set of data points at the current node
F is a feature that is being considered for splitting
$H(S)$ is the entropy of the set S
$H(S—F)$ is the conditional entropy of S given F
This approach can help reduce over-fitting and improve the generalization of the model. Since the entire time series data is transformed into a concatenation of temporal and spatial features, using a Machine Learning for final classification would extensively reduce time complexity and would also produce highly efficient results. We have used the Random Forest classifier because it can easily provide estimates of feature importance and can handle large datasets with high dimensionality.
We tried four other Machine Learning models: XGboost, SVM, K-Nearest Neighbours and Naive Bayes for classification. But Random Forest classifier yielded the highest accuracy.

V. Experimentation and analysis

A. Datasets

For experimentation, we have used the Wafer dataset from the UCR repository. The Wafer dataset is a time series classification dataset available in the UCR ('University of California, Riverside') Time Series classification archive. It consists of time series data representing the measurements of a single wafer in a semiconductor manufacturing process. The dataset includes 7,144 time series, each with 152 data points. The training set has 1000 time series and test set has 6144 time series. The time series represent the measurements taken from a single sensor during the manufacturing process. The measurements were taken at regular intervals, resulting in a time series of sensor readings. The motto of using the Wafer dataset is to classify whether each wafer is 'normal' or 'abnormal' based on the time series data. The dataset is imbalanced, with

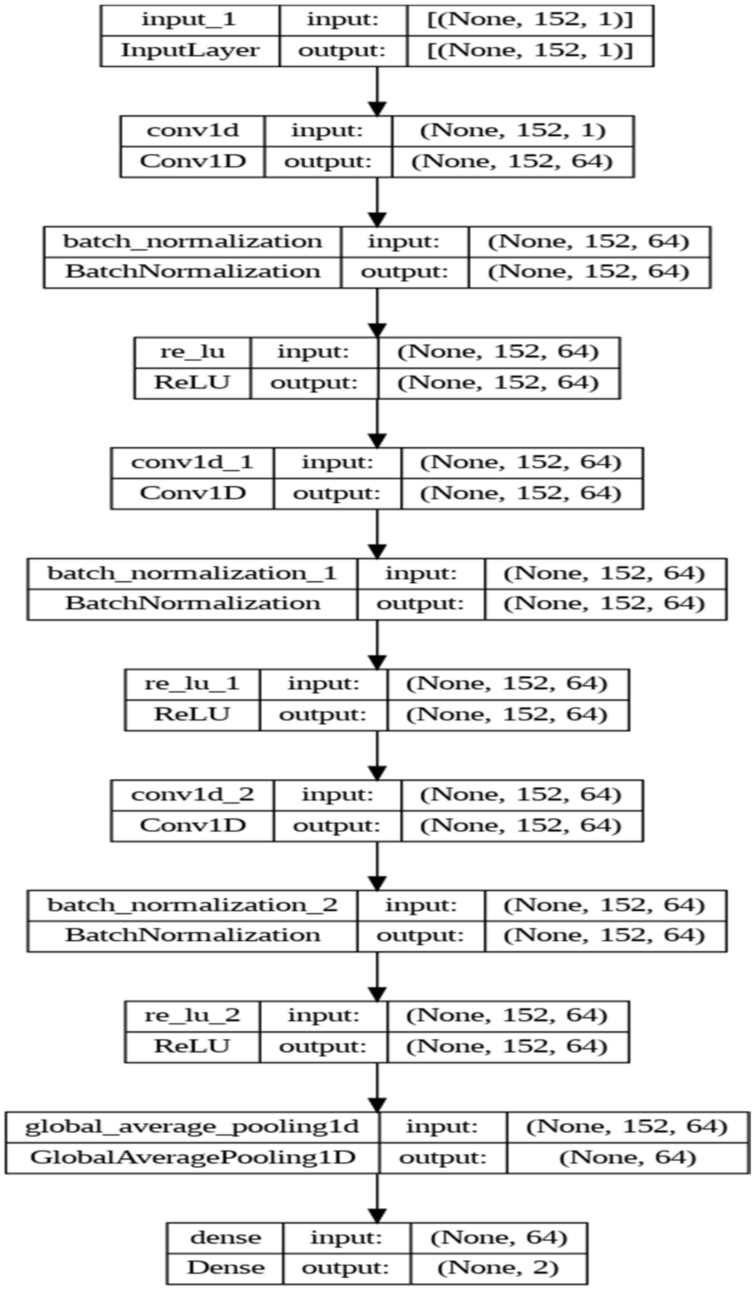
input_1 | input: | [(None, 152, 1)]
InputLayer | output: | [(None, 152, 1)]
conv1d | input: | (None, 152, 1)
Conv1D | output: | (None, 152, 64)
batch_normalization | input: | (None, 152, 64)
BatchNormalization | output: | (None, 152, 64)
re_lu | input: | (None, 152, 64)
ReLU | output: | (None, 152, 64)
conv1d_1 | input: | (None, 152, 64)
Conv1D | output: | (None, 152, 64)
batch_normalization_1 | input: | (None, 152, 64)
BatchNormalization | output: | (None, 152, 64)
re_lu_1 | input: | (None, 152, 64)
ReLU | output: | (None, 152, 64)
conv1d_2 | input: | (None, 152, 64)
Conv1D | output: | (None, 152, 64)
batch_normalization_2 | input: | (None, 152, 64)
BatchNormalization | output: | (None, 152, 64)
re_lu_2 | input: | (None, 152, 64)
ReLU | output: | (None, 152, 64)
global_average_pooling1d | input: | (None, 152, 64)
GlobalAveragePooling1D | output: | (None, 64)
dense | input: | (None, 64)
Dense | output: | (None, 2)

Fig. 3. Architecture of the CNN model

approximately 4,000 “normal” wafers and 3,144 “abnormal” wafers. The Wafer dataset is challenging due to the ‘high dimensionality’ of the time series data and the imbalanced class distribution. It is often used as a benchmark dataset for evaluating the performance of time series classification algorithms, particularly in the context of imbalanced datasets.

B. Results

The results of the experiment are given in Table 1. The proffered framework yields an accuracy of 99.86 % in contrast to the naive LSTM model which yields 96.9976 % and the CNN framework produces 98.670 %. From Fig. 4, it is clearly evident that Precision, Recall and F1 scores are also significantly high for the proposed framework. Out of the 6144 instances used for testing, 6057 instances were classified correctly by the proposed framework, whereas 5810 instances were classified correctly by the CNN framework and 5959 instances were classified correctly by the naive LSTM baseline model. It is clearly evident that the proposed framework transcends the other baseline models for all the datasets considered for evaluation, owing to its ability to capture the long-term temporal dependencies and spatial sparsity simultaneously. Whereas, the naive LSTM baseline model can record the long-term temporal dependencies alone and the CNN baseline

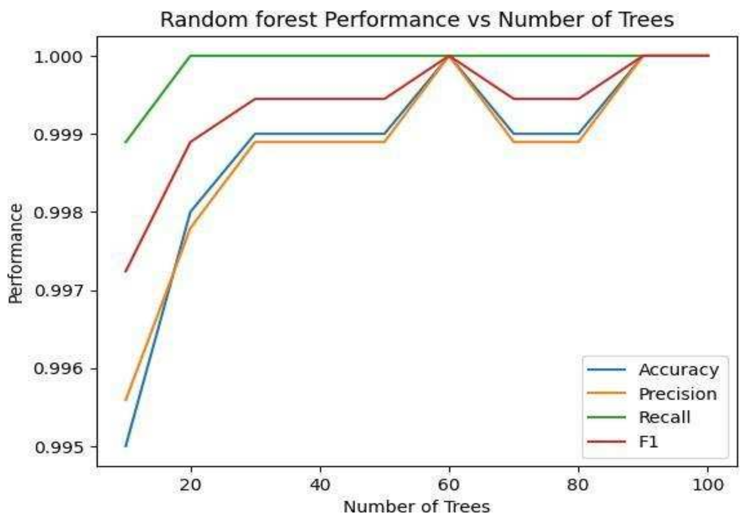

Fig. 4. Performance of the Random forest classifier vs Number of trees

TABLE I

ACCURACY PERCENTAGES FOR THE WAFER DATASET (UCI REPOSITORY)

LSTM	CNN	Attention-based LSTM-CNN
96.9976	98.675	99.86

framework can capture the spatial sparsity only.

VI. CONCLUSION

Time series classification is significant in everyday life because it enables us to make predictions and decisions based on patterns and trends in data that change over time. Time series data has various applications from financial forecasting to medical diagnosis to weather forecasting, and beyond. In this work, we focused on developing an ensemble of an attention-based LSTM and a CNN model to obtain the spatio-temporal features and then feed them into a Random Forest classifier to obtain the final output classes. Our framework yielded an accuracy of 99.86 %, transcending the baseline models. In our future work, we intend to incorporate attention mechanism in the CNN model also to deal with variable length input time series and extract spatial features effectively.

REFERENCES

[1] Mahato, V., O'Reilly, M., Cunningham, P. (2018). A Comparison of k-NN Methods for Time Series Classification and Regression. In AICS (pp. 102-113).

[2] Kim, M. (2013). Semi-supervised learning of hidden conditional random fields for time-series classification. Neurocomputing, 119, 339-349.

[3] Antonucci, A., De Rosa, R., Giusti, A., Cuzzolin, F. (2015). Robust classification of multivariate time series by imprecise hidden Markov models. International Journal of Approximate Reasoning, 56, 249-263.

[4] Ji, C., Zhao, C., Liu, S., Yang, C., Pan, L., Wu, L., Meng, X. (2019). A fast shapelet selection algorithm for time series classification. Computer networks, 148, 231-240.

[5] Deng, H., Runger, G., Tuv, E., Vladimir, M. (2013). A time series forest for classification and feature extraction. Information Sciences, 239, 142-153.

[6] Middlehurst, M., Large, J., Cawley, G., Bagnall, A. (2021). The temporal dictionary ensemble (TDE) classifier for time series classi- fication. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I (pp. 660-676). Springer International Publishing.

[7] Wang, Z., Yan, W., Oates, T. (2017, May). Time series classification from scratch with deep neural networks: A strong baseline. In 2017 International joint conference on neural networks (IJCNN) (pp. 1578-1585). IEEE.

[8] Karim, F., Majumdar, S., Darabi, H., Harford, S. (2019). Multivariate LSTM-FCNs for time series classification. Neural networks, 116, 237-245.

[9] Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., ... Petitjean, F. (2020). Inceptiontime: Finding alexnet for time series classification. Data Mining and Knowledge Discovery, 34(6), 1936-1962.

[10] Du, Q., Gu, W., Zhang, L., Huang, S. L. (2018, November). Attention-based LSTM-CNNs for time-series classification. In Proceedings of the 16th ACM conference on embedded networked sensor systems (pp. 410-411).

[11] Tang, W., Long, G., Liu, L., Zhou, T., Blumenstein, M., Jiang, J. (2020). Omni-Scale CNNs: a simple and effective kernel size configuration for time series classification. arXiv preprint arXiv:2002.10061.

[12] Malhotra, P., TV, V., Vig, L., Agarwal, P., Shroff, G. (2017). TimeNet: Pre-trained deep recurrent neural network for time series classification. arXiv preprint arXiv:1706.08838.

[13] Turbe´, H., Bjelogrlic, M., Lovis, C., Mengaldo, G. (2023). Evaluation of post-hoc interpretability methods in time-series classification. Nature Machine Intelligence, 1-11.