Biomedical Signal Processing and Control 84 (2023) 104754

Contents lists available at ScienceDirect

Biomedical Signal Processing and Control

journal homepage: www.elsevier.com/locate/bspc

ApneaNet: A hybrid 1DCNN-LSTM architecture for detection of Obstructive Sleep Apnea using digitized ECG signals

Gaurav Srivastava [a], Aninditaa Chauhan [a], Nitigya Kargeti [a], Nitesh Pradhan [a,*], Vijaypal Singh Dhaka [b]

[a] Department of Computer Science and Engineering, Manipal University Jaipur, Rajasthan, India
[b] Department of Computer and Communication Engineering, Manipal University Jaipur, Rajasthan, India

ARTICLE INFO

Dataset link: https://physionet.org/content/apnea-ecg/1.0.0/

Keywords:
Obstructive Sleep Apnea
ECG signals
Deep feature extraction
Convolutional neural networks
Long short-term memory
Alexnet
ApneaNet

ABSTRACT

Obstructive Sleep Apnea is a respiratory disorder that can be the origin of fatal heart and neurological health concerns if left untreated. Despite the availability of diagnosis methods, it is still undiagnosed in most cases due to the tiresome and impractical process of Polysomnography, the current medical standard test. In this study, the authors have worked towards finding a viable approach for easy and early diagnosis of sleep apnea using Electrocardiogram signals. The first model of this work was adapted from the Alexnet architecture with modifications done according to the input digitized signals. A Long–Short term memory layer was added to take care of temporal dependency in the dataset. It has shown an accuracy of 90.87%, specificity of 83.43%, and sensitivity of 95.48%. The hybrid architecture has 1.7 million parameters, much less than the Traditional Alexnet architecture. The second model, ApneaNet, has been introduced, which shows remarkable performance with an accuracy of 90.13%, specificity of 82.06%, and sensitivity of 95.14%, using only 0.9 million parameters which reduce the computational power significantly. The Proposed models have been implemented on a dataset split into 35 recordings for training and testing, showing a trailblazing accuracy of 95.69% and 96.37%, respectively. The authors proposed two deep learning models to detect sleep apnea events using Electrocardiography signals which have demonstrated competitive results compared to the state-of-the-art models at a low computational cost. We believe these methods have the potential to be successfully and efficiently used for the real-time detection of Sleep Apnea.

1. Introduction

Obstructive Sleep Apnea (OSA) is one of the most prevalent types of Sleep Apnea (SA). It is a potentially severe respiratory disorder that intrudes the natural cycle of breathing in a person [1]. Caused by limited or full impediments to the upper respiratory tract [2,3]. It is not uncommon to see multiple SA-related symptoms and events occur during a single Sleep cycle. In severe cases, it is known to cause brain defects and other heart issues. Apart from these, SA also results in higher blood pressure and congestive heart failure (CHF) and affects the cognition of a person [4].

Other Symptoms caused by this syndrome are chronic snoring, insomnia, gasping and breath-holding, unrefreshing sleep, and daytime sleepiness [5]. The diagnosis of SA is quantified by a system known as Apnea/Hypopnea index (AHI), and any patient scoring a score > 5 in addition to additive symptoms is clinically diagnosed with the syndrome. [6,7]. One of the most decisive and often used tests for SA is Polysomnography (PSG). The PSG process analyzes by keeping the test subjects in a laboratory for one or two sleep cycles and records many physiological signals like ECG, Respiratory signals, EEG, SpO2, etc. [8]. In addition, it also requires the patient to sleep in an actual laboratory for 8 h a day and even multiple days to detect, which can be cumbersome and uncomfortable for the patient leading to incorrect data.

The results from PSG are fairly decisive and comprehensive. Still, it usually creates an uncomfortable experience which causes irrational exceptions in the data traced further, resulting in lowered accuracy of the test. In addition, a PSG test requires a specific laboratory with state-of-the-art medical machinery, which further expands capital costs to the hospital/medical facility and the patient in question. Thus to ease this process of diagnostics, many studies have been done taking many signals as parameters like respiratory sound [9], SpO2 [10], ECG [11]. These studies have given good results with deep learning models. However, with a few variations, the proposed study further improves.

* Corresponding author.
E-mail addresses: mailto.gaurav2001@gmail.com (G. Srivastava), pradhan.nitesh943@gmail.com (N. Pradhan).

https://doi.org/10.1016/j.bspc.2023.104754
Received 16 April 2022; Received in revised form 4 January 2023; Accepted 18 February 2023
Available online 3 March 2023

Sleep Apnea is often prevalent amongst adults, most commonly middle-aged, but cases are also documented of children with ages ranging from 2–6 years. Apart from the reduction in the quality of life, another issue is that approximately 15%–20% of patients are medically diagnosed with the condition. Without proper diagnosis and treatment can lead to an increased risk and cause complications in cardiovascular activities. The author's motivation was to provide a simplistic yet accurate solution towards increasing the number of adequately diagnosed patients with genuine treatment, improving their quality of life, and reducing heart complications in the future.

Deep learning is a sub-branch of Machine learning, which is concerned completely with neural networks (inspired by the working and functioning of neurons in our brain) [12,13]. Deep learning techniques have been on a boost in the past few years due to their ability to learn without help and show efficient results like never before. [14,15]. Deep learning techniques use the backpropagation method to adjust its internal parameters as necessary, revealing detailed structure in large data sets [13].

Convolution Neural Networks (CNNs) has demonstrated remarkable results in medical imaging [16–18]. CNNs have also been used to extract deep features from ECG signals, and since the ECG signals have a temporal dependency, combining a CNN with LSTM can significantly boost performance. With this intuition, the authors modified the traditional pre-trained network to work with 1-dimensional ECG data. Then the authors experimented with various networks to determine the best-suited network for the specific dataset. Furthermore, keeping in mind the computational cost, the authors have also proposed a custom architecture — ApneaNet, which demonstrated remarkable performance in terms of accuracy and inference time on the Sleep Apnea ECG database. In addition, Convolutional Neural network based methods have shown significant performance even on small datasets. Research in this field for medical imaging has proven to give a state of the art results in both accuracy and efficiency, thus justified as a foundation of our research and study [19–22].

In this manuscript, the authors used deep learning approaches to tackle this problem since it has been shown that deep learning produces excellent results regarding disease detection. Furthermore, achieving competitive accuracy while ensuring that the computational cost stays reasonable was the primary driver for developing the custom model architectures. The authors aim to utilize various Deep learning techniques for the easy and efficient detection of Obstructive Sleep Apnea by analyzing previously used methods and experimenting to produce models with the best results.

The main contributions of this study can be summarized as follows:

1. The authors have experimented with and examined various pre-trained networks such as AlexNet, Xception, ResNet50, and DenseNet121 to extract features from ECG signals. Since the networks are trained for the Image dataset, the authors have modified their architecture in accordance with the 1-dimensional dataset to use these networks on the SleepApnea ECG dataset.
2. After experimentation and analysis, the authors proposed a hybrid 1-dimensional CNN-LSTM model based on Alexnet's architecture. The addition of the LSTM layer for sequential learning has shown significant improvements in terms of accuracy.
3. The authors proposed a custom Deep CNN and LSTM-based model — ApneaNet consisting of 13 layers taking efficiency into account. ApneaNet showed significant improvement in the computational cost compared to the other models balancing the accuracy simultaneously.
4. To prove the effectiveness of the proposed model, a thorough experimental assessment has been carried out, and the proposed model is tested with several different hyper-parameters. The authors experimented with different optimizers and inferred that Adam and Adamax had shown the best performance on most models compared to the other optimizers.

The remaining contents of the study can be summarized as follows. Section 2 is dedicated to analyzing the previous work of various scholars in detecting Apnea events. Section 3 dives into the various deep learning architectures examined and used for the study. Section 4 deals with the various Materials and methods and entails proposed algorithms. Section 5 discloses and covers experimentation and the results found from the aforementioned experiments.

2. Related works

OSA, a prevalent disorder in today's time, has taken its fair share of inputs towards its early-on diagnosis leading to uncomplicated cures. To fully fathom the problem, the authors undertook an extensive analysis of the significant contributions made by various medical experts, scientists, and scholars. As per the literature study, the authors used two types of classification identical to the dataset: per-segment and per-recording classifiers. In per-segment classification, each ECG recording is studied minute by minute to ensure robustness and accuracy and to validate the model, whereas in per recording, an entire recording is split into 1 min ECG signals for the same. These insights and studies were fruitful for our research as we gathered intuition and first-hand approaches to the task and helped us propose an improved model for its diagnosis. The works analyzed in this study are briefly explained in the succeeding paragraph.

Using single-lead ECG data, the author suggested a technique based on a Deep neural network (DNN) and Hidden Markov model (HMM) [2]. SVM and ANN were the two types of classifiers employed. Given the temporal dependence, HMM was utilized to increase classification accuracy. Finally, a decision fusion approach was applied to improve classification performance. The outputs of (DNN, SVM, HMM) and (DNN, ANN, and HMM) are referred to as Decision Fusion. In the per-segment OSA detection, almost 85% classification accuracy is achieved, with a sensitivity of up to 88.9%. A Multiscale Dilation Attention and 1 Dimensional Convolution Neural Network [23] was used to extract features. A weighted loss time-dependent loss function was applied to solve the data imbalance problem. Furthermore, because there is a temporal dependency between segments, the inclusion of HMM boosted the Classifier's performance. For per-segment OSA detection, the suggested technique has an accuracy, sensitivity, and specificity of 89.4%, 89.8%, and 89.1%, respectively. The Hidden Markov model [7] is coupled with other classification machine learning models, SVM, LR, LDA, and KNN, to boost classification accuracy. Per segment, OSA detection accuracy was 86.2%, while per recording, classification accuracy was 97.1%. S.Thompson [24] uses convolutional, max-pooling layers and a fully linked Multilayer Perceptron (MLP) with a hidden layer and SoftMax classifier to create a 1 Dimensional CNN model. The model has been trained on five distinct datasets with varied window widths of 500, 1000, 1500, 2000, and 2500. For window sizes 500, 1000, 1500, 2000, and 2500, the accuracy obtained was 93.77%, 95.28%, 91.61%, 90.86%, and 90.46%, respectively.

Further, Wang T [4] uses modified LeNet-5 CNN architecture to detect Sleep Apnea. Data preprocessing was done using a Hamilton Algorithm to find the R Peaks and R-R Intervals. For per segment, the specificity was 90.3%, the sensitivity was 83.1%, the accuracy was 87.6%, and the AUC was 0.950. Per recording, accuracy was 97.1%; sensitivity was 100%, specificity was 91.7%, AUC was 0.996, and Corr was 0.943. Mukherjee [25] uses three deep learning models to employ different ensemble techniques which two were CNN-based models, and one was a hybrid model of CNN and LSTM that had previously been suggested in the OSA detection area. Following that, four ensemble approaches were chosen: majority voting, sum rule, Choquet integral-based fuzzy fusion, and trainable ensemble using Multi-Layer Perceptron (MLP). Using the MLP-based ensemble technique, the highest OSA detection accuracy of 85.58% was achieved. Using single-lead electrocardiogram (ECG) inputs, A scalogram-based convolutional neural network (SCNN) [26] was suggested to identify obstructive sleep apnea. These scalograms were used to train a lightweight CNN model that uses 32 x 32 scalograms as input to extract deep features for OSA detection. In per-segment classification, an accuracy of 94.30%, a sensitivity of 94.30%, a specificity of 94.51%, and an F1-score of 95.85% was reached. For the UCDDB dataset, the model achieved an accuracy of 81.86%, a sensitivity of 71.62%, a specificity of 86.05%, and an F1-score of 69.63%.

Additionally, Chang XY [27] uses a one-dimensional CNN model consisting of ten similar CNN-based feature extraction layers, a flattening layer, four identical classification layers (mostly built of fully connected networks), and a softmax classification layer. For per-minute apnea detection, the suggested model obtains 87.9% accuracy, 92.0% specificity, and 81.1% sensitivity, while for per-recording classification, it achieves 97.1% accuracy, 100% specificity, and 95.7% sensitivity. Junming Zhang [28] proposed an automated OSA event detection method using a convolution neural network. A long short-term memory (LSTM) was added to learn the long-term dependencies, such as the OSA transition rules. A 10-second overlapping sliding window segments the raw ECG signals to detect a completed OSA event. The proposed model exhibits the results: Cohen's kappa coefficient of 0.92, a sensitivity of 96.1%, a specificity of 96.2%, and an accuracy of 96.1% concerning the Apnea-ECG dataset. Wang L [29] introduced RR intervals with a residual network to detect per-segment apnea. A novel approach, dynamic autoregressive representation, was introduced to represent the RR intervals by convolutional layers. A residual network with 31 residual blocks is implemented in this study, and a basic CNN with seven convolutional layers is also implemented for comparison. The residual network shows a performance of 94.4% accuracy, 93.0% sensitivity, and 94.9% specificity for per-segment apnea detection. A 10-fold cross-validation process and then a blind-fold validation for evaluation were conducted to check the robustness of the proposed model. The influence of input data length was analyzed, and it concluded that the model required data with longer input length to get better results; 3-min NN intervals were suggested.

The other successful works related to the domain includes the study conducted by Upadhyay et al. [30] was conducted to detect and classify EEG states to evaluate the effects of stress caused by external heat. ML algorithms such as Support Vector Machine (SVM) and Radial Basis Function Neural Networks (RBFNN) were used for the same. The results after experimentation favored SVM over RBFNN, with the SVM model showing an accuracy of 96.4% for chronic stress and 94.1% for acute stress. The RBFNN model showed an overall classification accuracy of 87%. This study supports the development of effective models for detecting stress levels in humans using the EEG power spectrum. The interesting work by Panda et al. [31] used hyperspectral image processing for Chronic Myeloid Leukemia (CML) detection by analyzing statistical distances such as Euclidean and Mahalanobis distance methods to classify neutrophils in CML from those in healthy blood samples. These distances were applied in the multidimensional space provided by hyperspectral images, which the authors state that has been used for the first time in this case. Principal component analysis was utilized to reduce dimensionality for computing efficiently. After experimentation, Euclidean distance was shown to be better in sensitivity for detecting CML neutrophils, while Mahalanobis distance was better for detecting healthy neutrophils. Borlea et al. [32] proposed a novel clustering technique — Unified Form that treats K-Means and Fuzzy C-Means algorithms as a single algorithm, with the software implementation solutions designed and validated for the same on the BigTim platform. The other novelty feature was the partitional implementation of the UF algorithm, which is designed to solve the issues with the processing of huge datasets, with the ability to overcome the hardware limitations with scaling big data. For future work, the authors aim to extend the work to clustering algorithms, including nonlinear functionalities specific to fuzzy sets that will be considered for performance improvement.

In the author's analysis of these distinctive approaches to diagnosing the disorder, however, we saw a striking pattern: an accuracy-parameter tradeoff with each method. Models giving higher accuracy performance possessed an exorbitant amount of parameters required, while the models with lesser and more efficient parameterizing hurt the accuracy of the proposed model. The authors further delved into analyzing various architectures to maximize accuracy and minimize parameters, as covered in the next section (Section 3). It also helped the authors to propose a novel approach based on the presented architectures.

3. Background of networks

Deep learning techniques are widely employed in effectively processing and identifying patterns from datasets [13]. Even with present-day technological advances, it is expensive in both computing and time requirements for constructing the models to solve the problems at hand. In addition, the tremendous leaps in the skill they impart on similar problems. Therefore It is a standard approach to use pre-trained models as the initial template for computer vision and natural language processing tasks; such a technique is called transfer learning [33]. There are many pre-trained models available nowadays which are trained on the Imagenet dataset [34] with 1000 output classes for classification problems. A pre-trained model [35] has previously been taught to deal with a similar problem. The authors start with a previously trained model on another problem rather than starting from the ground up to solve a homogeneous problem.

The traditional neural network architectures were designed to solve the problem of image classification. However, the time-series data used in our study is one-dimensional, unlike two-dimensional image recognition problems. So we have adopted the neural network architecture of traditional Models, modified it according to our dataset requirements, and used it in our study.

With the fast advancement of deep learning [13], a slew of new neural network designs have emerged to tackle a wide range of jobs and issues. Although there are countless neural network architectures, the authors have used the following architectures for their study.

3.1. Alexnet

AlexNet can be defined as a convolutional neural network that consists of convolutions, dense layers, and max-pooling as the basic building blocks of its architecture [36]. The model consists of 8 layers, with five convolutional layers combined with max-pooling layers, followed by three fully connected layers [37]. The input for the model is RGB images. The activation function used in all the layers is ReLU, and that in the output layer is Softmax. AlexNet uses Rectified Linear Units (ReLU) because a neural network using the ReLU function had shown a 25% error on the CIFAR-10 dataset, about six times faster than one using the tanh function. AlexNet also grants multi-GPU training by segregating one-half of the model's neurons on a GPU and the other half on another GPU. This helps optimize the model, improve its performance, and reduce training time. A significant problem in AlexNet can be overfitting due to the considerable number of parameters used [38]. Data augmentation and dropout layers are used to reduce this effect.This problem of overfitting is commonly solved using the above two strategies. Augmentation of data to increase the dataset size or reduce the number of parameters using Dropout. In Dropout, to introduce variations in the network, a predefined amount of neurons are rendered inactive and void, thus reducing the number of parameters used in training the model and reducing overfitting. By introducing overlap, a reduction of error by 0.5% was observed, and overlapping pooling makes the model less prone to overfit.

3.2. *Xception*

Xception comprises 71 layers of deep learning architecture that can be defined as a deep convolutional neural network that has taken inspiration from the architecture; Inception [39]. However, the replacement of the Inception modules takes place by depthwise separable convolutions. It has an input size of 299 by 299. Separable convolutional layers are more advantageous than traditional ones for optimal memory and computational usage. This is because the standard convolution transforms the image multiple times which takes up computational power [40]. In Separable convolutions performance of depthwise convolutions followed by pointwise convolutions is employed relating to a mix in the resulting output channels. While the implementation of the same used in Xception, is the inverse of the original method defined. It follows depthwise convolutions after a pointwise convolution. It changes the order in which the operations take place. This minor change makes it so that there is no intermediate ReLU Nonlinearity. The Xception architecture [41] slightly outperforms Inception V3 [42] on the ImageNet [34] dataset (which it was designed for) and shows much better performance on a larger image classification dataset, i.e., JFT dataset.

3.3. *ResNet50*

ResNet, also known as Residual Networks, is a classic neural network used for various Computer Vision tasks [43]. A residual neural network is an artificial neural network that stacks residual blocks on top of each other to form a network. Resnet50, used in our research, is the variant that can work with 50 neural network layers and 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer [44]. This network was first introduced in the 2015 computer vision research paper titled ‘Deep Residual Learning for Image Recognition’ [43]. The breakthrough with ResNet was that they allowed the training of extremely deep neural networks efficiently. Before ResNet, such networks were difficult to train due to the problem of vanishing gradients [45]. That is, when the gradient is back-propagated, it can become extremely small due to repeated multiplication. This causes saturation or even degradation of performance in networks with many layers. ResNet tackles this problem by using identity mapping layers and skip connection to add output from a previous layer to a later one [46].

3.4. *DenseNet121*

DenseNet, or Densely Connected Convolutional Network, is a Deep Learning model. It is based on upfront connections between every successive layer while keeping the feature map size indifferent [47]. The model’s strength lies in the number of layers that can be increased significantly, causing no harm to the optimization of the architecture. The model relies on fewer parameters than conventional convolutional Neural Networks (CNN) with more connections. It features 4 Blocks, namely Dense Blocks, Transition layers, Convolution layers, and Pooling layers.

In the proposed study, the authors used a specific model of DenseNet, the DenseNet-121, which comprises : 1 * 7×7 Convolution, 58 * 3×3 Convolutions, 61 * 1×1 Convolution, 4 Average pooling Layers, and 1 Fully Connected Layer, which cumulates a total of 120 Convolutions and 4 Average pooling Layer [47]. In brevity, the network has [6,12,24,16] dense layers, and Each dense block possesses a lineup of units. Every unit consists of 2 convolutions processed by Batch Normalization and ReLU activations. Each unit has a constant amount of feature vectors, and the parameter is known as the growth rate. The growth rate defines the transmission of new data in a layer. A layer between the dense blocks also performs a down-sampling of the features [48].

Table 1
Dataset description.

Subject recordings	ECG files	Group type	Apnea events (Mins)	Non-Apnea events (Mins)
A01–A20	20	Apnea set	6250	3811
B01–B05	5	Borderline Set	252	2060
C01–C10	10	Normal Set	12 6514	4740 10611

Table 2
Distribution of dataset.

	Normal event	Apnea event
Training Set (35)	210680	130050
Testing Set (35)	213830	13102

4. Materials and methods

4.1. *Dataset formation*

In this collection, the authors possess a total of 70 records split into a 50% split between training and test (35 each). Each recording is variable in length and ranges from 7 h to an estimated 10 h. Per recording comprises a constant ECG signal converted to a digital format. Apart from that, auxiliary signals are also considered, like oronasal airflow and oxygen saturation [49]. The ECG sample rate was measured at 100 Hz with a resolution rate of 12 bit. Every recording is further subclassified into 1 min segments upon which, if abnormal patterns arise, common apnea cases are labeled.

In addition, another type of classification occurred based on the Apnea-Hypopnea Index (AHI) as three distinct classes, namely A, B, and C, respectively, as mentioned in Table 1. The distribution of all apnea and normal events in the training and test sets is shown in Table 2.

4.2. *Dataset preprocessing*

The dataset considered is the Physio-Net Sleep Apnea Dataset [49]. A logical approach towards autonomous feature extraction from the dataset of 70 PSG recordings with quality Respiratory and ECG signals in terms of RR intervals and amplitudes. This process has been illustrated in Fig. 2. T.Penzel [50] states that Preceding contributions showed that variable heart rate does not quantify to give a good algorithm; therefore, 1-minute segments are extracted for preprocessing the dataset. R wave peak finding was implemented using Hamilton’s as described in [51]. The Newly found R peaks were used to calculate the Interval gap between the two consecutive R waves (RR Interval). However, the RR intervals extracted are diagnostically obscure and can hamper the training of the proposed model. To counter this, a median filter was used [52]. Median Filter is a preprocessing technique to weed out noise or unwanted artifacts from the data to give a smoother and more accurate scenario that the data represents. A local filter is used to curb the spikes and deviating artifacts while processing R-R Intervals. The Fig. 1 shows the difference between the noise levels present in an ECG Sample. Cubic interpolation was deployed to normalize the time difference between RR intervals to give us the processed data ready for training.

4.3. *Training parameters*

There is often a need for optimizing and improving deep learning models. The use of many techniques can do this. The methods and parameters most used for the best training performance of our models have been explained in detail in this section. These include the Cross-entropy loss function, Softmax activation function, and optimizers such as Adam and Adamax.

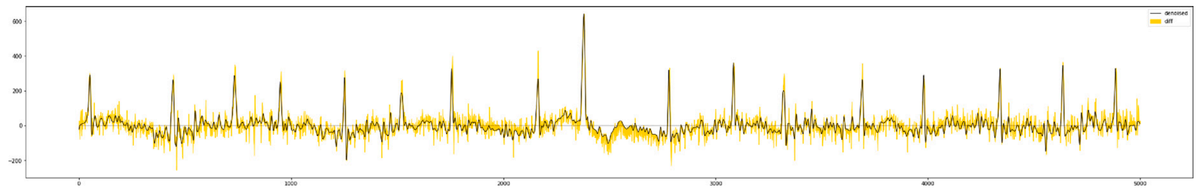


Fig. 1. Denoising using Local Median Filter in an ECG sample.

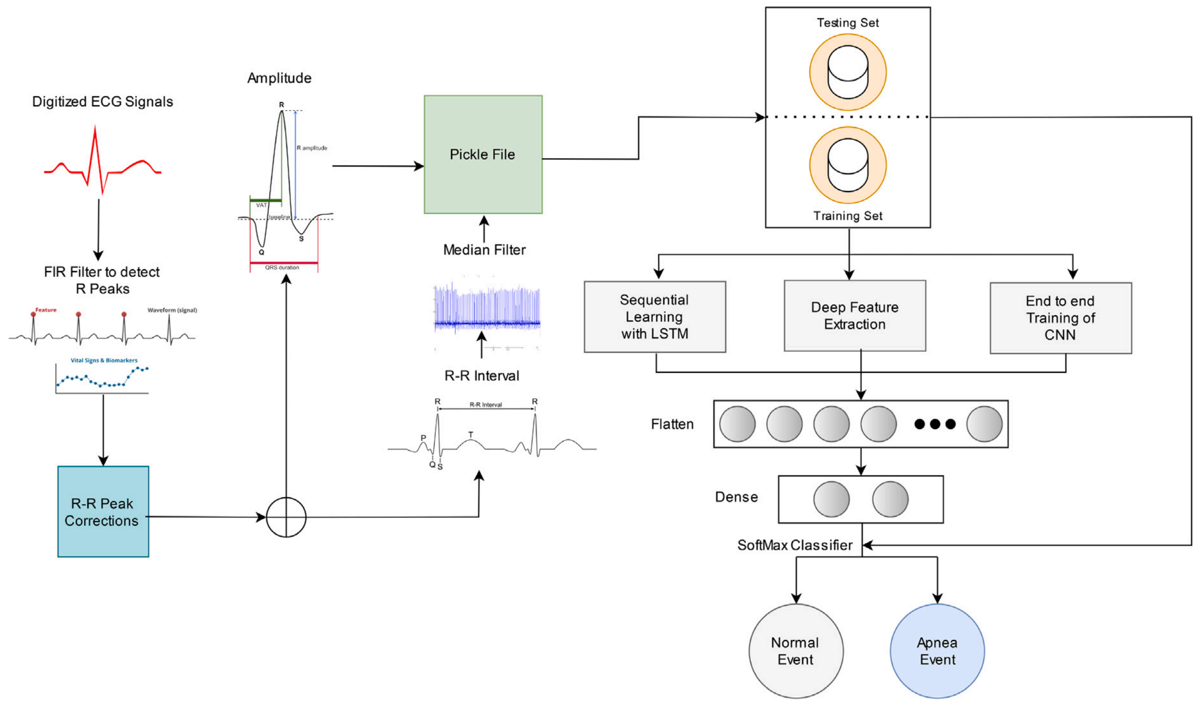


Fig. 2. Graphical abstract of the proposed work.

4.3.1. *Loss function*

While training machine learning models, loss or cost functions optimize the model's performance. The main objective is that the loss should be minimized; the lower the performance, the better. To optimize the classification model, the Authors implemented Cross-Entropy loss, the most widely used loss function. It is used to optimize classification models. The primary function of the Cross-Entropy is to use the output probabilities obtained from Softmax Activation to measure the distances from the truth values. Keras provides the following cross-entropy loss functions: binary, categorical and sparse categorical cross-entropy loss functions [53]. The following mathematical equation (1) explains the computation of the cross-entropy loss function:

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log\left(p_i\right), \text{ for n classes,} \tag{1}$$

where t_i is the truth label and p_i is the Softmax probability for the ith class.

4.3.2. *Optimizer*

This research uses two different optimizers, i.e., Adam and Adamax.

Adam: Adam optimization [54] is a stochastic gradient descent method that has its name derived from Adaptive Moment Estimation. This is because it uses estimations of the first and second moments of gradient descent to adapt and compute the individual learning rates for each network weight. It is inherited from RMSProp and AdaGrad and has their combined advantages. Adam optimization method is computationally very efficient as it requires less memory even when working with problems involving a large number of data and parameters. Adam [55] is intuitively a combination of RMSProp and Stochastic Gradient Descent with Momentum. It uses the moving average of the gradient in place of the gradient itself, like in SGD with momentum and it uses squared gradients to scale the learning rate, like in RMSProp. Adam optimizer uses an exponentially decaying average of past gradients (mt) and past squared gradients (vt) as defined in Eqs. (2) and (3), respectively. The term β_1 and β_2 are the forgetting factors for the mean and non-centered variance of the gradient, respectively.

$$m_t = \beta_1 m_{t-1} + \left(1-\beta_1\right)\left[\frac{\delta L}{\delta w_t}\right] \tag{2}$$

$$v_t = \beta_2 v_{t-1} + \left(1-\beta_2\right)\left[\frac{\delta L}{\delta w_t}\right]^2 \tag{3}$$

where,

1. ϵ = a small +ve constant to avoid 'division by 0 ' error when $\left(v_t - > 0\right) \cdot \left(10^{-8}\right)$
2. $\beta_1 \& \beta_2$ = decay rates of the average of gradients in the above two methods. $(\beta_1 = 0.9$ & $\beta_2 =$ 0.999)
3. $\boldsymbol{\alpha}$ - Step size parameter/learning rate (0.001)

Adamax: Adamax is an extension to the Adaptive Movement Estimation (Adam) optimization based on the infinity norm. Adamax has shown superior performance to Adam, especially in models with embeddings [56]. Adamax is an extension of the Adam version of gradient descent that generalizes the approach to infinity norm and is designed to accelerate the optimization process on some problems. Eq. (4) shows the mathematical representation of the Adamax optimizer:

$$x(t) = x(t-1) - (\alpha/(1-\beta_1(t)))^{*} m(t)/u(t) \tag{4}$$

To review, there are three hyperparameters as :

1. α: Initial step size (learning rate), a typical value is 0.001.
2. β_1: Decay factor for first momentum, a typical value is 0.9.
3. β_2: Decay factor for infinity norm, a typical value is 0.999.

An initial learning rate (α) of 0.001, the exponential decay rate for the first moment (β_1) as 0.9, the exponential decay rate for the second moment (β_2) as 0.999, and an epsilon value of 1e−7. These hyperparameters are selected utilizing the Grid search technique for model tuning and optimization. While training, the learning rate was varied continuously using the LRDecay, while the other hyperparameters remained constant.

4.3.3. Batch normalization

Batch normalization is a technique for training deep neural networks that standardize the input to a layer for each batch [57]. The input data collected for training is known as a batch; this normalization process takes place in batches and not as a single input. Normalization is a data pre-processing tool that brings numerical data to a common scale without distorting its essence.

Batch normalization adds new layers that make the neural networks efficient and more stable. Batch normalization layers allow every layer of the network to learn independently. It is primarily used to normalize the output of the previous layers, and it can be added at several points between the layers of the model. Batch normalization is also used as a regularization technique to overcome the problem of overfitting [58].

4.3.4. Activation function

In neural networks, the activation function specifies how the weighted sum of input is transformed into the output from nodes in a layer. The activation function for hidden layers is usually the same. However, depending on the prediction made by the model, the output layer would typically utilize a different activation function. Softmax is a mathematical function that converts a set of integers into a set of probabilities, with the probabilities proportional to the vector's relative values [59,60]. In Neural Network models, it is used as an activation function to normalize the outputs of each class, changing them from weighted sum values to probabilities that add up to one. The softmax function is defined in Eq. (5).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5}$$

where, σ = softmax, $\vec{z}$ = input vector, e^{z_i} = standard exponential function for input, K = number of classes in the multi-class, e^{z_j} = standard exponential function for output.

The sigmoid and tanh functions both saturate; high values snap to 1.0, and small ones snap to −1 or 0. It becomes difficult for the learning algorithm to continue adjusting the weights to enhance the model's performance once it reaches saturation. These non-linear activation functions fail to provide relevant gradient information to layers deep in extensive networks. Due to the vanishing gradient issue, deep (multi-layered) networks cannot successfully learn. Another non-linear activation function that has grown in prominence in the deep learning field is the ReLU function. Rectified Linear Unit is referred to as ReLU.

The ReLU function's primary benefit over other activation functions is that it does not simultaneously fire all the neurons. This signifies that the neurons will not stop firing until the linear transformation's result is less than 0. Thus, we used the Relu activation function in the hidden layers and the softmax activation function in the output layer. The softmax function returns a vector of values that add up to 1.0 and represent the probability of class membership. The argmax function, which returns 0 for all options and 1 for the selected option, is connected to this. A winner-take-all function may provide an output that resembles probability thanks to the "softer" form of argmax called softmax. A vector of real numbers serves as the function's input, and its output is a vector of the same length with values that add to 1.0, which is the probability.

4.4. Convolutional neural networks (CNNs)

Convolutional Neural networks [61,62] are increasingly being used in the field of Deep learning for performing image recognition and computer vision tasks that enable computers to extract meaningful information from images, videos, signals, and other forms of input, to predict their outcomes [63] successfully. They show superior performance with image and audio signal inputs [23] compared to other neural networks and have been used for the custom architecture in our research. Convolutional Neural networks have three main layers:

1. Convolutional layer: This is the network's core building block and does most of the computational work. The main components are input data, filter or kernel, and feature map. If the input data is a color image, it would be a matrix of pixels in 3D, corresponding to RGB. The feature detector, the filter, is passed over the receptive fields to check if the feature is present, known as convolution. The more filters affect the depth of the output. The filter then covers the image area, shifting by a stride(number of pixels) each time. This process is repeated until the entire image is covered. The final output obtained from the input and filter dot products is known as feature map [64].

In a 2D convolution process, we re-estimate the value at a specific input as a weighted average of inputs surrounding it. To acquire the re-estimated value of every pixel, we consider the value of its neighbors and compute the weighted average of these neighbors. This operation's mathematical formula is shown in Eq. (6).

$$S_{ij} = (I * K)_{ij} = \sum_{a=\left\lfloor -\frac{m}{2} \right\rfloor} \sum_{b=\left\lfloor -\frac{n}{2} \right\rfloor}^{\left\lfloor \frac{m}{2} \right\rfloor} I_{i-a,j-b} K \frac{m}{2} + a, \frac{n}{2} + b \tag{6}$$

where,

K-Matrix that represents the weights assigned to pixel values. It has two indices a,b - a denotes rows and b denotes columns.

I - Matrix containing the input pixel values.

$\mathrm{S_{ij}}$ - The re-estimated value of a pixel at a location.

Each convolutional layer has a filter ($m1$). The output $\mathrm{Y_i^l}$ of layer l consists of m_1^l feature map of with size $m_2^l \times m_3^l$. The i th feature map, $\mathrm{Y_i^l}$, is calculated on the bases of Eq. (7).

$$Y_i^{(1)} = f\left(B_i^{(l)} + \sum_{j=1}^{m_i^{(l-1)}} k_{i,j}^{(1)} \times Y_j^{(l-1)} \right) \tag{7}$$

where B_i^l demonstrates the bias matrix and $K_{i,j}^l$ the filter size.

In addition to CNNs, kernel convolution is a key component of various computer vision technologies. It is a technique in which we apply a small number matrix called a kernel or filter on our image, then transform it using the filter's values. This formula is used to compute feature map values as shown in Eq. (8).

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \tag{8}$$

where, f stands for the input image and h stands for our kernel. The result matrix's row and column indexes are denoted by m and n, respectively.

We had create an output feature map using a convolution method. In the convolution layer, each output feature map is blended with many input feature maps, as shown in Eq. (9).

$$x_j^l = f\left(\sum_{i \in M j} x_j^{l-1} * k_{ij}^l + b_j^l \right) \tag{9}$$

where, x_j^l is the current layer's output, x_j^{l-1} is the previous layer's output, k_{ij}^l is the current layer's kernel, and b_j^l are the current layer's biases. A collection of input maps is represented by M_j. The convolution results are then processed by a nonlinear activation function.

ReLU or Rectified Linear Unit is the non-linear activation function used here as shown in Eq. (10). ReLU does not activate all the neurons at the same time.

$$\text{ReLU} \quad \phi(x) = \begin{cases} 0 & x \le 0 \\ x & x > 0 \end{cases} \quad \phi'(x) = \begin{cases} 0 & x \le 0 \\ 1 & x > 0 \end{cases} \tag{10}$$

2. Pooling layer: By removing dominating characteristics, the pooling layer reduces the number of parameters and processing resources. A filter is used over the full input in the pooling procedure, but the filter does not contain any weights. There are two forms of pooling: max pooling (where the pixel with the highest value is delivered to the output) and average pooling (average value within the receptive field is sent to the output) [64].

The pooling layer is responsible for lowering the parameters by downsampling feature maps. It is applied throughout the layers in the 3D volume. A popular pooling layer, as shown in Eq. (11), employs a non-overlapping two cross two max filter with a stride of 2. A max filter would return the maximum value in the features inside the region.

$$x_j^l = \text{down}\left(x_j^{l-1}\right) \tag{11}$$

The hyperparameters within this layer are:

1. Dimensions of spacial extent: the value of n which could be taken for feature representation and mapped to a single value.
2. Stride: this is the number of horizontal and vertical steps that the filter takes over the original matrix.

Number of output features in each dimension can be calculated by use of the following formula, which has been explained in detail in Eq. (12).

$$n_{\text{out}} = \left[\frac{n_{\text{in}} + 2p - k}{s}\right] + 1 \tag{12}$$

where,
n_{in} : number of input features
n_{out} : number of output features
k : convolution kernel size
p : convolution padding size
s : convolution stride size

3. Fully Connected layer: In partially connected layers, the input data pixel values are not directly connected to the output layer. In contrast, in fully connected layers, each node in the preceding layer is directly connected to the output layer. While the preceding layers used the ReLu function to classify data, this layer used the softmax activation function to get a probabilistic result [64].

The working of fully connected layer can be represented by Eq. (13), if (l - 1) is a fully connected layer;

$$Y_i^{(1)} = f\left(Z_i^{(l)}\right) \text{ with } Z_i^{(l)} = \sum_{j=1}^{m_i^{(l-1)}} w_{i,j}^{(1)} \times Y_j^{(l-1)} \tag{13}$$

4.5. Modified network architectures

The one-dimensional data in our time series dataset [50] utilized in this work differed significantly from two-dimensional image classification problems, which is the input in all traditional Convolutional Neural Network Architectures. As a result, we had to modify our neural network topologies to make them suitable for the given RR interval input (900, 2). A one-dimensional convolution operation was used instead of a two-dimensional convolution operation to extract features. In the convolution layer, we have modified the proportions for strides of the convolution layer and kernel size in accordance with the one-dimensional convolution layer. The pool size and strides were also adjusted in accordance with the one-dimensional max-pooling layer. The long short-term memory (LSTM) layer [65] was added to the network so that the model can learn short-term dependencies as OSA transition rules. Internal memory is stored in an LSTM network which learns temporal information from segments of inputs through feedback connections. Furthermore, overfitting is possible due to the scarcity of data, so a dropout layer is also added. Finally, a softmax classifier with two output neurons is utilized, as shown in Fig. 2. In this research, the authors proposed two different Models - Modified 1D Alexnet + LSTM and ApneaNet with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and an epsilon value of 1e−07.

Algorithm 1: Training Procedure of the Proposed Model

```
Input:
$\delta 1$ : Sleep Apnea Training set
$\delta 2$ : Sleep Apnea Testing set
$d_i$ : the $i$th input sample of testing dataset.
$K$ : total number of classes.
$\mu$ : learning rate
$\epsilon$ : no. of epochs
$\beta$ : batch size
$\omega^*$ : initial weights of trained classifier.
Output :
Classification as Apnea Event or Normal Event
predicted class probability vector
begin;
1. Set the input layer of the CNN architecture and feed the input
   size
2. Set the head layers, CNN (Conv1D, Maxpooling1D, Flatten,
   Dense, Dropout)
3. Set the further layers as LSTM
4. Initialize the CNN parameters : $\mu$, $\epsilon$, $\beta$
5. Convert each input in the training set into 900 x 2 in
   accordance with the required input shape.
6. Train the CNN and compute the initial weights.
while $\eta = 1$ to $\epsilon$ do
    7. Randomly select a mini batch from (size : $\beta$) from training
       set ($\delta 1$)
    8. Forward propagation and compute the loss using Eq. (1)
    9. Back propagate the error and update the weights using
       Eq. (14) with adam optimizer
       $W_n = W_n - \eta * \frac{\partial J}{\partial W_n}$    (14)
    10. Repeat steps 7 to 9 until total loss become minimum.
end
while $i = 1$ to $\delta 2$ do
    11. Get the probabilistic output vector ($\mathbf{p}_m(d_i) = \{p_{m,k}(d_i)\}$
        for all $K$ classes where $1 \le k \le K$ ).
    12. $p_k(d_i) = \sum_{m=1}^{M} p_{m,k}(d_i) \cdot w_{i,k}$
    $c(d_i) = \arg\max_{1 \le k \le K}\{p_k(d_i)\}$
    Get the output class label with the maximum class
       probability for the $i$ th input sample.
end
```

4.5.1. Proposed model I - Modified 1D Alexnet + LSTM

The proposed hybrid 1DCNN-LSTM model is based on Alexnet's architecture and employs CNN layers for feature extraction, an LSTM layer for sequence learning, and a softmax classifier for binary classification. The proposed modified Alexnet Network architecture is shown in Fig. 3. The Alexnet designed by Alex Krizhevskya contained eight layers; the first five were convolutional layers, some followed by max-pooling layers, and the last three fully connected layers [36]. Alexnet

Table 3
Layerwise description of modified AlexNet.

Layer name	Layer type		Output shape	Parameters
input_1	Input Layer	900 x 2 time series data	[(None, 900, 2)]	0
conv1d	Conv1D	96 Filter, 4, S = 1, ReLU	(None, 223, 96)	2208
batch_norm	Batch Normalization		(None, 223, 96)	384
max_pooling1d	MaxPooling1D	pool size = 3, S = 2	(None, 111, 96)	0
conv1d_1	Conv1D	256 Filter, 5, S = 1, ReLU	(None, 111, 256)	123136
batch_norm_1	Batch Normalization		(None, 111, 256)	1024
max_pooling1d_1	MaxPooling1D	pool size = 3, S = 2	(None, 55, 256)	0
conv1d_2	Conv1D	384 Filter, 3, S = 1, ReLU	(None, 55, 384)	295296
batch_norm_2	Batch Normalization		(None, 55, 384)	1536
conv1d_3	Conv1D	384 Filter, 3, S = 1, ReLU	(None, 55, 384)	442752
batch_norm_3	Batch Normalization		(None, 55, 384)	1536
conv1d_4	Conv1D	256 Filter, 3, S = 1, ReLU	(None, 55, 256)	295168
batch_norm_4	Batch Normalization		(None, 55, 256)	1024
max_pooling1d_2	MaxPooling1D	pool size = 3, S = 2	(None, 27, 256)	0
lstm	LSTM	LSTM Layer of 256 units	(None, 256)	525312
dense	Dense	Dense Layer of 2 units	(None, 2)	514

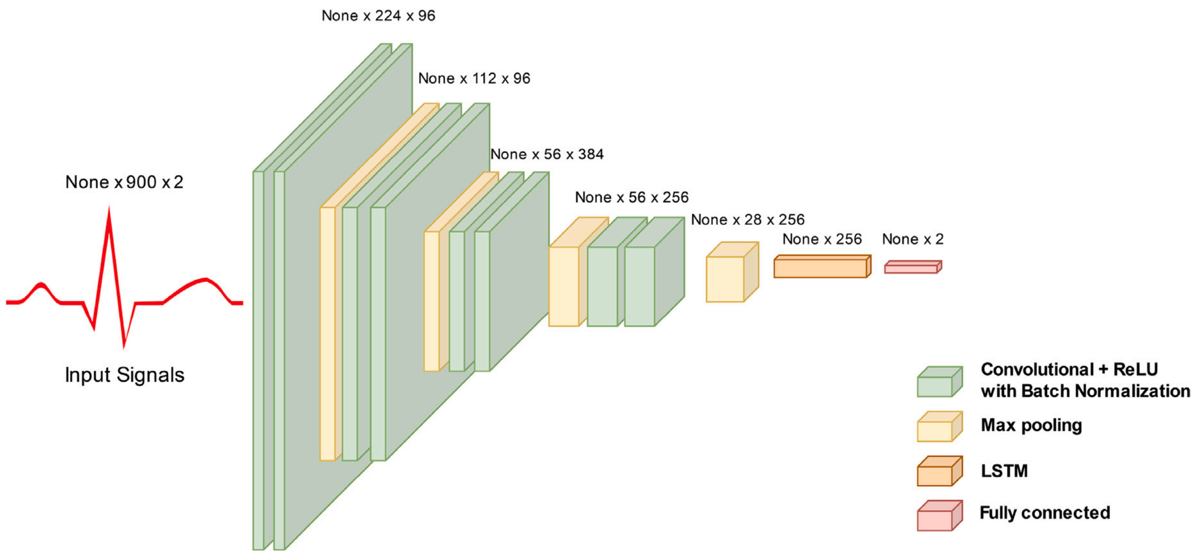


Fig. 3. 3D representation of the Modified AlexNet.

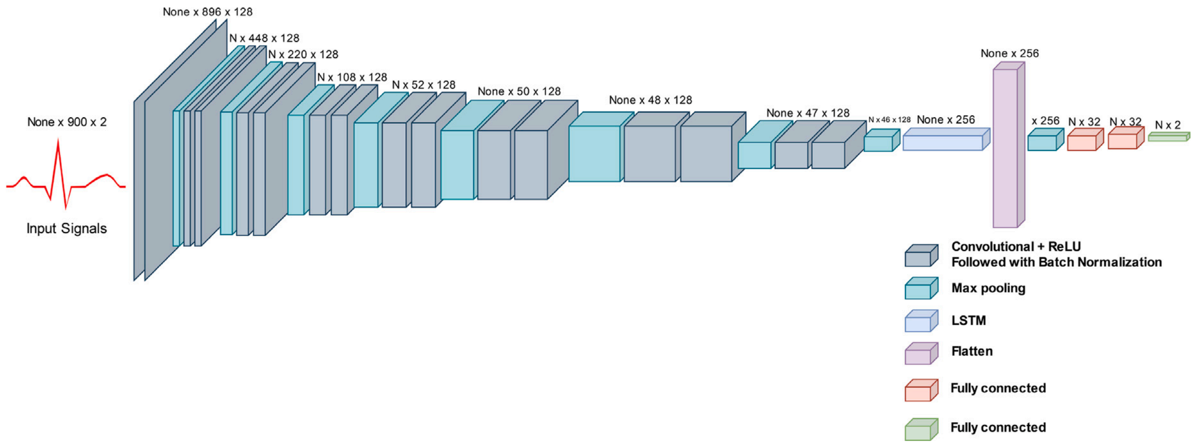


Fig. 4. 3D representation of the ApneaNet.

employs the non-saturating ReLU activation function. The original Alexnet architecture was designed for image data, but we had one-dimensional time series data in our study, thus, we made the following changes to use it for our research.

1. Modified all two-dimensional convolution layers to one-dimensional convolution layers by using Conv1D, changing strides and kernel size to a single digit number rather than using a tuple.
2. Modified all two-dimensional max pooling layers to one-dimensional max pooling layers by using Maxpooling1D, changing pool size and strides to a single digit number rather than using a tuple.

Table 4
Layerwise description of ApneaNet model.

Layer name	Layer type		Output shape	Parameters
input_1	Input Layer	900 x 2 time series data	[(None, 900, 2)]	0
conv1d	Conv1D	128 Filter, 8, S = 1, ReLU	(None, 893, 128)	2176
batch_norm	Batch Normalization		(None, 893, 128)	512
max_pooling1d	MaxPooling1D	pool size = 4, S = 2	(None, 447, 128)	0
conv1d_1	Conv1D	128 Filter, 8, S = 1, ReLU	(None, 440, 128)	131200
batch_norm_1	Batch Normalization		(None, 440, 128)	512
max_pooling1d_1	MaxPooling1D	pool size = 4, S = 2	(None, 220, 128)	0
conv1d_2	Conv1D	128 Filter, 8, S = 1, ReLU	(None, 215, 128)	98432
batch_norm_2	Batch Normalization		(None, 215, 128)	512
max_pooling1d_2	MaxPooling1D	pool size = 3, S = 2	(None, 108, 128)	0
conv1d_3	Conv1D	128 Filter, 6, S = 1, ReLU	(None, 103, 128)	98432
batch_norm_3	Batch Normalization		(None, 103, 128)	512
max_pooling1d_3	MaxPooling1D	pool size = 3, S = 2	(None, 52, 128)	0
conv1d_4	Conv1D	128 Filter, 4, S = 1, ReLU	(None, 49, 128)	65664
batch_norm_4	Batch Normalization		(None, 49, 128)	512
max_pooling1d_4	MaxPooling1D	pool size = 3, S = 1	(None, 49, 128)	0
conv1d_5	Conv1D	128 Filter, 2, S = 1, ReLU	(None, 48, 128)	32896
batch_norm_5	Batch Normalization		(None, 48, 128)	512
max_pooling1d_5	MaxPooling1D	pool size = 3, S = 1	(None, 48, 128)	0
conv1d_6	Conv1D	128 Filter, 1, S = 1, ReLU	(None, 47, 128)	32896
batch_norm_6	Batch Normalization		(None, 47, 128)	512
max_pooling1d_6	MaxPooling1D	pool size = 2, S = 1	(None, 47, 128)	0
conv1d_7	Conv1D	128 Filter, 2, S = 1, ReLU	(None, 46, 128)	32896
batch_norm_7	Batch Normalization		((None, 46, 128)	512
max_pooling1d_7	MaxPooling1D	pool size = 2, S = 1	(None, 46, 128)	0
lstm	LSTM	LSTM Layer of 256 units	(None, 256)	394240
flatten	Flatten	Flatten	(None, 256)	0
batch_norm_7	Batch Normalization		(None, 256)	1024
dense	Dense	Dense Layer of 32 units	(None, 32)	8224
dense_1	Dense	Dense Layer of 32 units	(None, 32)	1056
dense_1	Dense	Dense Layer of 2 units	(None, 2)	66

3. Removed fully connected layers in the end and avoided flattening layers to substantially reduce network complexity and parameters.
4. Removed Dropout layers at the end. **The dropout layer was used in the original AlexNet because it had 60 million parameters, which caused a substantial overfitting problem, but in proposed research removing it or retaining it does not make any difference since the proposed model has very few parameters compared to original Alexnet.**
5. Added an LSTM layer just before the last layer for sequence learning.

After making the following changes to the Alexnet architecture, we were able to reduce the number of parameters significantly, and the proposed model now has 1.7 million parameters instead of the original Alexnet architecture, which had 60 million parameters. The description of each layer is shown in Table 3.

Finally, the Adamax optimizer is used to optimize the model. Adamax is a variant of Adam based on the infinity norm. Adamax automatically adapts a different step size (learning rate) for each parameter in the optimization problem. A categorical Cross-Entropy loss function has been used.

4.5.2. Proposed model II - ApneaNet

A CNN comprises two essential components: feature extraction and classification. The model's initial layers may be thought of as descriptors of image features, while the latter levels are related to particular categories. Several convolution layers are used in feature extraction, followed by max-pooling and an activation function. The classifier is typically made up of fully connected layers and a softmax activation function. If there are more no. of classes in the dataset, there will be more no. of features for a model to learn. So the feature extraction component of a CNN should be deeper and more complex to learn the complex features.

Since there are two classes, a model has to discriminate between them, so we have used eight convolutional, max pooling, and activation layers. These parameters, such as no. of layers and no. of neurons, are determined by performing extensive experiments balancing efficiency and accuracy. We have kept increasing the number of layers until a particular threshold so that model does not overfit. Since there are a limited number of samples in our dataset, increasing the depth of the network might overfit the model and result in neurons memorizing the data.

The Proposed Model ApneaNet contains 13 layers, each having its own set of parameters that may be learned. The model consists of eight layers, starting with a layer that combines max pooling and batch normalization, an LSTM layer; then, three fully connected layers are used after flattening the layer. Except for the output layer, ReLU [66] activation function is employed in each of these layers as shown in Fig. 4.

The first two convolutional layers consist of 128 neurons, each of kernel size of 8, followed by a Batch Normalization layer and max pooling layer of pool size four and stride of 2. The following two convolutional layers also consist of 128 neurons but with a kernel size of 6, followed by the Batch Normalization layer and a max-pooling layer of pool size three and stride of 2. Then one convolutional layer with 128 neurons and kernel size four is added, followed by a Batch Normalization layer and a max-pooling layer of pool size three and stride of 2. Then, at last, three convolutional layers are there with a kernel size of 2, followed by a Batch Normalization layer, one max pooling layer with a pool size of 3, and the last two of 2 with a stride of 2.

Further, an LSTM layer is added with 256 units. Then a flatten layer is used to convert all data into a one-dimensional array. After normalizing the inputs, two fully connected Dense layer consisting of 32 neurons is used, and then, at last, the layer is mapped with an output layer consisting of two neurons; one is for Apnea Event, and another is for Normal Event with a softmax activation function for classification purposes. Unlike the networks pre-trained on ImageNet Dataset, where the model has to classify between 1000 classes, we have only two classes in our dataset. So, scaling down the dimensions of the classification layer does not hamper the performance much. We have

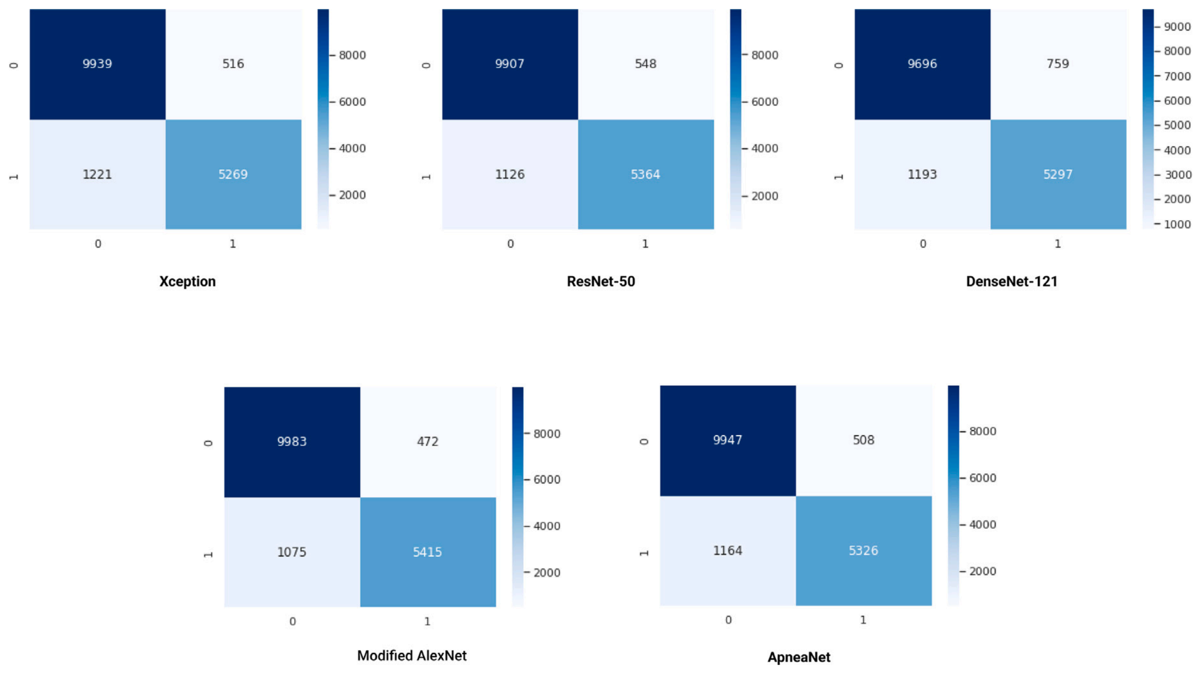


Fig. 5. Confusion matrices of different implemented models.

Table 5
Distribution of dataset in training and evaluation sets.

	Training data	Evaluation data
Evaluation 1	35 Recordings	35 Recordings
Evaluation 2	28 Recordings	7 Recordings

also determined these parameters by performing experiments balancing both efficiency and accuracy. A categorical Cross-Entropy loss function has been used. The description of each layer is shown in Table 4.

4.6. Proposed algorithm

The proposed algorithm is based on end-to-end training of the proposed model. The notations used in the algorithm are described here, with $\delta 1$ and $\delta 2$ referring to the training and testing datasets, respectively. μ is the model's learning rate, which indicates how rapidly it adapts to the problem. It usually has a value close to zero. ϵ is the total number of iterations (also known as epochs) for which the CNN model has been trained. β is another customizable hyper-parameter that usually has the form of 2n.

The algorithm starts by establishing the model's top input layer (refer to step 2 of the algorithm) and adding it as the head layer to the remaining six layers (refer to steps 1, 2, and 3 of the algorithm). Then a for loop with several iterations ranging from 1 to the entire number of iterations (ϵ) to train and update the weights using forward and backward propagation (refer to steps 7–9 of the algorithm).

5. Experimental setup and results

5.1. Dataset division

Apnea-ECG Database comprises 70 records, 35 of which are for training the model, and the rest 35 are for validating the model. Researchers in the past have conducted their study on both the 70 data records and only 35 training data records by splitting them into training and validation sets. As a result, we also validated our models on both Evaluation sets. The first consists of 35 training records and 35 testing records, the second one of 28 training records from 35 training data windows, and the rest 7 for validating the model, as shown in Table 5.

Table 6
Results of **Xception model** in terms of its accuracy and parameters on different optimizers.

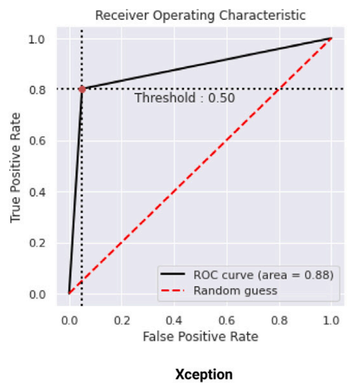
No.	Model	Optimizer	Training accuracy	Testing accuracy	Parameters
1	Xception	Adam	99.99	88.89	20,714,330
2	Xception	SGD	99.96	88.86	20,714,330
3	Xception	RMSprop	99.99	88.44	20,714,330
4	Xception	Adagrad	99.92	87.00	20,714,330
5	Xception	Adamax	99.99	89.75	20,714,330
6	Xception	Adadelta	99.12	87.53	20,714,330
7	Xception	Nadam	99.99	88.69	20,714,330

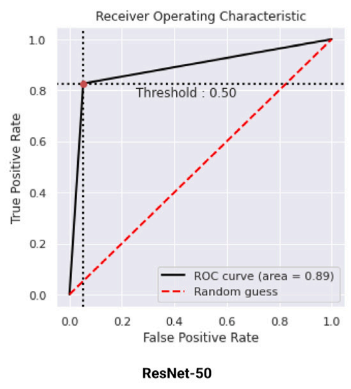
Table 7
Results of **ResNet50 model** in terms of its accuracy and parameters on different optimizers.

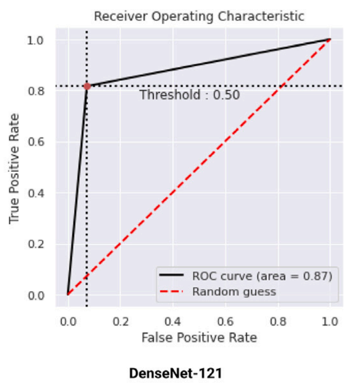
No.	Model	Optimizer	Training accuracy	Testing accuracy	Parameters
1	ResNet50	Adam	100	90.12	16,038,466
2	ResNet50	SGD	99.92	89.22	16,038,466
3	ResNet50	RMSprop	99.97	89.61	16,038,466
4	ResNet50	Adagrad	99.80	85.94	16,038,466
5	ResNet50	Adamax	99.99	89.17	16,038,466
6	ResNet50	Adadelta	86.86	84.22	16,038,466
7	ResNet50	Nadam	99.99	89.35	16,038,466

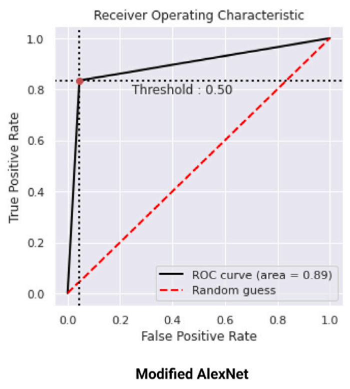
Table 8
Results of **DenseNet121 model** in terms of its accuracy and parameters on different optimizers.

No.	Model	Optimizer	Training accuracy	Testing accuracy	Parameters
1	DenseNet121	Adam	99.88	88.14	86,956,386
2	DenseNet121	SGD	99.88	88.46	86,956,386
3	DenseNet121	RMSprop	99.96	88.65	86,956,386
4	DenseNet121	Adagrad	99.97	87.97	86,956,386
5	DenseNet121	Adamax	99.97	88.48	86,956,386
6	DenseNet121	Adadelta	99.75	84.77	86,956,386
7	DenseNet121	Nadam	99.89	88.91	86,956,386









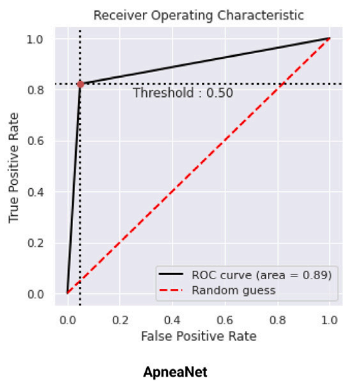


Fig. 6. AUC-ROC plots of different implemented models.

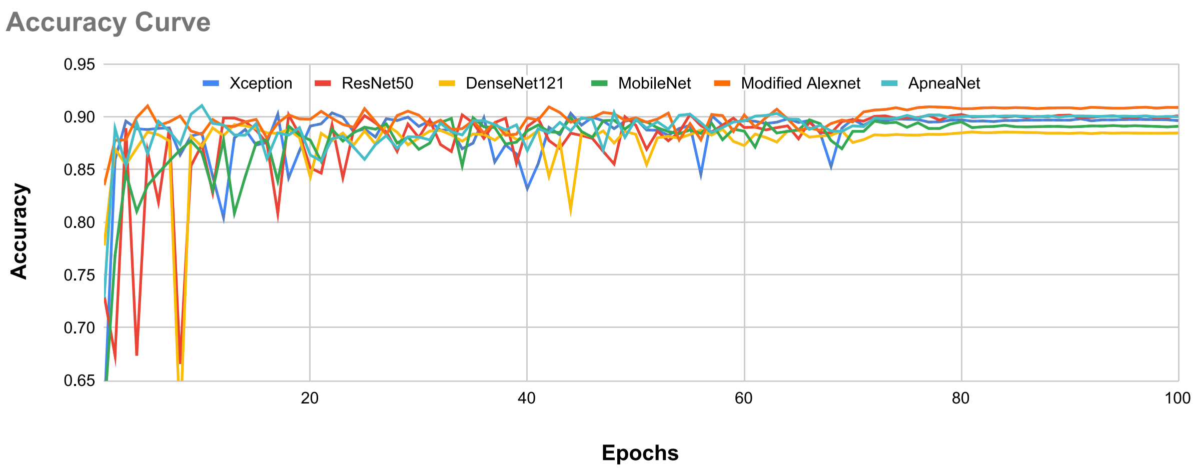


Fig. 7. Accuracy VS epochs of the top performing models.

Table 9
Results of **Modified AlexNet + LSTM model** in terms of its accuracy and parameters on different optimizers.

No.	Model	Optimizer	Training accuracy	Testing accuracy	Parameters
1	Modified Alexnet + LSTM	Adam	99.99	89.91	1,689,890
2	Modified Alexnet + LSTM	SGD	99.99	89.80	1,689,890
3	Modified Alexnet + LSTM	RMSprop	99.99	88.75	1,689,890
4	Modified Alexnet + LSTM	Adagrad	99.93	87.72	1,689,890
5	Modified Alexnet + LSTM	Adamax	100	90.87	1,689,890
6	Modified Alexnet + LSTM	Adadelta	96.24	87.16	1,689,890
7	Modified Alexnet + LSTM	Nadam	99.98	89.47	1,689,890

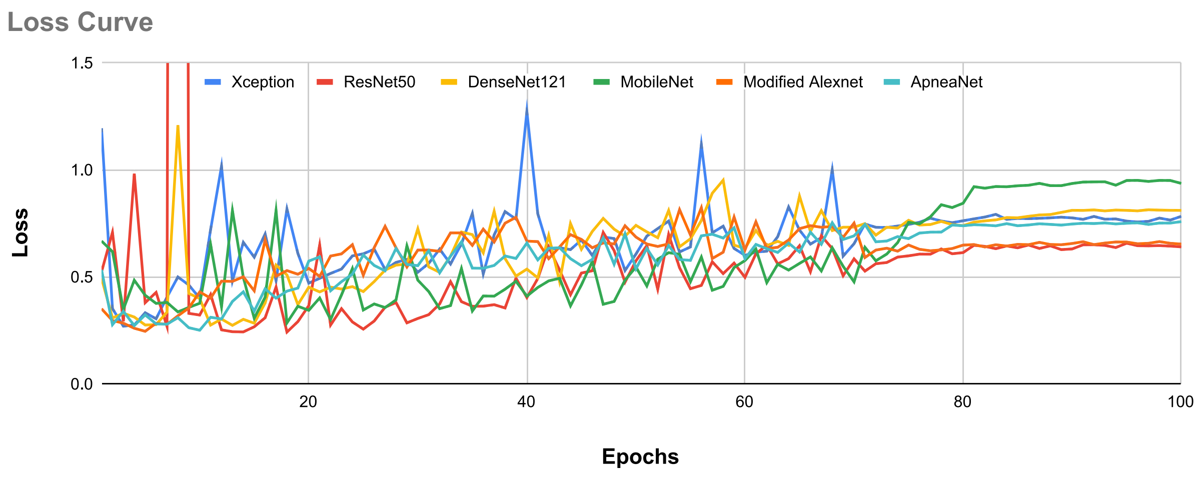


Fig. 8. Loss VS epochs of the top performing models.

Table 10
Results of **ApneaNet model** in terms of its accuracy and parameters on different optimizers.

No.	Model	Optimizer	Training accuracy	Cross validation accuracy	Parameters
1	ApneaNet	Adam	99.95	88.83	903,298
2	ApneaNet	SGD	99.98	88.66	903,298
3	ApneaNet	RMSprop	99.73	88.46	903,298
4	ApneaNet	Adagrad	99.65	85.79	903,298
5	ApneaNet	Adamax	99.99	90.13	903,298
6	ApneaNet	Adadelta	90.72	83.57	903,298
7	ApneaNet	Nadam	99.95	90.06	903,298

5.2. Experimental setup

All the code implementations were implemented with the Tensorflow framework in python. The training is performed for small epochs on personal computers with Intel(R) Core(TM) i7-6500U CPU 2.50 GHz, Nvidia 940M GPU with compute capability 5.0, and 16 GB RAM. The whole training part is performed on a workstation equipped with GPU Nvidia RTX 3080, having a compute capability of 8.60 and 32 GB of GPU RAM. Each Model was trained for 100–500 epochs to obtain the best training and testing accuracy.

5.3. Results

To validate the proposed approach, the authors used two tried and tested approaches per segment and per recording classification to measure the accuracy of our trained models.

5.3.1. Per segment classification

Per segment evaluation analyzes each recording minute after the minute is observed and evaluated by the model. This is crucial to establishing a robust model as it helps to predict SA in patients suspecting of the same accurately. Per segment classification for SA detection has been taken as the primary evaluation tool for comparing the performances of various models chosen based on better efficiency.

Xception, ResNet50, DenseNet121, Modified Alexnet, and ApneaNet have been experimented with different optimizers such as Adam, SGD, RMSProp, Adagrad, Adamax, Adadelta, and Nadam. Xception has the highest testing accuracy of 89.75% with Adamax optimizer, using roughly 20.7 million parameters as shown in Table 6. ResNet50 has demonstrated the highest testing accuracy of 90.12% with Adam optimizer, using approximately 20 million parameters as shown in Table 7. Next, DenseNet121 has the highest testing accuracy of 88.91% with Nadam optimizer, using roughly 86.9 million parameters as shown in Table 8, which shows that it takes the highest computational power. The proposed model 1 (Modified Alexnet) has the highest testing accuracy of 90.87% with Adamax optimizer, using just 1.7 million parameters as shown in Table 9. Whereas, the Proposed model 2 (ApneaNet) has a testing accuracy of 90.13% with Adamax optimizer with even lesser computational cost, using just 0.9 million parameters as shown in Table 10. The performance of these models is also evaluated using their confusion matrices and AUC-ROC plots as illustrated in Figs. 5 and 6.

These models have been trained and tested on a 35-35 split of the PhysioNet dataset. The training of the networks has been performed on 35 training samples, and rest 35 is kept for testing purposes. The highest accuracy is achieved through the Proposed Model 1 (Modified Alexnet), i.e., 90.87%. The authors also divided the 35 training data further into 28-7 data splits of training and validation sets, respectively, and then trained modified Alexnet and ApneaNet models on the divided dataset, i.e., 28 data windows. The modified Alexnet model showed a significant increment in the testing accuracy from 90.87% to 95.69%, whereas ApneaNet is from 90.13% to 96.37%.

Taking 35 training sets as a complete dataset is unreliable because there is already a shortage of data. However, the proposed model ApneaNet showed a testing accuracy of 96.37% and sensitivity of 97.52% on the dataset of 35 (28-7 split) recordings which surpassed all the proposed and widely accepted models as shown in Table 12.

Fig. 7 shows that the accuracy of various implemented models is proportional to the number of epochs. When the number of epochs is changed from 1 to 68, there is an uneven rise and drop in precision value. For different implemented models, the accuracy remains consistent at epoch 68. Fig. 8 further shows that the size of the loss is proportional to the number of epochs. When the number of epochs is changed from 1 to 70, there is an uneven rise and drop in loss value. For different implemented models, the loss value remains constant at epoch 70.

5.3.2. Per recording classification

Each recording refers to a collective set of ECG segments of a duration of 1 min. Classification for SA depends on the recording as a whole compared to per-segment classification. The classification occurs on an Apnea/Hypopnea Index (AHI). Traditionally if AHI $>$ 5, it is classified as SA.

Per recording, classification was done on Xception, ResNet50, DenseNet121, Modified AlexNet + LSTM, and ApneaNet, all having accuracy around 97.14% as shown in Table 11.

Table 11
Classification report of best performing models based on per segment and per recording.

No.	Model	Per segment accuracy	Per recording accuracy	Sensitivity	Specificity
1	Xception	89.75	97.14	95.064	81.11
2	ResNet50	90.12	97.14	94.75	82.65
3	DenseNet121	88.48	97.14	92.74	81.61
4	Modified AlexNet + LSTM	90.87	97.14	95.48	83.43
5	ApneaNet	90.13	97.14	95.14	82.06

Table 12
Previous works comparison table.

Author	Method	Dataset window	Accuracy (%)	Sensitivity (%)	Specitivity (%)
Kunyang Li [2]	Decision Fusion of (DNN, SVM, HMM) and (DNN, ANN, HMM)	70	84.7	88.9	82.1
Q. Shen [67]	(MSDA-1DCNN) with Hidden Markov Model	70	89.4	89.8	89.1
Song C [7]	Hidden Markov Model	70	86.2	82.6	88.4
S. Thompson [24]	1D - CNN	35	95.28	96.12	97.30
Wang T [4]	Modified LeNet-5	70	87.6	83.1	90.3
Mukherjee [25]	MLP Based Ensemble Model	70	85.58	84.43	88.26
Fazla Rabbi Mashrur [26]	Scalogram based CNN	35	94.30	94.30	94.51
Chang HY [27]	1D - CNN	70	87.9	81.1	92.0
Junming Zhang [28]	CNN - LSTM	35	96.1	96.1	96.2
Wang L [29]	A residual network with 31 residual blocks	35	94.4	93.0	94.9
Wang, T [68]	Time Window ANN with MLP	70	87.3	85.1	88.7
Almutairi, H [69]	CNN, CNN with LSTM and CNN with GRU	70	89.11	89.91	87.78
Dey, Debangshu [70]	CNN	35	94.33	93.88	95.67
Wang et al. [71]	CNN	70	82.36	81.78	83.93
Sharan et al. [72]	CNN	70	83.29	82.80	87.39
Bernardini et al. [73]	AIOSA (CNN + LSTM)	35	94.3	95.1	93.7
Tripathy et al. [74]	KELM (Kernel Extreme Learning Machine)	70	76.37	75.34	74.64
Proposed model	Modified ALexNet + LSTM	35	95.69	97.28	93.13
Proposed model	**Modified ALexNet + LSTM**	70	**90.87**	95.48	83.43
Proposed model	**ApneaNet**	35	**96.37**	**97.52**	94.53
Proposed model	ApneaNet	70	90.13	95.14	82.06

6. Conclusion and future scope

The Intuition of RR intervals can only help in predicting Sleep apnea caused due to neurological symptoms or cardiovascular symptoms. However, for physical impediments in the upper respiratory tract, RR intervals added with better respiration data can further help in improving the accuracy of our model.

In this study, the authors have implemented Xception, ResNet50, DenseNet121, and AlexNet model architectures on the SA PhysioNet dataset to detect Obstructive Sleep Apnea. From this inference, they have formulated their models as Modified AlexNet + LSTM and ApneaNet, which show better accuracy and performance than the state-of-art models with much less computational cost in terms of parameters. Modified ALexNet + LSTM shows an accuracy of 90.87%, specificity of 83.43%, and sensitivity of 95.48% using 1.7 million parameters which are 10 to 50 times lesser than the other models. This model has also been implemented on a 28-7 training and testing dataset split, on which it has shown remarkable performance with an accuracy of 95.69%. ApneaNet has demonstrated an accuracy of 90.13%, specificity of 82.06%, and sensitivity of 95.14%, using 0.9 million parameters which require even lesser computational cost as shown in Table 11. The accuracy of the proposed novel approach has proven to be the best on both the original dataset (35-35) and split dataset (28-7), as compared to the state-of-art.

The models proposed in this research can be further used to detect cardiovascular diseases such as heart arrhythmia, conduction disturbances, acute coronary syndromes, cardiac chamber hypertrophy, enlargement, etc., through ECG signals.

7. Code availability

Code will be made available on request.

CRediT authorship contribution statement

Gaurav Srivastava: Conceptualization, Methodology, Software, Code implementations, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Aninditaa Chauhan:** Literature survey, Investigation, Writing – original draft, Writing – review & editing. **Nitigya Kargeti:** Software, Data preprocessing, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft, Writing – review & editing. **Nitesh Pradhan:** Validation, Resources, Investigation, Writing – review & editing, Supervision, Research administration. **Vijaypal Singh Dhaka:** Resources, Supervision.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.bspc.2023.104754.

Data availability

The data used in this study is publicly available at https://physionet.org/content/apnea-ecg/1.0.0/.

References

[1] M. Clinic, Sleep Apnea, https://www.mayoclinic.org/diseases-conditions/sleep-apnea/symptoms-causes/syc-20377631.

[2] K. Li, W. Pan, Y. Li, Q. Jiang, G. Liu, A method to detect sleep Apnea based on deep neural network and hidden Markov model using single-lead ECG signal, Neurocomputing 294 (2018) 94–101.

[3] N.M. Punjabi, The epidemiology of adult obstructive sleep Apnea, Proc. Am. Thoracic Soc. 5 (2) (2008) 136–143.

[4] T. Wang, C. Lu, G. Shen, F. Hong, Sleep Apnea detection from a single-lead ECG signal with automatic feature-extraction through a modified LeNet-5 convolutional neural network, PeerJ 7 (2019) e7731.

[5] V. Kapur, K.P. Strohl, S. Redline, C. Iber, G. O'connor, J. Nieto, Underdiagnosis of sleep Apnea syndrome in US communities, Sleep Breath. 6 (02) (2002) 049–054.

[6] L.J. Meltzer, C.M. Walsh, A.A. Peightal, Comparison of actigraphy immobility rules with polysomnographic sleep onset latency in children and adolescents, Sleep Breath. 19 (4) (2015) 1415–1423.

[7] C. Song, K. Liu, X. Zhang, L. Chen, X. Xian, An obstructive sleep Apnea detection approach using a discriminative hidden Markov model from ECG signals, IEEE Trans. Biomed. Eng. 63 (7) (2015) 1532–1542.

[8] S.M. Caples, A. Garcia-Touchard, V.K. Somers, Sleep-disordered breathing and cardiovascular risk, Sleep 30 (3) (2007) 291–303.

[9] A. Azarbarzin, Z. Moussavi, Snoring sounds variability as a signature of obstructive sleep Apnea, Med. Eng. Phys. 35 (4) (2013) 479–485.

[10] R. Hornero, D. Álvarez, D. Abásolo, F. del Campo, C. Zamarron, Utility of approximate entropy from overnight pulse oximetry data in the diagnosis of the obstructive sleep Apnea syndrome, IEEE Trans. Biomed. Eng. 54 (1) (2006) 107–113.

[11] Y. Sun, N. Thakor, Photoplethysmography revisited: From contact to noncontact, from point to imaging, IEEE Trans. Biomed. Eng. 63 (3) (2015) 463–477.

[12] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G.B. Kim, J.B. Seo, N. Kim, Deep learning in medical imaging: General overview, Korean J. Radiol. 18 (4) (2017) 570–584.

[13] Y. Bengio, Y. LeCun, et al., Scaling learning algorithms towards AI, Large-Scale Kernel Mach. 34 (5) (2007) 1–41.

[14] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[15] L. Perez, J. Wang, The effectiveness of data augmentation in image classification using deep learning, 2017, arXiv preprint arXiv:1712.04621.

[16] G. Srivastava, A. Chauhan, M. Jangid, S. Chaurasia, CoviXNet: A novel and efficient deep learning model for detection of COVID-19 using chest X-Ray images, Biomed. Signal Process. Control (2022) 103848.

[17] G. Srivastava, A. Chauhan, N. Pradhan, CJT-DEO: Condorcet's Jury theorem and differential evolution optimization based ensemble of deep neural networks for pulmonary and colorectal cancer classification, Appl. Soft Comput. 132 (2023) 109872.

[18] G. Srivastava, N. Pradhan, Y. Saini, Ensemble of deep neural networks based on Condorcet's Jury theorem for screening Covid-19 and pneumonia from radiograph images, Comput. Biol. Med. 149 (2022) 105979.

[19] S.S. Yadav, S.M. Jadhav, Deep convolutional neural network based medical image classification for disease diagnosis, J. Big Data 6 (1) (2019) 1–18.

[20] S. Albawi, T.A. Mohammad, Understanding of Convolutional Neural Network 2017, IEEE Website, 2017.

[21] L. Lu, Y. Zheng, G. Carneiro, L. Yang, Deep learning and convolutional neural networks for medical image computing, Adv. Comput. Vis. Pattern Recogn. 10 (2017) 3–978.

[22] G. Srivastava, A. Chauhan, M. Jangid, A. Jain, An analysis of deep learning models to diagnose COVID-19 using radiography images, in: 2022 International Conference for Advancement in Technology, ICONAT, IEEE, 2022, pp. 1–7.

[23] S. Saha, A comprehensive guide to convolutional neural networks—the ELI5 way, Towards Data Sci. 15 (2018).

[24] S. Thompson, P. Fergus, C. Chalmers, D. Reilly, Detection of obstructive sleep Apnoea using features extracted from segmented time-series ECG signals using a one dimensional convolutional neural network, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–8.

[25] D. Mukherjee, K. Dhar, F. Schwenker, R. Sarkar, Ensemble of deep learning models for sleep Apnea detection: An experimental study, Sensors 21 (16) (2021) 5425.

[26] F.R. Mashrur, M.S. Islam, D.K. Saha, S.R. Islam, M.A. Moni, SCNN: Scalogram-based convolutional neural network to detect obstructive sleep Apnea using single-lead electrocardiogram signals, Comput. Biol. Med. (2021) 104532.

[27] H.-Y. Chang, C.-Y. Yeh, C.-T. Lee, C.-C. Lin, A sleep Apnea detection system based on a one-dimensional deep convolution neural network model using single-lead electrocardiogram, Sensors 20 (15) (2020) 4157.

[28] J. Zhang, Z. Tang, J. Gao, L. Lin, Z. Liu, H. Wu, F. Liu, R. Yao, Automatic detection of obstructive sleep Apnea events using a deep CNN-LSTM model, Comput. Intell. Neurosci. 2021 (2021).

[29] L. Wang, Y. Lin, J. Wang, A RR interval based automated Apnea detection approach using residual network, Comput. Methods Programs Biomed. 176 (2019) 93–104.

[30] P.K. Upadhyay, C. Nagpal, Wavelet based performance analysis of SVM and RBF kernel for classifying stress conditions of sleep EEG, Sci. Technol. 23 (3) (2020) 292–310.

[31] A. Panda, R.B. Pachori, N.D. Sinnappah-Kang, Classification of chronic myeloid Leukemia neutrophils by hyperspectral imaging using Euclidean and Mahalanobis distances, Biomed. Signal Process. Control 70 (2021) 103025.

[32] I.-D. Borlea, R.-E. Precup, A.-B. Borlea, D. Iercan, A unified form of fuzzy C-means and K-means algorithms and its partitional implementation, Knowl.-Based Syst. 214 (2021) 106731.

[33] L. Torrey, J. Shavlik, Transfer learning, in: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, IGI global, 2010, pp. 242–264.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.

[35] P. Marcelino, Transfer learning from pre-trained models, Towards Data Sci. (2018).

[36] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.

[37] S. Saxena, Introduction to The Architecture of Alexnet, Analytics Vidhya, 2021.

[38] J. Wei, AlexNet: The architecture that challenged CNNs, Towards Data Sci. 3 (2019).

[39] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1251–1258.

[40] F. Chollet, Xception: Deep learning with depthwise separable convolutions, 2017, pp. 1800–1807, http://dx.doi.org/10.1109/CVPR.2017.195.

[41] A. Sarkar, Xception: Implementing from scratch using tensorflow, 2020.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[43] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[44] G. Boesch, Deep Residual Networks (ResNet, ResNet50), https://viso.ai/deep-learning/resnet-residual-neural-network/.

[45] P. Dwivedi, Understanding and coding a ResNet in Keras, 2019, Retrieved.

[46] V. Feng, An overview of ResNet and its variants, 2017.

[47] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[48] I. Allaouzi, M.B. Ahmed, B. Benamrou, An encoder-decoder model for visual question answering in the medical domain, in: CLEF (Working Notes), 2019.

[49] A.L. Goldberger, L.A. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000) e215–e220.

[50] T. Penzel, G.B. Moody, R.G. Mark, A.L. Goldberger, J.H. Peter, The Apnea-ECG database, in: Computers in Cardiology 2000. Vol. 27 (Cat. 00CH37163), IEEE, 2000, pp. 255–258.

[51] P. Hamilton, Open source ECG analysis, in: Computers in Cardiology, IEEE, 2002, pp. 101–104.

[52] L. Chen, X. Zhang, C. Song, An automatic screening approach for obstructive sleep Apnea diagnosis based on single-lead electrocardiogram, IEEE Trans. Autom. Sci. Eng. 12 (1) (2014) 106–115.

[53] N. Pradhan, V. Dhaka, G. Rani, H. Chaudhary, Transforming view of medical images using deep learning, Neural Comput. Appl. 32 (2020) http://dx.doi.org/10.1007/s00521-020-04857-z.

[54] V. Bushaev, Adam—latest trends in deep learning optimization, Towards Data Sci. (2018).

[55] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[56] J. Brownlee, Gradient descent optimization with AdaMax from scratch, 2021.

[57] J. Bjorck, C. Gomes, B. Selman, K.Q. Weinberger, Understanding batch normalization, 2018, arXiv preprint arXiv:1806.02375.

[58] S. Santurkar, D. Tsipras, A. Ilyas, A. Mądry, How does batch normalization help optimization? in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 2488–2498.

[59] J. Brownlee, Softmax activation function with Python, 2020, Machine Learning Mastery Website. Available Online: https://Machinelearningmastery.Com/Softmax-Activation-Function-with-Python/ (Accessed on 1 November 2020).

[60] H. Mahmood, The softmax function, simplified, Towards Data Sci. (2018).

[61] S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology, ICET, Ieee, 2017, pp. 1–6.

[62] P. Kim, Convolutional neural network, in: MATLAB Deep Learning, Springer, 2017, pp. 121–147.

[63] P. Skalski, Gentle dive into math behind convolutional neural networks, vol. 5, 2019, Online: https://Towardsdatascience.Com/Gentle-Dive-Into-Math-behind-Convolutional-Neural-Networks-79a07dd44cf9 (Access: May 5).

[64] I.C. Education, Convolutional neural networks, 2020, IBM.[Online]. Tersedia: https://Www.Ibm.Com/Cloud/Learn/Convolutional-Neuralnetworks [Diakses: 16-Mei-2021].

[65] Y. Yu, X. Si, C. Hu, J. Zhang, A review of recurrent neural networks: LSTM cells and network architectures, Neural Comput. 31 (7) (2019) 1235–1270.

[66] A.F. Agarap, Deep learning using rectified linear units (relu), 2018, arXiv preprint arXiv:1803.08375.

[67] Q. Shen, H. Qin, K. Wei, G. Liu, Multiscale deep neural network for obstructive sleep Apnea detection using RR interval from single-lead ECG signal, IEEE Trans. Instrum. Meas. 70 (2021) 1–13.

[68] T. Wang, C. Lu, G. Shen, Detection of sleep Apnea from single-lead ECG signal using a time window artificial neural network, BioMed Res. Int. 2019 (2019).

[69] H. Almutairi, G.M. Hassan, A. Datta, Detection of obstructive sleep Apnoea by ECG signals using deep learning architectures, in: 2020 28th European Signal Processing Conference, EUSIPCO, IEEE, 2021, pp. 1382–1386.

[70] D. Dey, S. Chaudhuri, S. Munshi, Obstructive sleep Apnoea detection using convolutional neural network based deep learning framework, Biomed. Eng. Lett. 8 (1) (2018) 95–100.

[71] X. Wang, M. Cheng, Y. Wang, S. Liu, Z. Tian, F. Jiang, H. Zhang, Obstructive sleep Apnea detection using ecg-sensor with convolutional neural networks, Multimedia Tools Appl. 79 (23) (2020) 15813–15827.

[72] R.V. Sharan, S. Berkovsky, H. Xiong, E. Coiera, ECG-derived heart rate variability interpolation and 1-D convolutional neural networks for detecting sleep Apnea, in: 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society, EMBC, IEEE, 2020, pp. 637–640.

[73] A. Bernardini, A. Brunello, G.L. Gigli, A. Montanari, N. Saccomanno, AIOSA: An approach to the automatic identification of obstructive sleep Apnea events based on deep learning, Artif. Intell. Med. 118 (2021) 102133.

[74] R. Tripathy, Application of intrinsic band function technique for automated detection of sleep Apnea using HRV and EDR signals, Biocybern. Biomed. Eng. 38 (1) (2018) 136–144.