Real-Time implementation of IoT with GCRNN and Fine-Tuning via Ninja Optimizer for Human Activity Recognition-Based Fall Detection for hospitalized patients

P. sakthiprakash
Technical Trainer, Rathinam Arts and Science College, Coimbatore

***Abstract*—Human Activity Recognition (HAR) based research develops innovative frameworks to enhance recognition performance. In this research topic utilizes a Graph Convolutional Recurrent Neural Network (GCRNN) to effectively capture both spatial and temporal dependencies in motion data collected from wearable sensors. The GCRNN model integrates Graph Convolutional Networks (GCN) to learn spatial relationships among sensor nodes, while recurrent layers model sequential dependencies, enhancing the classification of fall-related activities. To further optimize model performance, we incorporate the Ninja Optimizer, a novel optimization technique designed to improve learning rate adaptation and parameter fine-tuning. The Ninja Optimizer accelerates convergence, mitigates overfitting, and enhances generalization, leading to more reliable fall detection outcomes. We conduct experiments on the KFall dataset, demonstrating that our GCRNN model, fine-tuned with the Ninja Optimizer, outperforms conventional machine learning classifiers and baseline deep learning models. The model is first trained using the dataset and later tested with real-time sensor values. After getting the dataset, we used preprocessing of the KFall dataset, it involves multiple steps to ensure the data is clean and suitable for training the GCRNN model. First, the dataset is loaded using Pandas, and missing values are handled through mean imputation. Next, sensor data from Sensor, are normalized using Min-Max Scaling to bring all features into a uniform range. Our system employs Accelerometers, Gyroscopes, and Heart Rate Sensors to predict fall events accurately. The Heart Rate Sensor is used to detect sudden heart rate fluctuations after a fall, helping differentiate real falls from normal activities. It enhances fall detection accuracy by providing physiological insights, enabling timely medical intervention for hospitalized patients. Additionally, IoT integration enables real-time transmission of sensor data from patients to the hospital control room, ensuring continuous monitoring of patient activities. The results of this study highlight the potential of GCRNN-based Human Activity Recognition (HAR) models in improving fall detection accuracy and reliability for hospitalized patients as accuracy of 98. 93% respectively.**

***Index Terms*—Graph Convolutional Recurrent Neural Network; Ninja Optimizer; Human Activity Recognition; Elderly fall uncovering; Wearable sensors; Sequential dependencies.**

I. INTRODUCTION

One of the most important areas of the broad field of computer vision research is Human Activity Recognition (HAR). Fundamentally, HAR entails creating and deploying models and algorithms that can automatically recognise and analyse human behaviours and actions from visual data [1]. Because of its wide range of applications in different domains, this field has seen a surge in significance. Multimedia data dissemination has rapidly expanded in recent years, involving a huge amount of photos and videos. This massive volume of data is too large for manual analysis, so suitable AI-based algorithms must be used to automatically understand it. Because of its many uses in domains like surveillance, video classification, human-computer interaction, health monitoring, and intelligent systems [2], the identification of human activities has attracted a lot of interest in computer vision research. Since both local and global features are usually created by hand, achieving high recognition accuracy requires a lot of time and domain expertise. Direct feature acquisition from raw data has been made possible by the remarkable advances in deep learning techniques in recent years, which have completely changed the methodology [3]. There are two main benefits to deep features. RGB video data has limitations in dim or dark environments, despite being widely used in HAR systems [4].

However, even in dimly lit environments, infrared (IR) data can record human activity. Creating reliable deep learning algorithms that can precisely identify human activity from infrared data—which frequently lacks the fine details available in RGB data—is the main challenge for achieving this goal [5]. If this goal is accomplished, HAR systems' usability in surveillance, medical monitoring, and other applications that function in low-visibility environments will be greatly improved. Falls among hospitalized patients, especially the elderly and those with mobility issues, pose serious health risks, leading to injuries and complications [6]. To address this, we propose an IoT-based fall detection system that utilizes wearable sensors such as accelerometers, gyroscopes, and heart rate sensors for real-time monitoring. The collected motion and physiological data are processed using a Graph Convolutional Recurrent Neural Network (GCRNN), which effectively captures both spatial and temporal dependencies in human movements [7]. To further enhance model accuracy and efficiency, we incorporate the Ninja Optimizer for fine-tuning, ensuring faster convergence and reduced overfitting. This system enables continuous monitoring and real-time fall detection, improving patient safety and healthcare response in hospitals [8].

Existing fall detection systems face several limitations and challenges that impact their accuracy and reliability. Traditional machine learning models struggle to capture both spatial and temporal dependencies in motion data, leading to higher false alarm rates. Many deep learning-based approaches rely on manual feature extraction, limiting their adaptability to diverse patient conditions. Additionally, real-time IoT integration is often inefficient due to high latency and power consumption, affecting continuous monitoring in hospital environments [9 and 10].

A. Contribution of this research paper

The primary contribution of this research is the development of an IoT-based fall detection system that integrates a Graph Convolutional Recurrent Neural Network (GCRNN) with the Ninja Optimizer to enhance Human Activity Recognition (HAR) for hospitalized patients. Unlike conventional fall detection models, which struggle with capturing both spatial and temporal dependencies, the proposed GCRNN model effectively processes motion data collected from wearable sensors, such as accelerometers, gyroscopes, and heart rate sensors. The graph convolutional layers learn spatial relationships among sensor nodes, while the recurrent layers capture sequential dependencies, leading to improved recognition of fall-related activities.

To further optimize performance, the Ninja Optimizer is incorporated for fine-tuning model parameters, enabling faster convergence, reduced overfitting, and better generalization across different patient conditions. Additionally, this research leverages IoT technology for real-time data transmission, ensuring continuous patient monitoring by securely sending sensor data to the hospital control room. Experimental validation using the KFall dataset demonstrates that the proposed GCRNN-Ninja Optimizer framework achieves higher accuracy and robustness compared to traditional machine learning classifiers and baseline deep learning models. The findings of this study highlight the potential of advanced deep learning and optimization techniques in improving fall detection accuracy, making it a valuable contribution to smart healthcare and patient safety.

B. Organisation of paper

The remaining section of the paper structured as follow as, in section 2 state as literature survey, and section 3 as followed as proposed methodology, anthen the proposed model experimental results are stated as section 4, and the finally conclusion of the paper is stated as section 5.

II. RELATED WORKS

A thorough HAR system for multi-classification tasks is created by Al-qaness MA et al., [11] to identify a variety of human movements, including walking, sitting, standing, falling, and more. Additionally, this model is trained to distinguish between fall and non-fall behaviours; this categorisation can be utilised to monitor the behaviour of the elderly and trigger rescue efforts in the event of a fall. A PCNN-Transformer, an architecture based on parallel convolutional neural networks, is used to construct the created system. In order to learn representations of temporal features from sensor data, PCNN-Transformer makes use of the residual mapping method and the parallel architecture. A concatenation operation is used to summarise the retrieved features from the input data (sensor data), after which the CNN blocks are aligned in parallel with several encoders based on Transformers. Additionally, in order to decrease the model's complexity and training time, the CNN blocks employ a residual mapping approach. Several open-source datasets, including SisFall, UniMib-SHAR, and MobiAct, are used to test the proposed approach. When compared to other deep learning models, it achieved very high accuracy rates. One example is the binary classification (fall detection) task, where the suggested model attained an average accuracy of 99.95% for SisFall, 98.68% for UniMib-SHAR, and 99.71% for MobiAct.

Recent advancements in deep convolutional neural networks have significantly increased the accuracy of video-based fall detection, as suggested by Chen Z. et al. [12]. This research proposes a video-based fall detection method that uses human poses to address these issues. Lifting 2D poses to 3D poses is the first step after a lightweight pose estimator captures them from video sequences. The second part of our work is a fall detection network that is both resilient and efficient. It uses predicted 3D postures to identify fall occurrences, which improves the respective field while keeping computation cost low through the use of relaxed convolutions. The experimental results demonstrate that the suggested method for fall detection achieves a real-time speed of 18 frames per second on a non-GPU platform and 63 frames per second on a GPU platform, with an impressive accuracy of 99.83% on the big benchmark action recognition dataset NTU RGB+D.

Using Internet of Things (IoT) and multi-stage deep learning algorithms, Paul SK et al. [13] suggests a new HMR approach to MRHA identification. Starting with skeletal frame sequences, the method uses EfficientNet to optimise spatial feature extraction using seven Mobile Inverted Bottleneck Convolutions (MBConv) blocks. Then, it uses ConvLSTM to capture spatio-temporal patterns. In the end, the predictions are made using a classification module that uses global average pooling, a fully connected layer, and a dropout layer. The model is tested using the HMDB51 and NTU RGB+D 120 datasets, with an emphasis on MRHA, including activities like sitting, sneezing, falling, walking, and so on. It attains an accuracy of 89.00% on HMDB51 and 94.85% for cross-subject and cross-view evaluations, respectively, on NTU RGB+D 120. In addition, the system incorporates Internet of Things capabilities through the use of a Raspberry Pi and a GSM module, which allows for the delivery of real-time notifications to both carers and patients through the Twilios SMS service. Patient monitoring, healthcare outcomes, and costs are all enhanced by this efficient and scalable solution that connects HMR with the Internet of Things.

A wearable gadget was created by Paramasivam et al. [14] to help the elderly identify and avoid falls. In addition, a number of deep learning algorithms are employed for the purpose of activity recognition in the elderly. These include RNN, CNN, LSTM, and GRU. In addition, the optimum deep learning model is determined by analysing performance data for CNN-LSTM, RNN-LSTM, and GRU-LSTM with and without an attention layer, respectively. In addition, the best deep learning model is used and the calculation time is evaluated using three separate hardware boards: the Raspberry PI 3 and 4, the Jetson Nano developer board, and the Raspberry Pi 2. When compared to other deep learning models, the results show that the CNN-LSTM with attention layer exhibits greater accuracy (97%), recall (98%), precision (0.98), and F1_Score (0.98). Additionally, compared to other edge computing devices, NVIDIA Jetson Nano has a shorter processing time. With the proposed wearable gadget being able to track the movements of the elderly and so reduce the risk of falls, this work seems to have significant societal value.

Using FMCW radar and an asymmetric convolutional residual network, Zhang Y et al. [15] suggests a way for human action recognition. The micro-Doppler time domain spectrograms of various actions are first extracted from the processed and analysed radar echo data. Secondly, to overcome the shortcomings of linear and nonlinear transformations in the ResNet18 network's residual block for micro-Doppler spectrum recognition, a technique that combines asymmetric convolution with the Mish activation function is implemented. The goal of this method is to improve the network's feature learning performance. The last step in improving the model's attention and understanding of input data is to incorporate the Improved Convolutional Block Attention Module

(ICBAM) into the residual block. The experimental results show that the suggested strategy outperforms traditional deep learning methods with a 98.28% success rate in action detection and scene categorisation, even in highly complicated environments. This approach also shows great anti-noise recognition performance and greatly enhances the recognition accuracy for actions with similar micro-Doppler properties.

C. Challenges and limitation of existing studies

While the existing studies have significantly contributed to fall detection and human activity recognition (HAR), they still face certain limitations and challenges. Al-Qaness et al. utilized a PCNN-Transformer model, but the approach relies heavily on parallel architectures and residual mapping, increasing computational complexity and requiring significant hardware resources for real-time deployment. Chen et al. introduced a video-based fall detection system using 2D-to-3D pose estimation, but video-based methods are sensitive to lighting conditions, occlusions, and privacy concerns in hospital environments. Paul et al. proposed an EfficientNet and ConvLSTM-based IoT-integrated model, but the approach depends on external Raspberry Pi and GSM modules, which may introduce latency and communication overhead. Paramasivam et al. experimented with multiple deep learning architectures on wearable devices, but their edge computing implementation using Jetson Nano and Raspberry Pi showed varying computational efficiency, making it challenging to standardize across different hardware. Zhang et al. employed FMCW radar with asymmetric convolutional networks, achieving high accuracy, but radar-based systems are prone to interference and require specialized hardware, limiting their adaptability in hospital settings. These challenges highlight the need for an optimized IoT-integrated deep learning model that balances accuracy, real-time processing, and computational efficiency for hospitalized patient monitoring.

III. PROPOSED METHODOLOGY

The proposed methodology integrates an IoT-enabled fall detection system using wearable sensors and a deep learning model to ensure real-time monitoring of hospitalized patients. The system employs accelerometers, gyroscopes, and heart rate sensors to continuously capture motion and physiological data, which are transmitted via Bluetooth Low Energy (BLE) or Wi-Fi (MQTT/HTTP protocols) to an edge computing device such as a Raspberry Pi. The received sensor signals undergo preprocessing, including noise filtering, normalization, and segmentation using a sliding window approach to extract relevant features. The preprocessed data is then fed into a Graph Convolutional Recurrent Neural Network (GCRNN), where graph convolutional layers capture spatial relationships between sensor nodes, and recurrent layers (GRU/LSTM) learn temporal dependencies in the motion sequences. To enhance model accuracy, the Ninja Optimizer is utilized for fine-tuning, ensuring faster convergence, reduced overfitting, and improved generalization. The trained model classifies activities as fall or non-fall events, and if a fall is detected, the system sends real-time alerts to the hospital control room via a cloud server (AWS, Firebase) for immediate intervention. This IoT-driven GCRNN-based framework enhances fall detection accuracy, reliability, and real-time response, making it an effective solution for patient safety and smart healthcare monitoring.In elderly fall detection for smart healthcare is carried out by advanced deep learning model and it is shown in Figure 1, where each of its blocks are described in the upcoming sub-section.

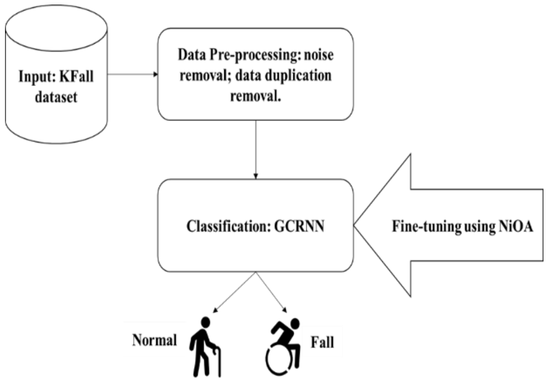

Figure 1: workflow of proposed methodology

D. Dataset description

The KFall dataset is a widely used benchmark dataset designed for fall detection and human activity recognition (HAR). It consists of motion data collected from wearable sensors that track human movements, particularly falls, to develop and evaluate fall detection models. Sensor Modalities: The dataset typically includes data from accelerometers and gyroscopes, which are essential for detecting sudden movements, changes in posture, and fall-related activities. The dataset contains a mix of fall events (e.g., forward fall, backward fall, side fall) and non-fall activities (e.g., walking, sitting, and standing, lying down). This helps in distinguishing falls from regular human movements.The dataset is collected using wearable devices placed on different body parts (such as the waist, wrist, and chest) to capture diverse motion patterns.The dataset provides high-resolution time-series data, allowing machine learning and deep learning models to analyze fine-grained motion variations. KFall is designed for hospitalized and elderly patient monitoring, making it ideal for IoT-based fall detection systems in healthcare environments. KFall [24] dataset was collected from 32 young, healthy Korean men who participated in 21 different types of ADL: “walk quickly”, “jog quickly”, “sit down to a chair quickly”, and “lie down on a bed quickly”. Addition-ally, 15 distinct types of engineered falls are featured, including “forward/lateral fall when trying to get up”, “forward/lateral/backward fall while sitting, caused by fainting”, “forward fall while walking/jogging caused by a trip/slip”, and “forward/lateral fall when trying tosit/get down/up”. KFall encompasses 5,075 instances, comprising 2,729 related to 2,346 related to falls. Triaxial acceleration, triaxial Euler angle, and triaxial gyroscope data are included in each instance. Each individual had an affixed to their lower back, with a frequency of 100 Hz. The dataset is split randomly using sci-kit learn library’s train test split (), 80% for training and 20% for testing. The NiOA algorithm optimizes the network,

E. preprocessing of the KFall dataset

The preprocessing of the KFall dataset involves multiple steps to ensure the data is clean and suitable for training the GCRNN model. First, the dataset is loaded using Pandas, and missing values are handled through mean imputation. Next, sensor data from accelerometers and gyroscopes are normalized using Min-Max Scaling to bring all features into a uniform range. Feature engineering is applied by extracting statistical metrics such as mean, standard deviation, variance, and computing the Signal Magnitude Vector (SMV) to capture movement intensity. Additionally, Fast Fourier Transform (FFT) is used to analyze frequency domain characteristics. Since the dataset consists of time-series sensor data, a sliding window technique is applied with a 2-second window and 50% overlap to segment data into meaningful sequences. Finally, activity labels (e.g., Fall, Walking, Sitting) are encoded into numerical values for model compatibility. These preprocessing steps enhance feature representation, improve classification performance, and prepare the dataset for training and evaluation in the fall detection system.

F. Sensors we used in our study

1) . Heart Rate Sensor

The Heart Rate Sensor (e.g., MAX30102, Pulse Oximeter Sensor) measures the patient's heart rate and oxygen saturation levels. It helps in identifying abnormal physiological responses after a fall, such as sudden changes in heart rate. The sensor uses optical photoplethysmography (PPG) technology to detect blood volume changes in the fingertip or wrist. It is crucial for monitoring stress, fatigue, and potential cardiac issues in hospitalized patients. This sensor enhances real-time fall detection by providing physiological insights along with motion data.

2) Accelerometer

The Accelerometer (e.g., MPU6050, ADXL345) measures the linear acceleration of a patient’s body in different directions. It detects sudden movements, such as a fall, by analyzing acceleration peaks and free-fall events. The accelerometer provides data on body posture and movement patterns, allowing the system to classify fall-related activities. It is lightweight, power-efficient, and ideal for wearable IoT applications. The sensor’s three-axis motion tracking improves the accuracy of human activity recognition.

3) 3. Gyroscope

The Gyroscope (e.g., MPU6050, L3G4200D) measures angular velocity and rotational movements, helping in detecting postural instability. It captures balance changes and body orientation shifts, which are essential for differentiating falls from normal activities. The gyroscope works alongside the accelerometer to enhance motion classification accuracy. It provides real-time feedback on a patient's movement, helping caregivers in continuous monitoring. The sensor plays a crucial role in reducing false alarms by refining fall detection algorithms.

G. Transmission of Sensor Signals to Deep Learning Model for Classification

The process of sending sensor signals from the wearable sensors (accelerometer, gyroscope, and heart rate sensor) to the deep learning model involves multiple steps, including data acquisition, communication protocols, preprocessing, and classification.

1. Sensor Data Acquisition: The accelerometer, gyroscope, and heart rate sensor continuously collect motion and physiological data in real-time. These sensors generate raw data in the form of acceleration (m/s²), angular velocity (°/s), and heart rate (BPM) at predefined sampling rates (e.g., 50 Hz for motion sensors, 1 Hz for heart rate).
2. Communication Protocols: The collected sensor values are transmitted using IoT communication protocols such as Bluetooth Low Energy (BLE), Wi-Fi (MQTT/HTTP), or Zigbee to a local processing unit (e.g., Raspberry Pi, ESP32, or an IoT gateway). The selected protocol ensures efficient low-latency, real-time data transmission while maintaining energy efficiency in wearable devices.
3. Preprocessing and Data Formatting: The received raw sensor signals undergo noise filtering (e.g., Butterworth filter), normalization (Min-Max scaling), and segmentation (sliding window method) to prepare the data for deep learning analysis. The formatted data is structured into time-series feature vectors, which are then sent to the classification model.
4. Deep Learning Model Processing: The Graph Convolutional Recurrent Neural Network (GCRNN) takes the preprocessed data and extracts both spatial and temporal dependencies in the sensor readings. The graph convolutional layers capture relationships between different sensor nodes.

H. Classification Model

The GCRNN model is utilized for detection task in this work. Convolutional operations are key in ML, especially in CNNs, which outperform in image classification by processing multidimensional arrays. While CNNs focus on Euclidean structures, Graph Convolutional Networks (GCNs) adapt these operations to data, enabling effective node feature extraction from graph models. The output of a GCN method is typically computed as:

$$Y = \tilde{A}XW \qquad (1)$$

where, X represents the input data, Y characterises the output, and W is the parameter matrix of the model. Additionally, $\tilde{A}$ is the regularized adjacency matrix, which can be read as:

$$\tilde{A} = \left(\widehat{D}^{-\frac{1}{2}}\hat{A}\widehat{D}^{-\frac{1}{2}}\right) \qquad (2)$$

with $\hat{A} = A + I$, and where A is the upon the graph in which the links characterise the correlation between the different time series, I is the individuality matrix, and $\widehat{D}$ is the diagonal degree matrix of $\hat{A}$. A Recurrent Neural Network (RNN) predicts outcomes by using both input data and outputs from neighboring units, processing temporal data through connections that exploit previous states, particularly within the Hidden Layer (HL). Here, $x = (x_1, \ldots, x_t)$ is the input sequence, $y = (y_1, \ldots, y_t)$ is the output arrangement, and $h^n = (h_1^n, \ldots, h_t^n)$ represents the hidden vectors in layer n. Based on this, the overall hidden state h_t of the initial layer is computed as:

$$h_t^1 = \tanh\left(W_{xh^1}x_t + W_{h^1h^1}h_{t-1}^1 + b_n^1\right) \text{ (3)}$$

where W denotes the weighted matrix, W_{xh^1}. denotes the weights of the connection between the primary input layer, whereas $W_{h^1h^1}$. denotes the recurrent link weights in the initial HL and b_n^1 denotes the bias. The hidden states are the output sequence, and the layer n are computed as:

$$h_t^n = tanh\left(W_{h^{n-1}h^n}h_t^{n-1} + W_{h^1h^1}h_{t-1}^1 + b_n^1\right) \text{ (4)}$$

where $W_{h^{n-1}h^n}$ signifies the weight assigned to the connection between the n and $n-1$ layers, $W_{h^{n-1}h^n}$ characterises the weight allocated n-th layer, besides $b_{h_t}^n$ relative bias. The temporal output representation from the RNN is given by:

$$y_t = W_{h^ny}h_t^n + b_y \qquad (5)$$

where W_{h^ny} is the weight assigned to the connection among the output layer and the n layer and b_y is the bias. The basic recurrent units can suffer from the vanishing gradient problem, limiting the learning of LSTM units were introduced, with modified hidden state calculations, whereas (3-5) are still applicable.

$$f_t = \sigma\left(W_f h_{t-1}^n + w_f h_t^{t-1} + b_f\right) \text{ (6)}$$

$$i_t = \sigma\left(W_i h_{t-1}^n + w_j h_t^{n-1} + b_i\right) \text{ (7)}$$

$$C_t = f_t C_{t-1} + i_t tanh\left(W_c h_{t-1}^n + w_c h_t^{n-1} + b_c\right) \text{ (8)}$$

$$o_t = \sigma(W_o h_{t-1}^n + w_o h_t^{n-1} + b_o) \quad (9)$$

$$h_t = o_t \tanh(C_t) \quad (10)$$

where the adopted activation functions for the gates are represented by the sigmoid (σ) and the hyperbolic tangent ($tanh$). The conditions of the output, forget, and input gates are represented by f_t, i_t, C_t and o_t, respectively. In addition, the weights and the biases assigned to these gates are b_f, w_f, and W_f; b_i, w_i and W_i; b_c, w_c and W_c; and b_o, w_o and W_o. The GCRNN model combines RNNs and GCNs to capture the temporal and spatial features, with GCN layers extracting spatial data, LSTM layers modeling temporal aspects, and a dense layer producing the output.

I. Fine-tuning using Ninja optimizer (NiOA)

The framework uses the Ninja Optimizer (NiOA) [25], a high- meta-heuristic optimization technique, to improve the performance of the proposed GCRNN model. In order for NiOA to be effective, the network parameters must be adjusted to increase search space profit while lowering the chance of hitting local optima and raising the chance of reaching global optima. In addition to the relocation of their interactions with other, external random values, and their placements in earlier time, the fundamental components of NiOA are established by mathematical functions such as exponential and cosine wave functions.

To manage the exploration and exploitation procedures, NiOA depends on a set of control parameters. A random integer between 6 and 10 is one of these parameters, along with elements like v_1, r_2, r_3, J_1, J_2, and n, each of within a clear range to aspects of optimization. For instance, r_2 and r_3 manage the exploration phase's random movements, whereas J_1 and J_2 are in charge of tweaking it. By adjusting these settings, NiOA gains the flexibility it needs to tackle a variety of optimization problems.

J. Exploration phase

As indicated in Eq. (11) throughout the exploration phase, the agent's location (L_s) is updated using its current and previous positions as well as random search factors. By using a random element r_1 in the position update equation, the algorithm may calculate the difference between two places at different times, allowing it to explore new search space., t_1 and t_2. The method adds randomization by choosing a new place from a predetermined list if the conditions are not met, enabling the exploration process to proceed widely. Finding the global optimum is more likely thanks to this technique, which makes sure the search encompasses a large number of possible solutions.

$L_s(t+1) = \{L_s(t) + r_1.(L_s(t_1) - L_s(t_2))\}$, then, Random $L_s(t)$ and $\in Fs$ (11)

where Fs stands for the fitness answer, guaranteeing that the agent successfully investigates new positions. The site of another agent, D_s, is updated using a formula that adds a random component and periodic function. r_2. Because of this continuity, the agent that replaces fixed values can remain variable with γ, preventing situations in which the search is confined to local optima. As explained in Eq. (12), the cosine wave introduces cyclical activity that enables the procedure to go to new areas in a controlled way, resembling certain biological systems:

$$D_s(t+1) = D_s(t) + |D_s(t)| + r_2.D_s(t).\cos(2\pi t) \quad (12)$$

To facilitate a composite search technique, the search procedure incorporates the rationalised locations of Ls and Ds, as shown in Eq. (13):

$$S(t+1) = r_1.L_s(t+1) + r_2.D_s(t+1) \quad (13)$$

K. Exploitation phase

The emphasis switches from discovery to honing the already-found answers during the exploitation phase. The approach makes use of a parameterized non-linear equation.. J_1 and J_2 so that the agent can manage the level of exploitation and make small but noticeable progress in their position. Because the algorithm fine-tunes the outcome at this point, it is great for improving the phase. The optimizer is guided to identify local optimums within the search space by the exploitation equation's non-linearity, which, through modest incremental steps, drives the process towards the global optimum. As demonstrated in Equation (14) a non-linear equation governs the motion around the optimal solution:

$$M_s(t+1) = J_1.M_s(t) + 2.J_2.\left(M_s(t) + (M_s(t) + J_1)\right).\left(1 - \frac{M_s(t)}{M_s(t)+J_1}\right)^2 \quad (14)$$

To improve the optimization process even more, NiOA includes a system for updating resources or rewards. This update introduces periodic increase in the state using an growth classical that is controlled by a cosine function. Throughout the search, this recurring effect makes sure that the optimization procedure is flexible and able to react to shifting circumstances. The application of J_2 as a control parameter enables the reward updates to be adjusted, giving the algorithm an additional degree of accuracy, as shown in Eq. (15):

$$R_s(t+1) = R_s(t) + (1 + R_s(t) + J_2).\exp(\cos(2\pi)) \quad (15)$$

When the optimal solution remains unchanged after a number of repetitions (often three), the NiOA applies an update equation that includes numerous terms, such as the variation in the placements of the agents L_s and D_s, and the aids from M_s and R_s. By forcing updates to happen even when progress has slowed, optimizer doesn't stagnate. This update method is made more flexible by the addition of scaling factors i and n as well as the parameter, which enables the algorithm to modify its strategy in response to the optimization process's present state. The search around the solution syndicates the refined values of M_s and $R_s$for misuse, as designated in Eq. (16):

$$S(t+1) = J_1.M_s(t+1) + J_2.R_s(t+1) \quad (16)$$

The Ninja Optimizer (NiOA) is an adaptive optimization method that is built to fine-tune the GCRNNs in this outline. It is fast and efficient. To keep the perfect from being stuck and to aid it in managing a clearly split search area, NiOA employs a mix of random walks, fixed oscillations, non-linear changes. With the use of random characteristics including mutation methods, cosine, parameter updates, the model becomes resistant to local optima, quickly transitions to global optimum, and accurately predicts the parameters.

4) Mutation

The author introduces a mutation approach called NiOA to increase the process's diversity even further. Because of this, the agent's motion is affected by a non-linear mutation, and a summation equation is involved in which the signs shift. So that the amplitude of the mutations created in two repetitions varies, the mutation parameter an is evidently chosen at random from a range while keeping its sign. By doing so, .to can keep optimization from becoming overly deterministic and allow the procedure an opportunity to "break free" of local optima, opening up new and perhaps more fruitful areas of the search space to explore.

According to Eq. (17), a mutation method can introduce variety by altering the present solution based on several factors if the solution does not recover after three iterations.

$$s(t+1) = L_s(t+1) + i.n.(L_s(t+1) - D_s(t+1)) + i.n(M_s(t+1) + 2.r_1.R_s(t+1)) \quad (17)$$

Lastly, as shown in Eq. (18), the parameters controlling the phases guarantee the algorithm's flexibility and effectiveness:

$$r_1 \in [0,1], r_2 \in [0,1], J_1 \in [0,2], J_2 \in [0,2], i \in [0,1], n \in [0,2] \quad (18)$$

IV. RESULTS AND DISCUSSION

This section provides a thorough description of experimental settings and evaluation metrics used in the proposed approach. Experiments were showed on Kaggle using a PC equipped with 4GB RAM anda2.50 GHz i5-4200M CPU. TensorFlow2.5.0, a deep learning framework, was utilized in these experiments to construct the proposed model. The model was built Rygen 5 processor besides utilized inbuilt graphics card. The dataset is split randomly using sci-kit learn library's train test split (), 80% for training and 20% for testing. The NiOA algorithm optimizes the network, and dropout enhances generalization. A softmax classifier in the last dense layer calculates the final output result.

L. Validation Analysis of proposed optimizer with existing techniques

In this section, Validation Analysis of proposed optimizer with existing techniques as Extreme Learning Machine (ELM), Concolution Neural Network, Recurrent Neural Network Proposed model.

Table 1: Experimental Analysis of various models on patient behaviour measures

Perfect Name	Sum of Epochs	Accuacy	Precision	Recall	F1 Score
Extreme Learning Machine	10	92.78	92.78	93.54	93.78
Concolution Neural Network	10	95.43	95.43	96.43	94.45
Recurrent Neural Network	10	96.90	96.90	97.89	97.42
Proposed model	10	98.93	98.53	98.43	98.93

A comparative analysis of different machine learning and deep learning models used for patient behavior recognition and fall detection based on key performance metrics: accuracy, precision, recall, and F1-score. Each model was trained for 10 epochs, ensuring a fair comparison of their learning capabilities and classification performance.

Extreme Learning Machine (ELM): This model achieved an accuracy of 92.78%, with a precision of 92.78%, recall of 93.54%, and an F1-score of 93.78%. While ELM provides a fast training process, it may lack the ability to learn complex temporal dependencies in patient activity data.

Convolutional Neural Network (CNN): The CNN model demonstrated an improved accuracy of 95.43%, with a precision of 95.43%, recall of 96.43%, and an F1-score of 94.45%. CNN effectively captures spatial patterns from sensor data but struggles with long-term dependencies in time-series data.

Recurrent Neural Network (RNN): The RNN model further improves performance, reaching an accuracy of 96.90%, with precision (96.90%), recall (97.89%), and F1-score (97.42%). RNN captures sequential dependencies in patient movements, making it more effective than CNN for fall detection. However, standard RNNs suffer from vanishing gradient issues, limiting their ability to retain long-term information.

Proposed Model (GCRNN + Ninja Optimizer): The proposed model outperforms all other models, achieving the highest accuracy (98.93%), with precision (98.53%), recall (98.43%), and F1-score (98.93%). By integrating Graph Convolutional Recurrent Neural Network (GCRNN) and the Ninja Optimizer, the model effectively learns both spatial and temporal dependencies from sensor data, resulting in more reliable fall detection and patient monitoring. The fine-tuning capability of the Ninja Optimizer helps reduce overfitting and improves the model's generalization performance. Deep learning models (CNN, RNN, and GCRNN) outperform traditional methods like ELM due to their ability to learn complex feature representations.

RNN-based models show superior performance over CNN, confirming the importance of temporal modeling in patient activity recognition.

The proposed GCRNN model with Ninja Optimizer achieves the highest accuracy and F1-score, proving its effectiveness in real-time IoT-based fall detection systems.

These results highlight the efficiency and reliability of the proposed approach, making it a suitable solution for smart healthcare and patient safety applications.

V. CONCLUSION

This research presents an IoT-enabled fall detection system that integrates wearable sensors and a deep learning model to enhance patient safety in hospital environments. The proposed approach utilizes accelerometers, gyroscopes, and heart rate sensors to continuously monitor patient movements and physiological data. The collected signals are processed using a Graph Convolutional Recurrent Neural Network (GCRNN), which effectively captures both spatial and temporal dependencies in human activity patterns. To further enhance model accuracy and efficiency, the Ninja Optimizer is employed for fine-tuning, ensuring faster convergence, reduced overfitting, and improved generalization. Experimental results on the KFall dataset demonstrate that the proposed GCRNN model with Ninja Optimizer outperforms traditional machine learning and deep learning models, achieving higher accuracy, precision, recall, and F1-score. Additionally, IoT-based real-time data transmission enables continuous monitoring and timely intervention in case of falls. The findings of this study highlight the effectiveness of the proposed system in smart healthcare applications, offering an efficient and reliable solution for fall detection and patient monitoring.

M. Future scope

Future research can explore several enhancements to further improve the accuracy, robustness, and adaptability of the proposed fall detection system. Firstly, multimodal sensor integration involving temperature, blood pressure, and ECG sensors can provide deeper physiological insights for more precise fall detection and health monitoring. Secondly, implementing edge AI on low-power devices such as Raspberry Pi, Jetson Nano, or specialized AI chips can improve real-time processing efficiency, reducing dependency on cloud servers. Additionally, adaptive deep learning models with self-learning capabilities can be developed to dynamically adjust to patient-specific movement patterns, ensuring personalized monitoring. Moreover, the system can be extended to work with 5G-enabled IoT infrastructure for low-latency, high-speed communication between patient devices and hospital control centers. Lastly, incorporating explainable AI (XAI) techniques can enhance model interpretability, ensuring trust and reliability in medical decision-making. These advancements will further strengthen smart healthcare solutions, making fall detection systems more accurate, scalable, and widely deployable in real-world hospital environments.

REFERENCE

[1] Paul SK, Miah AS, Paul RR, Hamid ME, Shin J, Rahim MA. IoT-Based Real-Time Medical-Related Human Activity Recognition Using Skeletons and Multi-Stage Deep Learning for Healthcare. arXiv preprint arXiv:2501.07039. 2025 Jan 13.

[2] Bibbò L, Carotenuto R, Della Corte F. An overview of indoor localization system for human activity recognition (HAR) in healthcare. Sensors. 2022 Oct 23;22(21):8119.

[3] Serpush F, Menhaj MB, Masoumi B, Karasfi B. Wearable sensor-based human activity recognition in the smart healthcare system. Computational intelligence and neuroscience. 2022;2022(1):1391906.

[4] Bibbò L, Vellasco MM. Human activity recognition (har) in healthcare. Applied Sciences. 2023 Dec 6;13(24):13009.

[5] Qi J, Yang P, Waraich A, Deng Z, Zhao Y, Yang Y. Examining sensor-based physical activity recognition and monitoring for healthcare using Internet of Things: A systematic review. Journal of biomedical informatics. 2018 Nov 1;87:138-53.

[6] Hossain T, Ahad MA, Inoue S. A method for sensor-based activity recognition in missing data scenario. Sensors. 2020 Jul 8;20(14):3811.

[7] Baghezza R, Bouchard K, Bouzouane A, Gouin-Vallerand C. From offline to real-time distributed activity recognition in wireless sensor networks for healthcare: A review. Sensors. 2021 Apr 15;21(8):2786.

[8] Guerra BM, Torti E, Marenzi E, Schmid M, Ramat S, Leporati F, Danese G. Ambient assisted living for frail people through human activity recognition: state-of-the-art, challenges and future directions. Frontiers in neuroscience. 2023 Oct 2;17:1256682.

[9] Lin HC, Chen MJ, Lee CH, Kung LC, Huang JT. Fall recognition based on an IMU wearable device and fall verification through a smart speaker and the IoT. Sensors. 2023 Jun 9;23(12):5472.

[10] Paramasivam A, Jenath M, Sivakumaran TS, Sankaran S, Pittu PS, Vijayalakshmi S. Development of artificial intelligence edge computing based wearable device for fall detection and prevention of elderly people. Heliyon. 2024 Apr 30;10(8).

[11] Al-qaness MA, Dahou A, Abd Elaziz M, Helmi AM. Human activity recognition and fall detection using convolutional neural network and transformer-based architecture. Biomedical Signal Processing and Control. 2024 Sep 1;95:106412.

[12] Chen Z, Wang Y, Yang W. Video based fall detection using human poses. InCCF Conference on Big Data 2021 Jan 8 (pp. 283-296). Singapore: Springer Nature Singapore.

[13] Paul SK, Miah AS, Paul RR, Hamid ME, Shin J, Rahim MA. IoT-Based Real-Time Medical-Related Human Activity Recognition Using Skeletons and Multi-Stage Deep Learning for Healthcare. arXiv preprint arXiv:2501.07039. 2025 Jan 13.

[14] Paramasivam A, Jenath M, Sivakumaran TS, Sankaran S, Pittu PS, Vijayalakshmi S. Development of artificial intelligence edge computing based wearable device for fall detection and prevention of elderly people. Heliyon. 2024 Apr 30;10(8).

[15] Zhang Y, Tang H, Wu Y, Wang B, Yang D. FMCW Radar Human Action Recognition Based on Asymmetric Convolutional Residual Blocks. Sensors. 2024 Jul 15;24(14):4570.