IntNet: Lightweight yet High-Performance Deep Learning System for Intuitive Radar Patterns Analysis and Human Fall Detection

Malek Almallah ¹ and Belal Sababha ²

¹Princess Sumaya University for Technology
²Affiliation not available

December 9, 2023
Abstract—The growing trend of solitary living among the elderly and young, coupled with the high risk of falls leading to injuries and death, highlights the need for fall monitoring systems. Emphasizing individuals’ privacy and comfort, these systems should rely on radar sensors instead of visual-based, acoustic-based, or wearable solutions. Current radar-based systems are yet to reach satisfactory real-world performance. This work proposes a radar-based fall detection system with superior performance in complex real-world scenarios while maintaining edge computing capabilities and minimum hardware resources. The proposed deep learning system achieved a recall of 98.99% and a precision of 99.32%. These unprecedented performance numbers are measured on the proposed dataset, which is the most real-life representative dataset in the literature. The system has 211.8k parameters and ~8.84 M Floating Point Operations (FLOPs), achieving an edge computing capability. Moreover, the efficient model construction eliminates redundant computation in real-time operation. Furthermore, this work proposes a novel metric that encompasses all dataset’s quality aspects into a single number, which can be applied to all classification problems. This metric can then be used as a correction factor for performance metrics to put them in the context of the dataset used for testing.

Index Terms—Fall detection, millimeter-wave radar, frequency modulated continuous wave radar, deep learning, convolutional neural network, long short-term memory, data quality, data integrity.

I. INTRODUCTION

It is estimated that by the year 2050 the world’s population aged above 65 will be more than 1.5 billion compared to 700 million in 2019 [1]. These people are in great danger to face a fall accident, which is one of the main cases of injuries and death [2]–[6]. The fall chance for a person aged 60 and above is at least 20.8%, which increases to at least 33.2% for people aged above 80 [7]. Moreover, 29% of community-dwelling adults reported a fall accident once a year at least [8]. The problem can be further escalated by including young adults who are facing a similar risk [9]. There is a great need for an intelligent system that can monitor daily movements and detect fall accidents to ensure timely medical treatment [10]. While current wearable [11]–[14] approaches do not ensure a person’s comfort nor guarantee full-time monitoring, visual [15]–[18] and acoustic [19] approaches raise privacy concerns [10], [20], [21]. Alternatively, radar sensors can give a detailed description of the observed environment while addressing the issues of the other methods [22]–[25].

All reported radar-based fall detection systems consider simple life scenarios [26], that is, a single person who is asked to perform one movement in a specific position, angle, and timing (i.e., in a deterministic way), as shown in Fig. 1 (a), while considering few movements and fall patterns [27]–[37]. However, real-world movements are highly random and have countless patterns, suggesting that any deterministic method to collect the data is not sufficient, and a vast number of movements must be considered. There were attempts to collect diverse and complex datasets by considering large numbers of non-fall and fall patterns [38], [39]. However, they still follow the same deterministic approach for designing the experiment. They also consider one movement at a time and ignore some complex scenarios, such as transitioning from standing up to suddenly looking under a table before looking around for something and sitting back again in a short time, as shown in Fig. 1 (b), which can be recognized by the radar as a single event if done quickly. If not, then the combination of movement endpoints and start points captured by the sliding window in real-time, or the quick transition between movements’ start points due to hesitation, presents a significant yet unaddressed challenge that can confuse the detection system and increase false alarm rates. Furthermore, the case of having guests was not considered in these attempts. While detecting falls is most important in the case of a single subject because there is no one to provide aid, the case of multiple subjects, such as when having guests, must be considered in non-fall samples to ensure that the system will not confuse it with fall, thus increasing false alarm rates.

Radar-based fall detection systems must be lightweight,
compact, and trustworthy. This is to ensure edge computing and independence from network latency and reliability while being able to detect fall accidents efficiently in complex real-world scenarios. Current systems, despite being designed for simple life scenarios, claim high performance but reach high hardware and/or software complexity [38]–[40]. On the other hand, some low-complexity systems were proposed, but they suffer from low performance in complex real-world and even in ideal scenarios [41], [42]. This unoptimized performance-complexity balance results from approaching the radar classification problem similar to approaching image classification problems [43]. Images contain sharp edges and tiny details that need a high-complexity classifier to understand. However, radar outputs are typically in the form of heatmaps, which do not contain as many details as images. This difference can be demonstrated by applying an edge detection filter on images and heatmaps, as in Fig. 1 (c). Moreover, these systems mostly rely on convolutional neural networks [44] (CNNs) to capture both spatial and temporal information, despite CNNs being less suitable for temporal data. This results in an inefficient increase in the classifier’s size to capture time-varying behaviors [40].

Additionally, in fall detection problems, an event is captured by converting the radar raw data into a sequence of heatmaps, where each heatmap interprets the subject’s posture at a specific time step, and the entire sequence interprets the movement. Current fall classifiers encompass this sequence in a volume of tens of heatmaps [38], [39] and feed it to a CNN with a filter depth equal to the number of heatmaps in that volume. However, in real-time implementation, events are sent to the classifier repeatedly and rapidly compared to the sequence length to ensure reliability, and thus these events will largely overlap. In other words, the sliding window technique will be applied. Therefore, a single heatmap will be processed by the classifier multiple times, resulting in redundant computations, as shown in Fig 1 (d). In some approaches, the whole sequence is lumped into one heatmap [45]–[49], but the same concept is applied on radar raw data frames, as shown in Fig 1 (e), leading to the same issue.

In this work, we propose a new field-specific approach toward designing deep learning classifiers for radar pattern classification. This approach suggests using a combination of dual branch CNN to estimate a person’s posture and recurrence neural network (RNN) to analyze posture’s time variation. This efficient approach reduces the classifier size to reach an edge computing level complexity with minimum hardware requirements. Yet, it guarantees an outstanding performance in complex real-world scenarios. Moreover, the
resulting classifier eliminates redundant computation significantly because the CNN part will process each heatmap independently only once without overlapping events, as shown in Fig 1 (f). We also designed a stochastic data collection protocol to achieve a real-world complexity level and ensure meaningful system verification. We asked the subjects to summarize part of their daily routine in front of the radar for 5 to 15 minutes. Then, the radar long record was cropped into small samples that largely overlap. Thus, the combination of movements endpoints and start points, and rapid behaviors such as in Fig 1 (b), are all considered in the proposed dataset. For fall samples, all possible angles, positions, and complex fall patterns were considered. Therefore, the proposed system is trained and verified using the highest quality data yet in the literature. Still, it achieves the highest reported performance while keeping the computation requirements and hardware complexity at a minimum level. Furthermore, because measuring the quality of the dataset in the field of radar-based fall detection is not obvious, we propose a novel metric that encompasses all quality aspects of a dataset in a single parameter. This parameter can then be used as a correction factor for performance metrics (i.e., recall and precision) to put them in the context of the used dataset’s quality. This metric can be used in all classification problems.

The detailed contribution of this work is summarized as follows:

1) A new neural network architecture called Intuitive Neural Network (IntNet) is developed based on an intuitive understanding of the radar heatmaps' spatial and temporal patterns. IntNet achieves the best performance-complexity balance compared to the literature.

2) The system is designed and tested considering the highest quality dataset in the literature that emulates real life.

3) The proposed system achieves the best performance compared to state-of-the-art, with a recall of 98.99% and a precision of 99.32%.

4) The proposed system has 211.8k learnable parameters with a time complexity of ~8.84 MFLOPs per heatmap, which is considered a lightweight model when compared to the literature.

5) The proposed model processes each radar heatmap only once in real-time operations.

6) Minimum hardware setup with only two antennas.

7) A novel metric that measures all data quality aspects and represents them in a single number that can be used as a correction factor for performance metrics to put them in the context of the used dataset. This performance correction methodology is significant for all classification problems in engineering and computer science.

The rest of this paper is organized as follows. Section II reviews the literature. The experimental setup, data collection protocol, and signal processing algorithm are discussed in Section III. The deep neural network architecture is presented in Section IV. In Section V, experimental results are discussed, and the performance correction methodology is proposed. Finally, the paper is concluded in Section VI.

II. LITERATURE REVIEW

One of the first attempts to build a high-performance and robust system for fall detection was presented by Y. Yao et al [39]. The system uses a frequency modulated continuous wave (FMCW) radar with a frequency range of 5.46 – 7.24 GHz with vertical and horizontal antenna arrays, each of which has 12 antennas with a total number of 24 antennas. This complex Multi-Inputs Multi-Outputs (MIMO) system is used to accurately calculate the Angle of Arrival (AoA) and then construct vertical and horizontal heatmaps that localize objects in the environment. Then, a sequence of 144 frames of vertical and horizontal heatmaps enters a CNN-based classifier as two volumetric images. The height and width of these images are those of the heatmaps, while the number of channels is 144. A sliding window technique was used to process long data, leading to significant reprocessing for a single frame. The system can capture some complex fall patterns, including slow falls and fast non-falls. However, the achieved recall and precision were 93.8% and 91.9%, respectively, even though the deep neural network is large, and the radar system is complex and bulky. Furthermore, the used dataset is large and considers a wide variety of movements, but it still follows a deterministic approach and considers one movement at a time.

Sadrezzam et al. [40] proposed a high-performance model that uses a 5.9–10.3 GHz Ultra-Wideband (UWB) radar with only two antennas. The system applies a Short-Time Fourier Transform (STFT) on the radar signal to construct a time-frequency heatmap, which is then discretized into a binary image and passed to an enormous CNN. While this study achieved a high precision of 98.37% and a moderate recall of 94.37%, it has been trained and tested on an oversimplified dataset that contains only 5 types of movements. Moreover, the deep learning model used in this study has ~72 million parameters with ~817 MFLOPs for a single heatmap, which is considered a very large model and hard to deploy on a small device. The system also suffers from redundant processing for the radar raw data in real-time monitoring.

The use of time-frequency heatmaps, as in [40], associates fall events with high-frequency peaks (i.e., fast movements) and non-fall events with flat frequency peaks (i.e., slow movements). However, there are several slow falls, such as falling due to a heart attack or grabbing a stick or a table while falling, which will slow down the fall. Also, there are many fast non-fall actions, such as looking under the bed quickly, running, jumping, etc. Thus, the use of time-frequency analysis limits the performance of the system [50], even with a very large and complex classifier. For example, in [27], the proposed neural network has 1.223M parameters and a time complexity of 1.45 GFLOPs but achieves only 95% recall even though the dataset is very simple. Even with small and well-designed architecture, as in [49], the time-frequency analysis results in 96.80% accuracy. Additionally, time-frequency analysis will result in a redundant computation when applying the sliding window technique in real-time.

End-to-end deep learning approaches have also been used, where the deep learning model takes radar raw data directly
without any engineering, which requires the model to learn its own engineering method. Thus, these approaches often require enormous neural networks to achieve a good performance. In [51], an end-to-end approach was applied with a moderate-size model, and even though the paper claims a good performance, it only considers 8 motions. Thus, no real validation for the system was made. The same argument can be applied to [41], where the study proposed the lightest model yet in literature at the cost of dropping the F1 score to 90.98%, considering only a simple dataset for testing. Moreover, in [42], a lightweight model with 259K parameters was developed and achieved 97.6% recall, but the dataset was simple and contained only 9 movements. On the contrary, some previous works [28] used a high level of engineering for the radar signal, where the range, velocity, and AoA are extracted and processed to generate a 3D point cloud localization for movements. Skeleton representation is then constructed using a clustering algorithm. An accuracy of 95.75% was achieved on a simple dataset, and although a low model complexity was achieved, the complexity was moved to the radar and signal processing part.

In [38], a diverse and large dataset was used, and a high recall and precision of 98.3% and 97.5%, respectively, were achieved. The radar signal was used to construct range-velocity, range-horizontal angle, and range-vertical angle maps, not a frequency-time heatmap like in most previous work. Therefore, it can detect slow fall and fast non-fall patterns. These heatmaps are then processed by three large parallel CNNs, followed by an information fusion technique for the three outputs. However, using three heatmaps instead of one increases the number of antennas from two to seven and therefore increases the radiated power and system complexity. Furthermore, two branches of CNN were used for the two additional heatmaps, tripling the complexity of the model. Most importantly, the study still follows a deterministic approach and considers only one movement at a time, reducing real life representation. Also, a single heatmap will be processed by the system multiple times when using the sliding window technique.

It can be noticed by following the ongoing discussion that the major three gaps in the previous works are: 1) Systems are designed for simple life scenarios only. 2) Systems achieve high performance at great cost in terms of software and hardware complexity. 3) The redundant computation in real-time operation. These three gaps are addressed in this study.

III. METHODS

A. Radar System

In this study, an FMCW radar is used instead of UWB radars because it can report both the velocity and the location of the movement simultaneously. It is called an FMCW radar because its working principle is based on sending a linear frequency-modulated sinusoidal signal called “chirp” for a short period “$T_{chirp}$”. This chirp can be represented in terms of frequency versus time, as shown in Fig. 2 (a) left, which is a more suitable representation to understand the functionality of the radar, or in terms of amplitude versus time, as shown in Fig. 2 (a) right. The difference between the final frequency and the starting frequency “$f_0$” of the chirp is called the chirp’s bandwidth “BW”.

The specific FMCW radar module used in this study is Infineon Technologies 60GHz XENSIV BGT60TR13C radar [52] which is shown in Fig. 2 (b). The radar module is shown in the red box, while the rest of the board is for development purposes and interfacing. The radar has four Antennas in Package (AIP), which can be seen in the red box, where three are for receiving and only one for transmission. Only one receiver is used in this work, achieving the minimum number of antennas. Furthermore, it can be noticed that the form factor of the radar module is extremely small, with a size of 6.5 x 5.0 x 0.9 mm$^3$, thanks to the AIP technology and the high operation frequency. Thus, this radar is suitable for compact edge systems.

Fig. 2 (c) shows the radar block diagram. First, the phase locked loop (PLL) is controlled by the frequency control word (FCW) to generate a chirp with specific characteristics for $T_{chirp}$ period. Then, the chirp is directed using the Power Splitter in two directions, to the transmitter (Tx) and to the Mixer. The Tx sends the chirp, the object reflects it back, and the receiver (Rx) receives the echo, which is an exact replica compared to the original chirp in terms of frequency variation with time. This echo is then amplified using a Low-Noise Amplifier (LNA), which will reduce the signal-to-noise ratio as well. The mixer receives the original chirp at the exact time it was sent, and it receives the echo after a roundtrip delay “$t_\lambda$”, which is much less than $T_{chirp}$ practically, making the two signals largely overlap in time, as shown in Fig. 2 (d). It can be noticed that the two signals are identical in terms of slope, where the slope value is known by design.

The mixer generates a signal that has an intermediate frequency (IF) tone “$f_d$” that is equal to the frequency difference between the two inputs, which is constant in time, as shown in Fig. 2 (e). It also generates a high-frequency tone that is equal to the frequency sum of the two inputs, but this tone is filtered out by the Low-Pass Filter (LPF). Finally, $f_d$ is directly related to the distance “$d$” at which the object is located, as shown in (1), where $c$ is the speed of light.

$$d = \frac{f_d T_{chirp}}{2cBW}$$

Because there are multiple objects in the environment and thus multiple reflections, the LPF output has multiple frequency tones corresponding to each object. Therefore, to extract the distance for all objects at the same time, the output of the filter is sampled using an Analog to Digital Converter (ADC), where the output of the ADC is referred to as Radar Raw Data. Then, an FFT algorithm can be applied to the raw data to extract the frequency spectrum and thus the IF frequency tone for each object, as shown in Fig. 2 (f) where one object is detected. This FFT operation is called range-FFT. By using (1), the distance for each object can be then extracted.

A single chirp is needed to detect the distance for all objects
in the environment. But to determine the velocity $v$ of these objects, $N$ chirps must be sent repetitively. Then, by observing the phase variation for each IF tone (i.e., for each object) between chirps, the Doppler effect can be extracted, and the velocity for each object can be calculated using (2), where $T$ is the time between the two chirps, $\omega$ is the phase shift, $\lambda$ is the wavelength.

$$v = \frac{\lambda \omega}{4\pi T} \quad (2)$$

This can also be done efficiently by applying an FFT operation on each frequency tone between chirps, which is called doppler-FFT.

**B. Signal Processing Algorithm**

The radar sends $N$ chirps separated by a time $T$ and receives the echo for each. While doing that, the raw data is organized in a matrix, as shown in Fig. 2 (g), where each chirp is sampled into $M$ samples and stored in a different row. With $N$ chirps, the final dimension of the matrix is $N \times M$. Then, a new matrix will be constructed, and so on. Each matrix is called a radar frame, and it will describe the instantaneous posture of the observed moving objects after applying the signal processing algorithm. Therefore, a sequence of radar frames describes the temporal change in the posture, as if the radar outputs a video. It is worth mentioning that in the radar frame, the $M$ dimension is referred to as the “fast time axis,” while the $N$ dimension is referred to as the “slow time axis.”

Once a frame is constructed, the signal processing algorithm is applied to extract the range-velocity heatmap. The algorithm is described and visualized in Fig. 2 (h)-(k). The first step is removing the effect of static objects such as walls, chairs. This step is vital, as it eliminates the effect of the environment and helps with model generalization. Fig. 2 (h) shows a three-dimensional (3D) representation of the radar frame before any processing. Because most reflected power comes from static objects, it can be noticed that the same behavior dominates all chirps, and thus the reflected power from moving objects will be minor. To remove static objects, for each slow time axis, the average value for this axis is calculated and subtracted from all the points on that axis as in [38]. The resulting matrix is shown in Fig. 2 (i).
The next step is applying the range-FFT along the fast time axis to achieve the frequency spectrum of the raw data and then using (1) to translate it into a range-time map, as shown in Fig. 2 (j). It can be noticed that two objects were detected in this case, and they appear as two peaks along an entire slow time axis. Then, the doppler-FFT is applied along the slow time axis to extract the phase information and then the velocity using (2). Fig. 2 (k) shows the final output, which is the range-velocity map. In this case, there is more than one speed for each IF tone. This can happen if there are two people standing at the same distance from the radar and moving at opposite or different velocities. However, this also could happen because body parts move at different or opposite instantaneous velocities. For instance, if a person rotates to look backward, then one shoulder will move forward while the other shoulder will move backward. The final step is to remove the zero-speed pin, which will hold no information due to static object removal in the first step.

C. Experimental Settings

The radar was placed in the experiment’s environment at around 180 cm height and tilted down by around 20 degrees, as shown in Fig. 3. The reason for tilting the radar is to capture both vertical and horizontal components of the movements. In each environment, an error was introduced intentionally to the height and the tilting angle to add a level of randomness to the data collection process. A summary of the radar configurations and settings is shown in Table I. A chirp is sampled to 128 samples, with 64 chirps per frame. 10 frames were captured per second, and by setting the sample length to 5 seconds, the total number of data points in a single data sample is $128 \times 64 \times 50$. The radar maximum range is set to 3.3 meters, and the half power beamwidth (HPMW) for the used radar on the horizontal axis (the E-plane) is 80 degrees, making this system able to work in a standard bedroom size or a bathroom.

The radar was connected to a personal laptop with an Intel Core i7-11800H processor with 8 cores, NVIDIA GeForce RTX 3060 with 6 GB memory, and 32 GB of RAM. The communication between the radar and the computer was through a Hi-Speed USB 2.0 interface. The raw data was captured and then stored on the laptop. Then the signal processing algorithm and the classification were done on the laptop using MATLAB offline.

D. Data Collection Protocol

The dataset was collected using two methods. For non-fall actions, we asked the subjects to try to summarize their different daily routines in 5-15 minutes while the radar was recording. All the conducted experiments to collect non-fall recordings are summarized in Table II. The recorded routines include morning routine, night routine, kitchen routine, living room routine, cleaning, cooking, office routine, taking care of a pet, workout, prayer, and more. The case of having guests could increase the number of false alarms, and it was not included in the literature before. In this study, the case of having 1-5 guests is also recorded. All these recordings have been conducted on several subjects. Then the recordings have been cropped into 5-second-long samples with a sliding window of 1 second to capture between movements, not just the movement itself, and rapid behaviors. Fifteen subjects aged 18-54 were involved in the study. The experiments were conducted in 12 environments, with 2 reserved for testing.

Previous systems that recorded sample by sample did not consider that in real-time operation, the sliding window will capture the end of the movement and the beginning of another. Thus, if the system was not trained in such cases, it might increase the false alarm rate in real-time operation. The proposed protocol addresses this limitation and ensures real life representation because subjects are doing their daily life normally without any instructions.

For fall actions, the same subjects and environments were involved, and the same methods used in most literature were used here as well, where a record of 5 seconds was recorded for a specific action. However, each single fall pattern was performed on a different random angle and position, not in a systematic way like in previous work, to make the data stochastic. We considered complex patterns of falls, as summarized in Table III, that cover a wide variety of real life fall scenarios.

The final dataset size is 19384 samples, 1260 of which are fall samples. Considering the complexity of the fall patterns, the large number of fall samples, the randomness in the direction and the place when performing the movements, and
TABLE I
NON-FALL DATA COLLECTION EXPERIMENTS

<table>
<thead>
<tr>
<th>Exp. No.</th>
<th>Environment</th>
<th>Subjects No.</th>
<th>Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Living Room 1</td>
<td>1</td>
<td>Entering the room, sitting on the couch, writing on a notebook, scrolling through a tablet, lying on the couch while checking the tablet, close the window, reopen it, flip while lying on the couch, sleep, wakeup, stand up, check the drawer under the table, grab something and leave.</td>
</tr>
<tr>
<td>2</td>
<td>Living Room 2</td>
<td>1</td>
<td>Entering the room carrying some food, sit on a couch, eat, hair combing, checking phone, go to the couch, sleep, turnover while sleeping, hold something and leave.</td>
</tr>
<tr>
<td>3</td>
<td>Living Room 3</td>
<td>2</td>
<td>Subject 1 enters the room, prays while sitting, checking the phone, drink tea, make a phone call. Subject 2 enters the room (guest), and they start chatting, watching TV, laughing, using remote control.</td>
</tr>
<tr>
<td>4</td>
<td>Kitchen</td>
<td>2</td>
<td>Subject 1 enters the kitchen, wash dishes, clean the floor, clean the table, cooking, make tea, leave. Subject 2 enters to make a sandwich, drink water, dance while eating, searching for something and then leave.</td>
</tr>
<tr>
<td>5</td>
<td>Corridor 1</td>
<td>1</td>
<td>Organizing indoor plants, water the plants, checking their soils, clean a mirror, organize photos on the wall.</td>
</tr>
<tr>
<td>6</td>
<td>Guests Room 1</td>
<td>3</td>
<td>A family meeting between three subjects, start arguing, subject A stands up to organize his clothes and sit back, subject B start pointing to the ceiling, subject C shows them something on her phone.</td>
</tr>
<tr>
<td>7</td>
<td>Guests Room 2</td>
<td>6</td>
<td>Celebrating a birthday, singing, cutting the cake, start eating in a different position.</td>
</tr>
<tr>
<td>8</td>
<td>Bedroom 1</td>
<td>1</td>
<td>Sitting on a desk, eat while sitting on the desk, open the door slowly, leave, close the door slowly, cleaning the desk, through a bag, through cushion and start organizing them, checking the phone, put legs on the dick while sitting on the chair, lower them back quickly.</td>
</tr>
<tr>
<td>9</td>
<td>Laboratory</td>
<td>10</td>
<td>Each subject individually did a combination of the following: Stretching, warm up, cardio, yoga, walking, walking while holding a stick, limping, sitting on a chair, making a phone call while swinging the chair, jumping, cleaning the floor and see a bug and jump back, catch something on the floor, droop and sleep on the floor, rise back, sit on the floor, rise back, move something heavy, move something light, sit on a chair, stand up.</td>
</tr>
<tr>
<td>10</td>
<td>Living Room 4</td>
<td>1</td>
<td>Cleaning the room, clean the table, drop down quickly to clean under the table, organize flowers, move something.</td>
</tr>
<tr>
<td>11</td>
<td>Bedroom 2</td>
<td>1</td>
<td>Islamic prayer and the radar observed it from different angles.</td>
</tr>
<tr>
<td>12</td>
<td>Bedroom 3</td>
<td>1</td>
<td>Start walking while holding things to organize them. Lay on the bed and check a tablet.</td>
</tr>
</tbody>
</table>

TABLE II
NON-FALL DATA COLLECTION EXPERIMENTS

<table>
<thead>
<tr>
<th>Fall Patterns</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lose consciousness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tripped over something</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fall while holding a stick</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due to insufficient power</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due to failure jump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slip</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stand up and fall</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fall from a sitting place</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>From kneeling position</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hit something and fall</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Due Heart attack</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE III
FALL DATA MOVEMENTS

<table>
<thead>
<tr>
<th>Fall Patterns</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lose consciousness</td>
<td>733</td>
<td>296</td>
<td>231</td>
</tr>
<tr>
<td>Tripped over something</td>
<td>12208</td>
<td>5220</td>
<td>696</td>
</tr>
</tbody>
</table>

TABLE IV
THE COLLECTED DATASET AND HOW IT WAS DIVIDED

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall Samples No.</td>
<td>733</td>
<td>296</td>
<td>231</td>
</tr>
<tr>
<td>Non-Fall Samples No.</td>
<td>12208</td>
<td>5220</td>
<td>696</td>
</tr>
</tbody>
</table>

Fig. 4. Samples from the dataset. (a) Lose consciousness and then fall. (b) Grasping a stick while falling, which slow down the fall. (c) Trap over something and then fall. (d) Taking care of indoor plants. (e) Eating while sitting. (f) Checking the tablet while setting in bed. (g) Office work. (h) Chatting with one guest. (i) Through something. This action is considered a fast non-fall action that can be detected as fall when using time-frequency analysis. (j) Writing while sitting. (k) Making tea. (l) Pray.
the stochastic method of collecting non-fall actions, this is the best-reported dataset in the literature. The dataset was collected in two phases. In the first phase, 18457 samples were collected, including 1029 falls. This dataset was split into training and validation datasets, with a ratio of ~77% for training and ~23% for validation. In the second phase, 927 samples were collected in two new environments, with 50% of the subjects being new to test the system generalization. The number of fall samples in the second phase was 231. The numbers of samples per class in the training, validation, and test sets are shown in Table IV. The dataset is made available as described in the “Data and Code Availability” section.

Fig. 4 shows twelve photos that were taken when collecting the dataset. For example, Fig. 4 (a) and Fig. 4 (b) show complex fall patterns that could happen in real life. Fig. 4 (c) shows a slow fall pattern, where the subject is grasping a stick and trying to resist the fall using this stick. On the other hand, Fig. 4 (i) shows someone throwing something, which is considered a fast non-fall pattern.

IV. DEEP LEARNING ARCHITECTURE

A. Methodology and Intuition

Fig. 5 (a) shows a high-resolution range-velocity heatmap generated using a high ADC sampling rate and a large number of chirps. When comparing it to the cat image shown in Fig. 5 (b), it can be noticed that the radar heatmap does not contain the same number of details as in the cat image. The comparison is more obvious when comparing the edge detector results for the two images, as shown in Fig. 5 (c) and (d), respectively. The heatmap is black for the most part, and most of the information about the person’s instantaneous posture comes from the yellow bubbles and their locations in the heatmap. A single filter in the CNN layer can learn to detect simple patterns, such as these bubbles. Therefore, using a CNN with only a few filters per layer to process the heatmap should be sufficient to fully analyze it. On the contrary, when designing a CNN to process the cat image, it is often necessary to use many filters per each convolutional layer, because the number of details and patterns in the cat image are large [53], [54].

Therefore, this study suggests that using only a few filters per CNN layer, and thus a small size CNN, to capture the person’s instantaneous posture from the radar heatmap should be sufficient in principle. The reason why most previous radar-based fall detection systems use large-size CNN is because of the poor construction of the architecture and because the development process was not guided by radar intuition. Rather it was following the same methodology followed in image classification problems. Additionally, they relied on CNN technologies to capture the temporal variation between radar frames. Here, we construct a CNN that is inspired by human intuition for understanding radar heatmaps. This CNN is responsible for encoding the instantaneous posture into a vector. Then, we use LSTM to analyze the temporal pattern for a sequence of instantaneous posture vectors to classify the observed movement as fall or not-fall.

The proposed neural network architecture is shown in Fig. 6. Each radar frame will be processed using two parallel CNN branches, each of which has its own architecture. The upper branch consists of three convolution layers with large filter sizes and only 3 filters per layer to detect large patterns and behaviors such as the ones shown in Fig. 5 (a). The lower branch consists of 3 convolution layers with only 3 filters per layer, as well as 2 max pooling layers. It has small filter sizes to detect small patterns such as the ones shown in Fig. 5 (a). The output of the two CNN branches will be concatenated and further processed using two fully connected layers (FCs).

The radar recording is a video, where each frame (i.e., the range-velocity map) gives the posture of the observed person, and the sequence of frames gives the movement. Therefore, each frame will be processed alone using the two branches and the FC layers to give a vector representation of the person’s posture. Then, because RNNs are suitable for capturing time-varying behaviors, the posture vector is processed using an RNN. Thus, each frame will be processed by the CNN part only once, reducing computation requirements, and then the RNN will process each posture vector in real-time. The RNN architecture used in this model is Long-Short-Term-Memory (LSTM) because it is suitable to capture long and short time-varying behaviors such as fast and slow falls.

The proposed system has ~8.84 MFLOPs per frame, and since the frame repetition time is 0.1 sec. This means that the
The proposed system needs a device that can perform \( \sim 88.4 \) MFLOPS per second, which is considered an edge performance level. For example, NVIDIA Jetson Nano [55] (an edge AI development kit) is capable of 472 GFLOPS per second, and thus the proposed model will use only \( \sim 0.019\% \) of its capabilities. Moreover, the proposed system can be deployed on Arduino Portenta H7 [56], which has much lower power consumption and computational capabilities compared to NVIDIA Jetson Nano. While the last one is considered a minicomputer, the first one is considered an embedded device. Arduino Portenta H7 has a dual-core processor (STM32H747) that supports floating-point arithmetic and is powered by a graphics engine. This processor has a clock frequency of 480 MHz, and with its single-cycle and single instruction/multiple data (SIMD) capabilities, it can support up to 480 MFLOPs per second. Thus, the proposed model will consume only 18.42\% of the processor capabilities in theory. Nevertheless, the proposed model has 211.8k learnable parameters. Assuming each parameter is stored as a 32-bit floating-point number (float32), this model has a size of 0.808 megabytes (MB), which can be fitted in an embedded device.

**C. Model Training**

Because the fall detection problem is a classification problem, the cross-entropy loss function, defined in (3), was used to validate the model output, where \( t \) is the class label, and \( p \) is the likelihood given to this label by the neural network. The model was trained on an NVIDIA GeForce RTX 3060 with 6 GB memory GPU. Adam optimizer [57] was used to optimize the model, with a mini-batch size of 50 samples. Moreover, to ensure a stable convergence of the model, a learning rate drop factor technique was used. The initial learning rate was set to 0.01 with a drop rate of 0.1 applied every 5 epochs. The data was shuffled after each epoch to reduce overfitting. After several tuning cycles of the training hyperparameters, the retrieved model was at epoch 13.

\[
\text{Loss} = -[t \cdot \log(p) + (1 - t) \cdot \log(1 - p)]
\]  

(3)
Therefore, the area under the curve (AUC) for the Receiver Operating Characteristic (ROC) curve, where the optimal value is when AUC is equal to one. The ROC for the proposed system on the validation and test sets are shown in Fig. 7 (c) and (d), respectively. The AUC for both curves is approximately 0.9999, indicating outstanding discrimination between classes and further room for decreasing the model in future work.

B. Proposed Benchmark

To compare the proposed work with the previous work quantitatively, there is a need to put the performance in the context of the used dataset. This is because achieving high performance in a simple and small dataset is meaningless, and the model might perform poorly in the real world. Therefore, it is essential to find a way to correct the performance metrics according to the quality of the dataset. In other words, there must be a correction factor based on the dataset quality, with a maximum theoretical value of one and a minimum theoretical value of zero. Then, this correction factor must be multiplied by the reported performance metrics and thus adjust their values based on the used dataset.

To do that, this study introduces the following new metrics. To measure data quality (DQ) and represent it in a single parameter, \( (4) \) is introduced, where \( NF \) is the number of movement types considered for non-falls, \( F \) is the number of movement types for falls, and \( N \) is the dataset size.

\[
DQ = \frac{2 \cdot \tan^{-1} NF}{3\pi} + \frac{2 \cdot \tan^{-1} F}{3\pi} + \frac{2 \cdot \tan^{-1} N}{3\pi}
\]

(4)

The building block of \( (4) \) is \( (5) \). This function was chosen because when looking into its curve in Fig. 8 (a), it goes to one when \( x \) goes to infinity. This is an important characteristic because the goal is to design a correction factor with a maximum possible value of one. Also, the value \( y \) will not
increase significantly when \( x \) is larger than approximately 15, while it will drop quickly to zero when \( x \) is below approximately 5. This is also important because when designing a correction factor equation, there must be a range where a practically achievable value of \( x \) can be considered good enough, which is 5-15 in this case. Additionally, in (4), the three terms are divided by 3 to make their sum go to one in the ideal case. Thus, if all \( NF, F, \) and \( N \) go to infinity, then the data quality goes to one. If these numbers are too small, then the data quality will drop towards zero quickly.

\[
y = \frac{2 \cdot \tan^{-1}(x)}{\pi}
\]

Equation (4) can be further generalized as in (6), where each term can be given a different weight in the total sum. The parameters \( \alpha_{nf} \) and \( \alpha_f \) are used to give a weight to \( NF \) and \( F \), respectively, and the sum of these two variables must be less than or equal to one. The remaining weight is assigned to \( N \), and therefore, if the sum of \( \alpha_{nf} \) and \( \alpha_f \) is equal to one, then \( N \) will have no effect on the total sum and will not contribute to the performance correction. In other words, the sum of all three weights of \( NF, F, \) and \( N \) shall exactly be equal to one.

\[
DQ = \alpha_{nf} \cdot \frac{2 \cdot \tan^{-1}(NF)}{\pi} + \alpha_f \cdot \frac{2 \cdot \tan^{-1}(F)}{\pi} + (1 - \alpha_{nf} - \alpha_f) \cdot \frac{2 \cdot \tan^{-1}(N)}{\pi}
\]

It is important to note that the good-enough range in equation (5) is approximately between 5 and 15. However this range is not always suitable, such as for the number of samples in the dataset \( N \), where it should be much more than 15 samples to consider the dataset as good. The good-enough range can be adjusted by scaling \( NF, F, \) and \( N \) by an appropriate value. Therefore, (6) can be further generalized to be as (7), where \( \beta_{nf}, \beta_f, \) and \( \gamma \) are real numbers used as scaling parameters for the good-enough range of \( NF, F, \) and \( N \), respectively. To make the good-enough range higher than 5 to 15, then the scaling parameter shall be decreased below one, while to make it lower, then the scaling parameters shall be increased.

\[
DQ = \alpha_{nf} \cdot \frac{2 \cdot \tan^{-1}(\beta_{nf} \cdot NF)}{\pi} + \alpha_f \cdot \frac{2 \cdot \tan^{-1}(\beta_f \cdot F)}{\pi} + (1 - \alpha_{nf} - \alpha_f) \cdot \frac{2 \cdot \tan^{-1}(\gamma \cdot N)}{\pi}
\]

For the sake of the comparison done in this work, \( \alpha_{nf} \) and \( \alpha_f \) were both set to 1/3, giving the same weight for \( NF, F, \) and \( N \). Additionally, \( \beta_{nf} \) and \( \beta_f \) were both set to one, keeping the good-enough range between 5 and 15. Although more than 15 actions must be considered to make the dataset represent real life, especially for non-fall sample, the good-enough range is kept below 15. This is to avoid putting some previous works at a great disadvantage, putting the comparison slightly in the favor of the literature. Yet, the proposed dataset achieves the highest quality score. Moreover, the size of the dataset must
The proposed benchmarking and performance correction methodology can be used for all classification problems. Assuming a cat-dog classification problem, each class must have many subclasses, such as breeds, animal position, skin color, etc. In this case \( NF \) can be replaced by the number of subclasses for the cat class, while \( F \) can be replaced by the number of subclasses for the dog class. Then, (7) and (8) can be used.

**C. Comparison with the Literature**

A comparison between the proposed model and state-of-the-art systems in terms of model complexity, corrected performance, precision, recall, and dataset size, is shown in Table VI. As it can be seen, while the proposed model is more complicated compared to [41], it outperformed all previous work in terms of corrected performance, and even in terms of precision and recall without any correction. Fig. 8 (b) shows a virtual representation for this comparison, where the horizontal access is the model complexity indexed from 0 to 4, with 4 being the highest system complexity and 0 being the optimal system complexity. For the proposed system the complexity index is equal to one. On the other hand, the vertical access is 100 divided by the corrected performance. Therefore, the best system would achieve a value of one, while the worst system would reach infinity. Therefore, the optimal point is (0,1), which represents the best performance-complexity balance. It can be noticed that the proposed system is by far the closest system to the optimal point compared to the literature.

**VI. Conclusion**

In this study, a radar-based fall detection system was proposed. The system uses a radar-specific intuitive approach to build a deep learning model with outstanding real-world performance. It achieved the best reported performance compared to the literature, with a recall of 98.99%, precision of 99.32%, and F1 score of 99.15%. Yet, the model achieves a low complexity with only 211.8k parameters and ~8.84 MFLOPs per radar frame. The system replaces the sliding window technique usage in real-time operation by the feedback mechanism in LSTM. Thus, the system eliminates redundant computations. First, at each time step, two branches of CNN process a single radar frame only once to estimate the posture of the person and encode it into a vector. Then, the LSTM will process this vector directly at the same time step, and it uses its feedback to capture the temporal relationships between frames without any need to re-process previous frames. An FMCW radar with a frequency range of 60 GHz – 63 GHz was used in this study. Only two antennas were used, achieving the minimum possible antenna configuration. The dataset was collected in a stochastic protocol to reflect real life diversity and complexity and to ensure real life performance validation for the proposed system. The proposed dataset’s quality outperforms previously proposed datasets, and it contains complex behaviors such as movements endpoints and start points combination, and rapid movements sequence in a single sample. To measure the dataset quality, and for a fair comparison between fall detection systems, a novel data quality metric and performance correction methodology were proposed, which can be used in all classification problems.

**ACKNOWLEDGMENT**

The authors would like to thank Mr. Almallah family and all Princess Sumaya University for Technology undergraduate students who voluntarily participated in the data collection process.

**DATA AND CODE AVAILABILITY**

The dataset, after applying the signal processing algorithm and being separated, is made publicly available through (https://www.kaggle.com/dss26470). Moreover, the pre-trained model and the model before training are made publicly available through (https://github.com/malekmallah/InfNet.git). This is to ensure a fast model development over this work by fellow researchers and to ensure quick results duplication.

**REFERENCES**


Malek Y. Almallah received his MSc degree from Princess Sumaya University for Technology (PSUT), Amman, Jordan, in 2023, and was supervised by Prof. Belal H. Sababha. Since September 2019, he has been with Golden Electronics, Amman, Jordan, as a Research and Development Engineer, working on developing the next generation of semiconductor foundation IPs. Prior to that, he worked as an undergraduate Research Assistant with Prof. Yanal S. Faouri at the University of Jordan during his bachelor’s degree. His research interests lie at the intersection of electronics design, signal processing, and deep learning.

Belal H. Sababha (Senior Member, IEEE) is a Professor of Electrical and Computer Engineering and a Drones & Embedded Systems Consultant. He has been with Princess Sumaya University for Technology (PSUT) since 2012. Dr. Sababha is a US patent holder for three granted patents and served as a Chair for two international IEEE conferences. Prior moving to academia, Dr. Sababha has worked in the Automotive Industry. He worked as a Senior Controls Engineer in the Powertrain Controls department at Chrysler Group LLC, Michigan USA. Received his PhD degree in Electrical and Computer Engineering – Embedded Systems from Oakland University, MI, USA in 2011. He has taught electrical and computer engineering undergraduate and graduate courses at various universities in the US and Jordan. Dr. Sababha has extensive experience in embedded systems design, control algorithm design and software development with applications related to Gasoline Engine Controls and Unmanned Aerial Vehicles (UAVs). He is a consultant in the fields of Embedded Systems and UAV design and control for various governmental and commercial firms. His research concentration areas are UAV development and control, Biomedical instrumentation, embedded sensors, embedded RTOS and CAN networks, distributed embedded systems, graceful degradation in embedded systems, rapid prototyping, machine vision, and artificial intelligence. Dr. Sababha has served in several senior leadership positions as a dean for two terms, Acting Executive Dean, Associate Executive Dean, Director, and Department Chair. He is a Senior Member of IEEE and a member of several national and international professional organizations.