Baltic J. Modern Computing, Vol. 12 (2024), No. 3, pp. 327–348
https://doi.org/10.22364/bjmc.2024.12.3.07

Accuracy vs Complexity: A Small Scale Dynamic Neural Networks Case

Martynas DUMPIS[1], Dalius NAVAKAUSKAS[2]

[1] Vilnius Gediminas Technical University, Plytines g. 25-234, Vilnius LT-10105, Lithuania
[2] Vilnius Gediminas Technical University, Sauletekio al. 11-1103, Vilnius LT-10223, Lithuania

martynas.dumpis@vilniustech.lt, dalius.navakauskas@vilniustech.lt

ORCID 0009-0003-4335-372X, ORCID 0000-0001-8897-7366

Abstract. This research provides a detailed analysis of small-scale dynamic neural network (NN) models for human activity recognition using data from smartphones. We evaluate eight dynamic NN: Finite Impulse Response (FIRNN), Infinite Impulse Response (IIRNN), Gamma Memory (GMNN), Lattice Ladder (LLNN), Time Delay (TDNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRUNN), and Long Short-Term Memory (LSTMNN), utilizing a publicly available dataset from Kaggle. The study focuses on comparing these models in terms of higher accuracy, smallest scale, adaptivity to the task (walking vs running classification), and memory utilization. Different NN architectures and synapse configurations are evaluated by their accuracy and computational complexity. The findings reveal which NN architectures offer the best performance while being the least computationally and memory demanding. Among the models, the IIRNN achieved the highest accuracy at 99.86% in the recognition of specified activities. Additionally, the TDNN model demonstrated impressive performance with 99.27% accuracy while requiring fewer computational resources: 2 binary additions, 2 multiplications, and 2 activation functions.

Keywords: Small-Scale Neural Networks, Time-Varying Signals, Smartphone Sensor Data, Human Activity Recognition; Accelerometer

1 Introduction

Recognition of human activities such as walking and running using smartphone sensor data plays a crucial role in advancing health and fitness applications. This capability holds significant promise for healthcare monitoring, athletic training, and lifestyle management. Additionally, integrating Human Activity Recognition (HAR) with IoT technologies paves the way for innovative smart healthcare solutions, demonstrating the convergence of wearable technology and health monitoring (Gomaa and Khamis, 2023). This synergy is beneficial for developing smart cities and personalized health monitoring systems, thereby expanding the influence of HAR technologies (Serpush et al., 2022). Furthermore, evaluating cognitive-mental abilities such as reaction times, focus, and anticipation through digital solutions can significantly enhance athletic performance and healthcare outcomes, showcasing the importance of integrating cognitive assessments with physical health monitoring (Butkevičiūtė et al., 2023).

This study focuses on a comparative analysis of small-scale dynamic NN models, evaluating their performance, computational requirements, and memory usage in processing accelerometer data. Unlike studies that concentrate on classifying specific activities such as walking and running, this research aims to compare various NN architectures, including TDNN, FIRNN, IIRNN, GMNN, LLNN, RNN, GRUNN, and LSTMNN. Each network is assessed for its efficacy in accurately processing time-sequenced activity data and its adaptability to the task. *The central question* here addressed is what the least complex dynamic NN architectures are and their features that allow one to accurately enough classify walking and running activities using accelerometer data.

The article is structured as follows: the introduction is followed by a review of related work to situate the research within the existing literature. Section on the proposed investigation technique presents the dataset preparation, NN selection, NN models, their complexity, and training procedure. The investigation results encompass investigation coverage and findings on the four main NNs investigation aspects: recognition accuracy, computational complexity, adaptability to the task, and memory utilization. Finally, the implications of the findings are discussed.

2 Related Work

Time-variant data, characterized by its changes over time, is a critical component in numerous research fields. Studies in this area utilize diverse datasets and applications, including HAR, financial forecasting, audio data analysis, medical data, environmental monitoring, and visual data processing.

HAR uses sensor data to recognize human activities, essential for applications in health monitoring and mobile health apps. This field has grown significantly due to the implementation of various models of NNs that enhance the accuracy and efficiency of activity recognition (Kumar et al., 2024). Financial forecasting involves predicting stock prices and other economic indicators using historical financial data (Li et al., 2022). Audio data applications, such as speech recognition, have has played a pivotal role in enabling us to adapt to novel modes of communication: it not only empowers individuals with disabilities to interact, share knowledge and engage in open conversations, but also holds promise for revolutionizing communication between machines using natural languages (Kasparaitis and Antanavičius, 2023; Al-Fraihat et al., 2024).

In the medical domain, time-variant data includes longitudinal patient health records, sensor data from medical devices, and various diagnostic data. These datasets are used to predict and monitor health conditions such as Alzheimer's Disease, diabetes, heart failure, and Parkinson's Disease (Alhudhaif, 2024; Davidashvilly et al., 2024). Environmental data, including air quality and meteorological data, parking data, and Cal-trans

Traffic Performance Measurement System data, is crucial for monitoring and forecasting purposes (Zhang et al., 2024; Weerakody et al., 2021).

Various mathematical models have been developed to handle time-variant data, each with distinct strengths and limitations. Traditional machine learning models such as generative models (maximum a posteriori, Gaussian mixture or hidden Markov) and standard machine learning models (support vector machine, random forest, k-nearest neighbors), linear and autoregressive models (linear regression, autoregressive, autoregressive moving-average, autoregressive conditional heteroskedasticity) have been extensively used. However, these models often fall short in capturing long-term dependencies due to their lack of memory functions, these models are more sensitive to short-term relationships than long-term dependencies, which can not capture some important recurring features (Hamzacebi et al., 2019; Jiang et al., 2020). Generative models, for example, require expert knowledge and preprocessing of text for Automatic Speech Recognition (ASR), making them less flexible than end-to-end ASR models that rely on paired acoustics and language data (Hari et al., 2017).

In recent years, deep learning techniques, particularly recurrent NNs, have gained prominence due to their ability to handle long-term dependencies in sequential data. RNNs, LSTMNN, and GRUNN have demonstrated superior performance in various applications, including healthcare and finance (Kosar and Barshan, 2023). For instance, Deepcare model, which uses Diabetes and Mental Health patient data, achieved a higher F-score (79) compared to traditional models like support vector machine (66.7) and random forests (71.4) (Pham et al., 2016; Weerakody et al., 2021). LSTMNNs, in particular, are known for avoiding long-term dependency issues, making them highly suitable for tasks requiring the recall of information over extended periods (Pham et al., 2016; Hochreiter and Schmidhuber, 1997).

GRUNNs are shown to be less computationally intensive compared to LSTMNNs due to their simpler architecture, resulting in faster training times. However, LSTMNNs generally achieve better predictive performance (Weerakody et al., 2021). Specifically, LSTMNNs demonstrated superior accuracy in capturing long-term dependencies within the data, making them more effective for complex sequence modeling tasks. Furthermore, innovations like Bi-directional GRUNN have improved the accuracy and reliability of HAR systems (Helmi et al., 2023). Additionally, RNNs have been employed in embedded systems for HAR, utilizing data from accelerometers and other sensors to achieve high accuracy, proving their efficiency in real-time applications within resource-constrained environments (Alessandrini et al., 2021). This study adopts streamlined versions of LSTMNN, GRUNN, and RNN architectures to directly compare their performance in HAR.

Primary models such as TDNN, FIRNN, IIRNN, GMNN, and LLNN have also been utilized for handling time-variant data. These models demonstrated good results by incorporating long-term dependencies of input signals but were not directly compared with latest RNNs, GRUNNs, or LSTMNNs. The inclusion of these primary models in this study allows for a comprehensive comparison against latest architectures. TDNNs, known for their implementation simplicity and ability to remember previous signal input values (Paliwal, 1991). FIRNN and IIRNN models excel in signal processing capabilities, while GMNNs are effective in managing long-range dependencies in sequences (Lawrence et al., 1995). LLNNs, with their unique lattice-ladder structure, offer flexibility and learning potential in dynamic environments, making them suitable for sound and image processing (Back and Tsoi, 1992; Navakauskas et al., 2014; Navakauskienė et al., 2021).

Convolutional Neural Networks (CNNs) are utilized in computer vision tasks. They are also used in HAR with datasets such as KU-HAR, UCI-HAR, and WISDM, showing good results with accuracies of 96.86%, 93.48%, and 93.89%, respectively (Akter et al., 2023). Additionally, models that mix CNNs with LSTMNNs have been effective in identifying a broad range of activities, reaching an accuracy of 90.89% (Khan et al., 2022). However, CNNs typically have pooling layers for down sampling, usually performing average or max pooling to reduce the feature maps spatial resolution (Dentamaro et al., 2024). Such a complexity of NN architecture makes it not suitable for this research due to small size CNNs inability to handle sequential dependencies effectively.

Transformer models (Vaswani et al., 2017) have shown impressive results in natural language processing and image processing, and are beginning to make progress in time series forecasting applications, but are not included in this study due to their complexity and resource-intensive nature. Transformers struggle with feature extraction in time-series data and have high memory requirements, making them unsuitable for small-scale NNs (Weerakody et al., 2021). Another drawback of typical transformers for very long sequences is their memory intensity. Taks needing 1000 s of timestep are particularly difficult due to their quadratic time complexity, which is higher than that of RNNs (Li et al., 2019). Despite advancements in attention mechanisms to address these issues, RNN-based models remain more practical for this study's scope and objectives.

Analysis of related work shows that from the perspective of dynamic NN models TDNN, FIRNN, IIRNN, GMNN, LLNN, LSTMNN, GRUNN, and RNN already are or have potential to be employed in embedded systems for HAR. Thus, it is important to evaluate their performance in recognition of human activities with high accuracy but keeping architectures at a small-scale. This study fills a gap in the literature by including primary dynamic models and comparing them directly with the latest RNN architectures. The development and optimization of these models contribute to a broader understanding of time-variant data applications, including agricultural monitoring and the integration of IoT systems for real-time data processing (Laktionov et al., 2023).

3 Proposed Investigation Technique

This section describes the methodology used for dataset preparation, the selection of dynamic NN models, the specifics and complexity of models, and their training process. The approach is structured to assess the performance of various NN architectures in classifying time-varying human activity data collected by smartphone sensors.

3.1 Dataset Preparation

The primary dataset used in this study is the KU-HAR dataset, obtained from Kaggle (Sikder and Nahid, 2021). This extensive dataset comprises 18 different activities recorded from 90 participants using smartphone sensors, such as accelerometers and gyroscopes. For the purposes of our research, we focused on accelerometer data corresponding to walking and running activities, which are crucial for HAR in health and fitness applications.

Dataset Characteristics. The KU-HAR dataset contains 1945 raw activity samples and 20,750 subsamples derived from these, captured in both controlled and uncontrolled settings. The activities range from static postures, like standing and sitting, to dynamic movements, such as walking and running. Each sample was carefully recorded to ensure the precision and reliability of the sensor data.

Data Processing. For this study, we used the time domain samples provided in the dataset, which includes 20,750 subsamples, each representing 3 s of non-overlapping accelerometer data. The sampling rate of 100 Hz was standartized across all samples withing all activities.

Magnitude of accelerometer x, y, and z axis coordinates was computed to prepare the data for input into NNs. Acceleration magnitude was calculated as follows:

$$u(n) = \sqrt{x(n)^2 + y(n)^2 + z(n)^2}. \tag{1}$$

Data Partitioning. The subsamples were split into training, validation, and testing sets with a ratio of 60%, 20%, and 20%. This division ensures that the NN models are thoroughly trained, validated for parameter tuning, and finally evaluated on new, unseen data to assess their generalizability and performance.

3.2 Neural Networks Selection

The selection of dynamic NNs for this study was guided by their distinct abilities to manage time-varying signals and intricate data structures. TDNNs were chosen for their simplicity in implementation and their capability to retain previous signal input values (Paliwal, 1991). FIRNN and IIRNN models were selected due to their strong performance in signal processing tasks. GMNNs were included for their proficiency in handling long-range dependencies in sequences, which is especially advantageous for time-varying signals (Lawrence et al., 1995). LLNNs were selected for their unique lattice-ladder structure, offering enhanced flexibility and learning potential in dynamic environments (Back and Tsoi, 1992). This structure has been widely applied in the analysis of sound and image processing (Navakauskas et al., 2014), and its adaptive capabilities have been demonstrated in the study of complex biological datasets in epigenetics (Navakauskienė et al., 2021). RNNs were incorporated due to their fundamental role in learning sequential dependencies within data, which is crucial for modeling cognitive tasks (McClelland and Rumelhart, 1987). GRUNNs, known for their efficiency in sequence modeling as highlighted by Cho et al. (2014), provide a simplified yet robust approach to temporal data processing. LSTMNNs were chosen for their ability to mitigate long-term dependency issues, making them highly suitable for tasks that require remembering information over long durations (Hochreiter and Schmidhuber, 1997).

3.3 Neural Network Models

The forward propagation in each NN model used in this study is specifically tailored to the experimental conditions and architectures employed here, rather than representing generic models. This customization is essential for understanding their operational mechanisms and predictive capabilities. In the following mathematical expressions describing NN models we denote by N_{I} the total number of NN inputs; N_{D} – the order of NN synapse (memory, filter, recurrency, etc.); N_{H} – the total number of NN hidden neurons; Φ_{HT} – the hyperbolic tangent activation function; Φ_{LS} – the logistic sigmoid activation function; $s^{(l)}(n)$ – the l-th layer recall at the time instance n (note, that $s^{(0)}(n)$ is NN input obtained by (1)).

The Output Layer Recall. For all eight selected NN models the output layer recall is calculated similarly and can be expressed by:

$$s^{(2)}(n) = \Phi_{\mathrm{LS}}\left(\sum_{h=1}^{N_{\mathrm{H}}} w_h^{(2)} s_h^{(1)}(n) - \bar{w}^{(2)}\right), \tag{2}$$

here $\boldsymbol{w}^{(2)}$ and $\bar{w}^{(2)}$ – the output neuron layer N_{H} weights vector and a single bias.

The Recall of Hidden Neurons. For each considered NN model the recall of hidden neurons is calculated separately and is provided below.

The hidden neurons of TDNN as synapses use time delay filters, thus the h-th hidden neuron recall is:

$$s_h^{(1)}(n) = \Phi_{\mathrm{LS}}\left(\sum_{i=1}^{N_{\mathrm{I}}} w_{ih}^{(1)} s^{(0)}(n-i) - \bar{w}_h^{(1)}\right), \tag{3}$$

here $\boldsymbol{W}^{(1)}$ and $\bar{\boldsymbol{w}}^{(1)}$ – $N_{\mathrm{I}} \times N_{\mathrm{H}}$ size weights matrix and N_{H} biases vector.

The hidden neurons of FIRNN as synapses use finite impulse response filters $\boldsymbol{w}_{ih}^{(1)}$, thus h-th hidden neuron recall is:

$$s_h^{(1)}(n) = \Phi_{\mathrm{LS}}\left(\sum_{i=1}^{N_{\mathrm{I}}}\sum_{j=1}^{N_{\mathrm{D}}} w_{ijh}^{(1)} s^{(0)}(n-j-i) - \bar{w}_h^{(1)}\right), \tag{4}$$

here $\boldsymbol{W}^{(1)}$ and $\bar{\boldsymbol{w}}^{(1)}$ – $N_{\mathrm{I}} \times N_{\mathrm{D}} \times N_{\mathrm{H}}$ size weights matrix and N_{H} biases vector.

The hidden neurons of IIRNN as synapses use infinite impulse response filters, thus the h-th hidden neuron recall is:

$$s_h^{(1)}(n) = \Phi_{\mathrm{LS}}\left(\sum_{i=1}^{N_{\mathrm{I}}} s_{ih}^{(1)}(n) - \bar{w}_h^{(1)}\right); \tag{5a}$$

$$s_{ih}^{(1)}(n) = \sum_{j=0}^{N_{\mathrm{D}}} b_{ijh}^{(1)}(n) s_i^{(0)}(n-j) + \sum_{j=1}^{N_{\mathrm{D}}} a_{ijh}^{(1)}(n) s_{ih}^{(1)}(n-j), \tag{5b}$$

here $s_{ih}^{(1)}(n)$ is the output of IIR filter; $b_{ijh}^{(1)}(n)$, $a_{ijh}^{(1)}(n)$ are coefficients of feedforward and recursive parts of IIR filter correspondingly; $\boldsymbol{W}^{(1)}$ and $\bar{\boldsymbol{w}}^{(1)}$ – $N_{\mathrm{I}} \times (2N_{\mathrm{D}}+1) \times N_{\mathrm{H}}$ size weights matrix and N_{H} biases vector.

The hidden neurons of GMNN as synapses use Gamma Memory (tuned by $\eta_{ih}^{(1)}$), thus h-th hidden neuron recall is:

$$s_h^{(1)}(n) = \Phi_{\mathrm{LS}}\left(\sum_{i=1}^{N_{\mathrm{I}}}\sum_{j=0}^{N_{\mathrm{D}}} w_{ijh}^{(1)} s_{ijh}^{(0)}(n) - \bar{w}_h^{(1)}\right); \tag{6a}$$

$$s_{ijh}^{(0)}(n) = \begin{cases} s^{(0)}(n-i-1), & j=0; \\ \eta_{ih}^{(1)} s_{i(j-1)h}^{(0)}(n) + \left(1-\eta_{ih}^{(1)}\right) s_{ijh}^{(0)}(n-1), & j \in [1, N_{\mathrm{D}}], \end{cases} \tag{6b}$$

here $s_{ijh}^{(0)}(n)$ – the output signal of the j-th tap of the Gamma Memory connecting i-th input with h-th neuron; $\boldsymbol{W}^{(1)}$ and $\bar{\boldsymbol{w}}^{(1)}$ – $N_{\mathrm{I}} \times (N_{\mathrm{D}}+1) \times N_{\mathrm{H}}$ size weights matrix and N_{H} biases vector.

The hidden neurons of LLNN as synapses use lattice-ladder filters (controlled by lattice $\boldsymbol{k}_{ih}^{(1)}$ and ladder $\boldsymbol{v}_{ih}^{(1)}$ parameters):

$$s_h^{(1)}(n) = \Phi_{\mathrm{LS}}\left(\sum_{i=1}^{N_{\mathrm{I}}}\sum_{j=1}^{N_{\mathrm{D}}} v_{ijh} b_{ijh}^{(1)}(n) - \bar{v}_h^{(1)}\right); \tag{7a}$$

$$\begin{cases} f_{ijh}^{(1)}(n) = f_{i(j-1)h}^{(1)}(n) + k_{ijh}^{(1)} b_{i(j-1)h}^{(1)}(n-1); \\ b_{ijh}^{(1)}(n) = b_{i(j-1)h}^{(1)}(n-1) - k_{ijh}^{(1)} f_{i(j-1)h}^{(1)}(n), \end{cases} \tag{7b}$$

for $j \in [1, N_{\mathrm{D}}]$, with such initial and boundary conditions

$$b_{i0h}^{(1)}(n) = f_{i0h}^{(1)}(n), \qquad f_{i(N_{\mathrm{D}}-1)h}^{(1)}(n) = s^{(0)}(n-i-1). \tag{7c}$$

Here $f_{ijh}^{(1)}(n)$ and $b_{ijh}^{(1)}(n)$ – the lattice forward and errors of backward prediction at the j-th tap, respectively; $\boldsymbol{K}^{(1)}$ – lattice $N_{\mathrm{I}} \times N_{\mathrm{D}} \times N_{\mathrm{H}}$ size weights matrix; $\boldsymbol{V}^{(1)}$ – ladder $N_{\mathrm{I}} \times N_{\mathrm{D}} \times N_{\mathrm{H}}$ size weights matrix; $\bar{\boldsymbol{v}}^{(1)}$ – N_{H} biases vector.

The hidden neurons of RNN together with forward connections use recurrent ones that store neurons hidden states:

$$s_h^{(1)}(n) = \Phi_{\mathrm{HT}}\left(\sum_{i=1}^{N_{\mathrm{I}}} w_{ih}^{(1)} s^{(0)}(n-i-1) + \sum_{j=1}^{N_{\mathrm{D}}} k_{jh}^{(1)} s_h^{(1)}(n-j) - \bar{w}_h^{(1)}\right), \tag{8}$$

here $\boldsymbol{W}^{(1)}$ – forward $N_{\mathrm{I}} \times N_{\mathrm{H}}$ size weights matrix and $\bar{\boldsymbol{w}}^{(1)}$ – N_{H} biases vector; $\boldsymbol{K}^{(1)}$ – recurrent $N_{\mathrm{D}} \times N_{\mathrm{H}}$ size weights matrix.

The neuron's state in *the hidden neurons of GRU* is changed with a candidate state $\tilde{s}_h^{(1)}(n)$, which is updated using the reset gate $s_{\mathrm{R}h}^{(1)}(n)$ and update gate $s_{\mathrm{U}h}^{(1)}(n)$ signals, thus h-th hidden neuron recall is expressed by:

$$s_h^{(1)}(n) = s_{\mathrm{U}h}^{(1)}(n) s_h^{(1)}(n-1) + \left(1 - s_{\mathrm{U}h}^{(1)}(n)\right) \tilde{s}_h^{(1)}(n); \tag{9a}$$

$$\tilde{s}_h^{(1)}(n) = \Phi_{\mathrm{HT}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) s_{\mathrm{R}h}^{(1)}(n) \right) - \bar{w}_h^{(1)} \right); \tag{9b}$$

$$s_{\mathrm{R}h}^{(1)}(n) = \Phi_{\mathrm{LS}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{\mathrm{R}ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) \right) - \bar{w}_{\mathrm{R}h}^{(1)} \right); \tag{9c}$$

$$s_{\mathrm{U}h}^{(1)}(n) = \Phi_{\mathrm{LS}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{\mathrm{U}ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) \right) - \bar{w}_{\mathrm{U}h}^{(1)} \right), \tag{9d}$$

here update gate, reset gate and candidate state are represented by: $\boldsymbol{W}_\mathrm{U}^{(1)}$, $\boldsymbol{W}_\mathrm{R}^{(1)}$, $\boldsymbol{W}^{(1)}$ – combined input and hidden state $N_\mathrm{I} \times N_\mathrm{D} \times N_\mathrm{H}$ size weights matrixes and $\bar{\boldsymbol{w}}_\mathrm{U}^{(1)}$, $\bar{\boldsymbol{w}}_\mathrm{R}^{(1)}$ and $\bar{\boldsymbol{w}}^{(1)}$ – N_H biases vectors, correspondingly.

The hidden neurons of LSTM is support gating of the hidden state. Input gate $s_{\mathrm{I}h}^{(1)}(n)$, forget gate $s_{\mathrm{F}h}^{(1)}(n)$, output gate $s_{\mathrm{O}h}^{(1)}(n)$, and input node $\tilde{s}_{\mathrm{C}h}^{(1)}(n)$ signals are used to construct memory cell internal state $s_{\mathrm{C}h}^{(1)}(n)$. Thus h-th hidden neuron recall is expressed by:

$$s_h^{(1)}(n) = s_{\mathrm{O}h}^{(1)}(n) \Phi_{\mathrm{HT}} \left(s_{\mathrm{C}h}^{(1)}(n) \right); \tag{10a}$$

$$s_{\mathrm{C}ijh}^{(1)}(n) = s_{\mathrm{F}h}^{(1)}(n) s_{\mathrm{C}h}^{(1)}(n-1) + s_{\mathrm{I}h}^{(1)}(n) \tilde{s}_{\mathrm{C}h}^{(1)}(n); \tag{10b}$$

$$\tilde{s}_{\mathrm{C}h}^{(1)}(n) = \Phi_{\mathrm{HT}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{\mathrm{C}ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) \right) - \bar{w}_{\mathrm{C}h}^{(1)} \right); \tag{10c}$$

$$s_{\mathrm{I}h}^{(1)}(n) = \Phi_{\mathrm{LS}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{\mathrm{I}ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) \right) - \bar{w}_{\mathrm{I}h}^{(1)} \right); \tag{10d}$$

$$s_{\mathrm{F}h}^{(1)}(n) = \Phi_{\mathrm{LS}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{\mathrm{F}ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) \right) - \bar{w}_{\mathrm{F}h}^{(1)} \right); \tag{10e}$$

$$s_{\mathrm{O}h}^{(1)}(n) = \Phi_{\mathrm{LS}} \left(\sum_{i=1}^{N_\mathrm{I}} \sum_{j=1}^{N_\mathrm{D}} w_{\mathrm{O}ijh}^{(1)} \left(s^{(0)}(n-i-1) + s_h^{(1)}(n-j) \right) - \bar{w}_{\mathrm{O}h}^{(1)} \right), \tag{10f}$$

here input, forget, output gate and candidate memory state are denoted by: $\boldsymbol{W}_\mathrm{I}^{(1)}$, $\boldsymbol{W}_\mathrm{F}^{(1)}$, $\boldsymbol{W}_\mathrm{O}^{(1)}$, $\boldsymbol{W}_\mathrm{C}^{(1)}$ – combined input and hidden state $N_\mathrm{I} \times N_\mathrm{D} \times N_\mathrm{H}$ size weights matrixes and $\bar{\boldsymbol{w}}_\mathrm{I}^{(1)}$, $\bar{\boldsymbol{w}}_\mathrm{F}^{(1)}$, $\bar{\boldsymbol{w}}_\mathrm{O}^{(1)}$ and $\bar{\boldsymbol{w}}_\mathrm{C}^{(1)}$ – N_H biases vectors, correspondingly.

3.4 Neural Networks Complexity

Table 1 provides an overview of the computational demands for eight selected NN models, detailing their operational requirements based on the total number of inputs (N_I), hidden neurons (N_H), and synapse order (N_D). The complexity of these dynamic NNs is illustrated by the total number of basic computing elements: bin. additions ($N_{2\Sigma}$), bin. multiplications ($N_{2\Pi}$), and act. functions (N_Φ). The highest values in each category are highlighted with a gray background.

Table 1. Dynamic NNs Complexity in Terms of the Total Number of Basic Computing Elements

NN	Total Number of Parameters*		
Model	$N_{2\Sigma}$	$N_{2\Pi}$	N_Φ
TDNN	$N_\mathrm{I}N_\mathrm{H} + N_\mathrm{H}$	$N_\mathrm{I}N_\mathrm{H} + N_\mathrm{H}$	$N_\mathrm{H} + 1$
FIRNN	$N_\mathrm{I}N_\mathrm{D} + N_\mathrm{I}N_\mathrm{H} + N_\mathrm{H}$	$N_\mathrm{I}N_\mathrm{D} + N_\mathrm{I}N_\mathrm{H} + N_\mathrm{H}$	$N_\mathrm{H} + 1$
IIRNN	$N_\mathrm{I}N_\mathrm{H}(2N_\mathrm{D} + 1) + 2N_\mathrm{H}$	$2N_\mathrm{I}N_\mathrm{H}N_\mathrm{D} + N_\mathrm{H}$	$N_\mathrm{H} + 1$
GMNN	$N_\mathrm{I}N_\mathrm{H}\left(N_\mathrm{D} + \frac{1}{2}(N_\mathrm{D}^2 - N_\mathrm{D})\right) +$ $N_\mathrm{I}(N_\mathrm{H} - 1) + 2N_\mathrm{H}$	$N_\mathrm{I}N_\mathrm{H}\left(\frac{1}{2}(N_\mathrm{D}^2 + N_\mathrm{D}) + 1\right)$	$N_\mathrm{H} + 1$
LLNN	$2N_\mathrm{I}N_\mathrm{H}N_\mathrm{D} + 2N_\mathrm{H}$	$N_\mathrm{I}N_\mathrm{H}(2N_\mathrm{D} + 2)$	$N_\mathrm{H} + 1$
RNN	$N_\mathrm{I}N_\mathrm{D} + N_\mathrm{I}N_\mathrm{H} + N_\mathrm{H}$	$N_\mathrm{I}N_\mathrm{D} + N_\mathrm{I}N_\mathrm{H} + N_\mathrm{H}$	$N_\mathrm{H} + 1$
GRUNN	$3N_\mathrm{H}N_\mathrm{D}(N_\mathrm{I} + N_\mathrm{D}) + N_\mathrm{D}$	$3N_\mathrm{I}N_\mathrm{H}(N_\mathrm{I} + N_\mathrm{D}N_\mathrm{H})$	$4N_\mathrm{D}N_\mathrm{H} + 1$
LSTMNN	$4N_\mathrm{H}N_\mathrm{D}(N_\mathrm{I} + N_\mathrm{D}) + N_\mathrm{D}$	$4N_\mathrm{I}N_\mathrm{D}(N_\mathrm{I} + N_\mathrm{D}N_\mathrm{H})$	$6N_\mathrm{D}N_\mathrm{H} + 1$

* By the gray background, the biggest values of $N_{2\Sigma}$, $N_{2\Pi}$ and N_Φ are outlined.

When examining the number of bin. multiplications and bin. additions, GMNN stands out for its complexity, reflecting its design to manage detailed temporal data, which may lead to superior performance in tasks requiring comprehensive historical trend analysis. LLNN also shows a high number of operations, particularly in bin. additions ($2N_\mathrm{I}N_\mathrm{H}N_\mathrm{D} + 2N_\mathrm{H}$), highlighting its ability to process data extensively in both forward and backward passes through the layers.

TDNN is characterized by its relatively low complexity, with bin. additions and bin. multiplications scaling linearly with the number of inputs and hidden neurons. This makes TDNN a good choice for simpler, real-time applications. FIRNN, with its higher order synapse filters, introduces more complexity than TDNN but remains efficient in terms of binary operations, making it suitable for tasks requiring finer temporal resolution. IIRNN, while more complex due to its infinite impulse response filters, balances its computational load with enhanced capability to model long-term dependencies, offering a middle ground between simplicity and detailed sequence handling.

The LSTMNN uses the highest number of act. functions ($6N_\mathrm{D}N_\mathrm{H}+1$), indicating its intricate design aimed at effective memory management. GRUNN, on the other hand, uses fewer act. functions ($4N_\mathrm{D}N_\mathrm{H}+1$), simplifying some aspects of LSTM's complexity while still maintaining strong sequence processing capabilities. GMNN, LLNN, and RNN utilize the same number of act. functions ($N_\mathrm{H} + 1$), suggesting these models are more straightforward and potentially more suitable for real-time applications like smartphone-based human activity recognition.

3.5 Neural Networks Training

The various NNs in this study were trained using gradient descent methods specifically tailored to their architectures.

TDNN introduces time delays deeper into the structure of NN, requiring modifications to standard NN training algorithms. As proposed by Waibel et al. (1989), the backpropagation (BP) algorithm is adapted for this purpose.

FIRNN training utilizes a variant of the BP algorithm specifically adapted for finite impulse response filters, referred to as temporal BP with gradient descent (Wan, 1990). The training involves updating FIR weights based on the gradient descent method, accounting for the delay elements within the network.

IIRNN training employs BP through time (BPTT) specifically tailored for IIRNN (Campolucci et al., 1999). This method involves unrolling the network through time for each sequence and updating the feedforward weights and IIR filter coefficients based on the gradients calculated throughout this temporal expansion, effectively handling the recursive components of the network.

GMNN employs stochastic gradient descent with BP, with updates applied to both network weights and the Gamma memory parameters (de Vries and Principe, 1992).

LLNN utilizes a simplified stochastic gradient descent with temporal BP, accounting for its lattice-ladder synapse structure (Navakauskas et al., 2014).

RNN uses BPTT (Werbos, 1990), unrolling the network through time for each sequence to update weights based on gradients computed across this temporal expansion.

Both *LSTMNN* and *GRUNN* also use BPTT (Vlachas et al., 2020). LSTMNN incorporates BP through structures that consist input, forget, and output gates, whereas GRUNN uses a similar approach but with simplified update and reset gates.

The Glorot/Xavier *weight initialization* method was adopted to maintain consistent training conditions and optimize weight scaling across different network architectures (Glorot and Bengio, 2010). For initializing hidden layer weights, it is:

$$\boldsymbol{W} = 2r\,\mathrm{rand}(N_\mathrm{I}, N_\mathrm{H}) - r; \tag{11a}$$

$$r = \sqrt{\frac{6}{N_\mathrm{I} + N_\mathrm{H}}}, \tag{11b}$$

here $\mathrm{rand}(\cdot)$ – a matrix of random numbers, in the range $[0,\ 1]$, generator; r – the range for the uniform distribution.

The training process was halted upon meeting *early-stopping criteria* such as when the maximum number of epochs was reached or when the validation loss ceased to improve over several epochs (Prechelt, 1998).

4 Investigation Results

This section presents the investigation coverage and the results of NNs accuracy and complexity evaluation in human activity recognition tasks.

4.1 Investigation Coverage

During the investigation, the size of the NN models was varied while maintaining a fixed three-layer setup.

For FIRNN, IIRNN, LLNN and GMNN, the number of inputs N_I ranged from 1 to 10, the synapse order N_D ranged from 1 to 10, and the number of hidden neurons N_H ranged from 1 to 10, resulting in $10 \times 10 \times 10 = 1000$ architectures for each model.

For TDNN, GRUNN, LSTMNN, and RNN, the synapse (recurrency) order N_D varied from 1 to 10, and the number of hidden neurons N_H ranged from 1 to 10, resulting in $10 \times 10 = 100$ architectures for each model.

Each NN architecture was initialized and trained 100 times to avoid suboptimal local minima. The architecture with the highest accuracy of these trials was used for further evaluation. In total, this resulted in 440,000 different NN implementations.

To analyze how training duration of various NN models differ, we investigated the distribution of the number of epochs required for each NN model to achieve the highest accuracy as shown in Fig. 1. This analysis was conducted over 100 training runs for each NN model. The IIRNN shows a high median around 140 epochs with significant variability, indicated by a tall box, suggesting less efficient training. The RNN has a low median of 3 epochs, with minimal spread and few outliers, highlighting its efficiency and stable training performance. Similarly, the LSTMNN exhibits a low median of 4 epochs with limited variability, reinforcing its efficient training process. The GRUNN shows the highest median around 180 epochs, indicating it requires the most epochs for training, accompanied by substantial variability and numerous outliers, reflecting poor training efficiency.

The TDNN has a median of 4 epochs with slightly higher variability than RNN and LSTMNN, yet still demonstrates efficient training. The FIRNN displays a low median of 5 epochs with slight variability, maintaining its status as one of the efficient networks. The LLNN shows a higher median around 10 epochs with considerable variability and many outliers, indicating less efficient training compared to others. Lastly, the GMNN

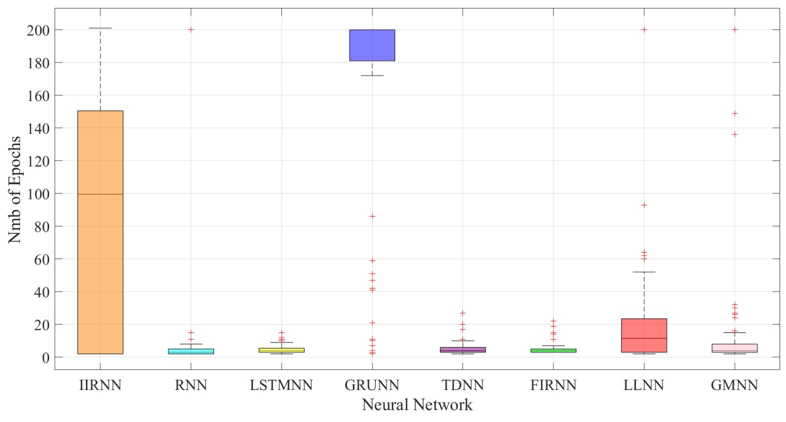


Fig. 1. Training duration analysis of eight dynamic NN models achieving highest accuracy during 100 runs.

network has a low median of 4 epochs with a compressed box towards the bottom, suggesting stable and efficient training performance.

4.2 Neural Networks Evaluation

A comprehensive evaluation of various NN models utilized for HAR was performed. Investigation was focused on the four main aspects: recognition accuracy, computational complexity, adaptability to the task, and memory utilization. Key assessment metrics included accuracy, the number of bin. additions, bin. multiplications, act. functions, weights, and delays, inputs, hidden neurons, and memory allocation size.

Highest Accuracy Neural Networks. Table 2 presents only those NNs that attained the highest accuracy across all architectures within each model. Some models having several entries with the same highest accuracy value are distinguished by additional subscripts. *The main objective* of the analysis is to identify architectures that not only achieve top accuracy (rows in the table are primarily sorted in decreasing accuracy order) but also balance computational efficiency, minimizing the number of bin. additions, bin. multiplications, and act. functions (complementary sorting of rows in the table in increasing number of parameters order).

Table 2. Highest Accuracy Achieving Dynamic Neural Networks and Their Complexity

Neural	Total Number of Parameters[**]					Highest
Network[*]	$N_{2\Sigma}$	$N_{2\Pi}$	N_{Φ}	N_{W}	N_{D}	ACC, %
IIRNN$_1$	612	549	10	5	603	99.86
IIRNN$_2$	644	595	8	5	637	99.86
IIRNN$_3$	920	850	11	7	910	99.86
RNN$_1$	14	14	5	6	44	99.77
RNN$_2$	19	19	7	7	84	99.77
TDNN$_1$	35	35	6	6	35	99.77
TDNN$_2$	40	40	6	7	40	99.77
TDNN$_3$	49	49	8	6	49	99.77
TDNN$_4$	63	63	10	6	63	99.77
GMNN	1503	1512	4	10	597	99.75
LLNN$_1$	1638	1782	10	10	189	99.75
LLNN$_2$	1820	1980	11	10	210	99.75
LSTMNN	490	480	61	10	490	99.66
FIRNN	35	35	6	5	80	99.58
GRUNN$_1$	196	189	29	7	196	99.55
GRUNN$_2$	306	297	37	9	306	99.55

[*] The gray background outlines the NNs with highest ACC or lowest parameter values.
[**]$N_{2\Sigma}$ – bin. additions; $N_{2\Pi}$ – bin. multiplications; N_{Φ} – act. func.; N_{W} – weights; N_{D} – delays.

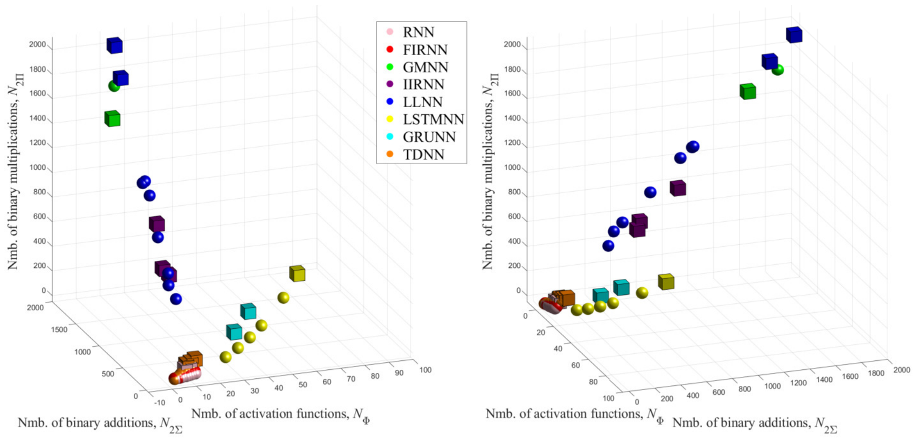
IIRNN$_1$, IIRNN$_2$, and IIRNN$_3$ all achieved the highest accuracy of 99.86%. However, they required a significant amount of computational resources. The least resource-intensive of the three still required 604 bin. multiplications, 549 bin. additions, and 10 act. functions. Although RNN_1 required the fewest bin. multiplications (14) and bin. additions (14) and achieved the second highest accuracy (99.77%). Although RNN_2 had similar accuracy, it needed marginally more computational resources, with 19 bin. multiplications and 19 bin. additions. GMNN and both LLNN configurations also reached high accuracy at 99.75%. Despite GMNN having the fewest act. functions (4), it required a large number of other parameters, with 1512 bin. multiplications and 1503 bin. additions, indicating significant computational demand. $LLNN_1$ and $LLNN_2$ also demanded extensive computational resources, with 1782 and 1980 bin. multiplications, and 1638 and 1820 bin. additions, respectively, demonstrating a trade-off between accuracy and computational load. Notably, LSTMNN achieved a high performance with an accuracy of 99.66%, but it required the most act. functions, totaling 61.

Close to the Highest Accuracy Neural Networks. Computational complexity data of an expanded range of NN architectures is presented in Fig. 2 on the next page. *The main objective* of the analysis is to relax the demands on accuracy (lowering acceptable accuracy level or looking for the simplest complexity NN architecture) to get insights on trade-off between computational complexity and accuracy across investigated NN models.

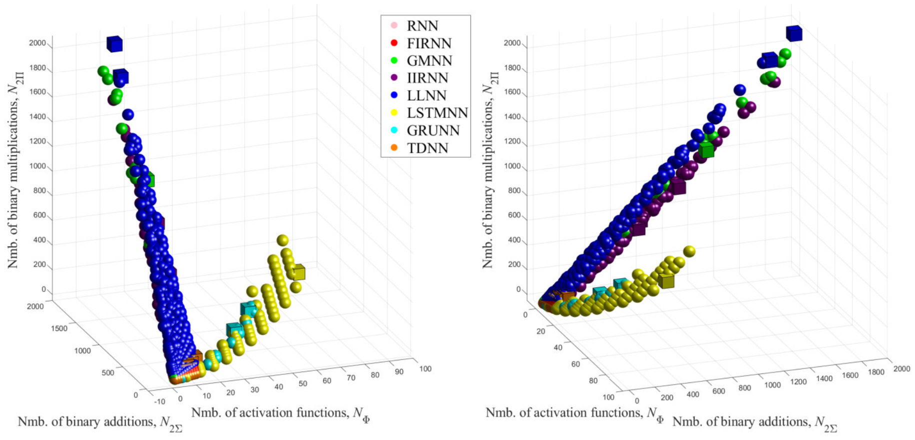
The graph on the lest side in Fig. 2(a) primarily focuses on the computational complexity between the number of act. functions and bin. multiplication operations across various NN models, while also showing bin. additions in a 3D perspective. It includes 19 TDNN, 24 FIRNN, 3 IIRNN, 2 GMNN, 9 LLNN, 22 RNN, 6 LSTMNN, and 2 GRUNN architectures. Cubes denote the NN architectures with the highest accuracy, while spheres represent those within 99.9% of the highest accuracy attained by the leading $IIRNN_1$ network. A dense cluster of FIRNNs, TDNNs and RNNs at the bottom of the graph indicates these networks require fewer act. functions. IIRNN, GMNN, and LLNN exhibit a moderate number of act. functions, with differing bin. multiplication needs. GRUNN shows a moderate increase in both bin. multiplications and act. functions, indicating a higher but manageable computational load. However, LSTMNNs are characterized by the highest number of act. functions.

The graph on the right side in Fig. 2(a) focuses on the computational complexity between the number of bin. multiplications and bin. additions, while also showing act. functions in a 3D perspective. LLNN shows a noticeable linear relationship, with an increase in bin. multiplications generally corresponding to an increase in bin. additions. FIRNN, TDNN, and RNN configurations cluster towards the lower end of the graph, while GRUNN and LSTMNN occupy the middle range, with GRUNN showing slightly lower computational needs than LSTMNN. In contrast, IIRNN, and LLNN are spread across the range, indicating variable computational requirements. GMNN stands out, reflecting the highest demands for both bin. multiplications and additions.

To explore a broader spectrum of our implemented NN architectures, we established a new threshold of 99% accuracy across all NN models. This threshold enables us to investigate deeper relationships between computational parameters and identify smaller, less resource-intensive architectures that maintain high accuracy.



(a) Neural network architectures achieving an accuracy rate of 99.9% of the highest accuracy attained by the leading IIRNN$_1$ network



(b) Neural network architectures achieving an accuracy rate of 99% or higher

Fig. 2. Computational complexity analysis of an expanded range of NN architectures. The perspectives offered include: on the left side – the relationship between bin. multiplications and act. functions, and on the right side – the relationship between bin. multiplications and bin. additions. Cubes (instead of spheres) denote the NN architectures with the highest accuracy.

The Fig. 2(b) expands the analysis to a wider range of NN models (all achieving an accuracy rate of 99% or higher): 71 TDNN, 248 FIRNN, 142 IIRNN, 55 GMNN, 360 LLNN, 92 RNN, 58 LSTMNN, and 15 GRUNN. This broader dataset reveals that the linear dependency between the number of bin. multiplications and bin. additions is not exclusive to LLNN; it is also evident in FIRNN, GMNN, IIRNN, LSTMNN, and GRUNN. Furthermore, a linear trend is observed for LSTMNN and GRUNN in terms of act. functions with respect to both bin. additions and bin. multiplications. This suggests a more generalizable relationship across different NN architectures, highlighting consistent computational patterns and dependencies that could inform optimization strategies for various NN designs.

Finally, to visualize not only the best accuracy achieving NNs but also the smallest architecture NNs, with the current threshold set at 99% accuracy we looked for the smallest architectures (sorting then according to the smallest number of bin. additions, bin. multiplications, act. functions, weights, and delays). Table 3 provides a comprehensive comparison of NN structures that achieve an accuracy of 99% or higher while maintaining minimal computational complexity within each NN model.

Table 3. Neural Network Architectures with Minimal Computational Complexity Achieving 99% or Higher Accuracy

Neural Network[*]	Total Number of Parameters[**]					Highest ACC, %
	$N_{2\Sigma}$	$N_{2\Pi}$	N_{Φ}	N_{W}	N_{D}	
TDNN	2	2	2	2	1	99.27
GMNN	3	2	2	5	1	99.35
FIRNN	3	3	2	2	1	99.26
RNN	1	3	2	3	1	99.18
LLNN	6	8	2	3	1	99.27
GRUNN	7	6	5	7	1	99.23
LSTMNN	9	8	7	9	1	99.35
IIRNN	44	34	3	2	42	99.29

[*] The gray background outlines the NNs with highest ACC or lowest parameter values.
[**] $N_{2\Sigma}$ – bin. additions; $N_{2\Pi}$ – bin. multiplications; N_{Φ} – act. func.; N_{W} – weights; N_{D} – delays.

The FIRNN model demonstrates exceptional efficiency, achieving an accuracy of 99.27% with the lowest parameter counts: 1 bin. addition, 2 bin. multiplications, 2 act. functions, 2 weights, and 1 delay. The RNN achieves a similar accuracy of 99.18% with slightly higher parameter requirements. TDNN also shows a high accuracy of 99.24% with modest computational complexity. The LLNN model, while achieving 99.27% accuracy, requires more parameters, particularly in bin. additions and multiplications. GRUNN achieves 99.23% accuracy but with a considerable increase in the number of parameters. LSTMNN, achieving the highest accuracy of 99.35%, also requires the highest number of parameters among the simpler networks. Finally, the IIRNN structure, with an accuracy of 99.29%, demands a significant number of parameters, indicating a higher computational complexity. This table highlights the trade-off between computational complexity and accuracy across different NN models.

Adaptability of Neural Networks to the Task. Table 4 presents NNs that achieved 99% or higher accuracy, architectural parameters, specifically the number of delays, inputs, and hidden neurons. *The main objective* of the analysis is to get insights on NN models intrinsic flexibility to adapt to the given HAR task.

For RNNs, it is noticeable that most of the most accurate architectures vary in terms of the number of hidden neurons, ranging from 2 to 10. However, the number of delays

Table 4. Summary of Neural Network Architectures that Achieved 99% or Higher Accuracy

Neural	Total Number of Parameters*		
Network	Inputs, N_I	Delays, N_D	Hidden Neurons, N_H
TDNN	1, 2, 5, 6, 7, 8	1	1 – 5 – 10
FIRNN	1, 2	1 – 5 – 10	1 – 5 – 10
GMNN	1 – 9	1 – 3 – 10	1 – 10
IIRNN	3, 4, 6, 7	2 – 9, 10	4, 5 – 10
LLNN	2 – 9 – 10	1 – 9 – 10	1 – 10
LSTMNN	1 – 2 – 6	1 – 10	1
RNN	1	1 – 4 – 10	2 – 6 – 10
GRUNN	1, 2	1 – 7 – 10	1

* The gray background outlines the architecture parameters of the NNs that achieved the highest accuracy.

varies across the entire tested range (from 1 to 10). The most accurate architecture for RNNs has 1 input, 4 delays, and 6 hidden neurons. In FIRNNs, the optimal choice is to use 1 or 2 hidden neurons, as adding more tends to decrease the network's performance for this task. The number of inputs and delays for FIRNNs does not significantly affect performance, as they are spread across all tested variations (from 1 to 10). The most accurate architecture for FIRNNs has 2 inputs, 5 delays, and 5 hidden neurons.

For GMNNs, it is suggested to use between 1 and 6 hidden neurons. A clear linear relationship between the number of inputs and delays is noticeable, suggesting that using the same number of delays and inputs for a single structure, varying from 1 to 10, is beneficial. The most accurate architecture for GMNNs has 9 inputs, 3 delays, and 10 hidden neurons. For IIRNNs, it is recommended to use 3, 4, 6, or 7 inputs with 4 or more hidden neurons and 2 or more delays. The most accurate architecture for IIRNNs has 6 inputs, 9 delays, and 5 hidden neurons.

LLNNs show that there should be at least 2 inputs, and the number of delays should be equal to or greater than the number of inputs. The number of hidden neurons for LLNNs does not matter much, as the most accurate architectures are equally spread across all tested variations (from 1 to 10). The most accurate architecture for LLNNs has 9 inputs, 9 delays, and 10 hidden neurons. For LSTMNNs, the number of delays is less important than the number of inputs. It is noticeable that when the number of inputs exceeds 6, the number of good architectures decreases significantly. The most accurate architecture for LSTMNNs has 2 inputs, 10 delays, and 1 hidden neuron.

For GRUNNs, using 1 input with 1 to 10 delays is suggested. The most accurate architecture for GRUNNs has 2 inputs, 7 delays, and 1 hidden neuron. For TDNNs, most of the most accurate architectures appear when there are 1, 2, 5, 6, 7, or 8 inputs.

Memory Utilization of Neural Networks. The memory utilization patterns of different NNs architectures are shown in Fig. 3. Each graph illustrates the relationship between memory usage and accuracy (ACC, %) for a specific NN architecture. Each dot in the graphs represents a different configuration of the respective architecture, varying

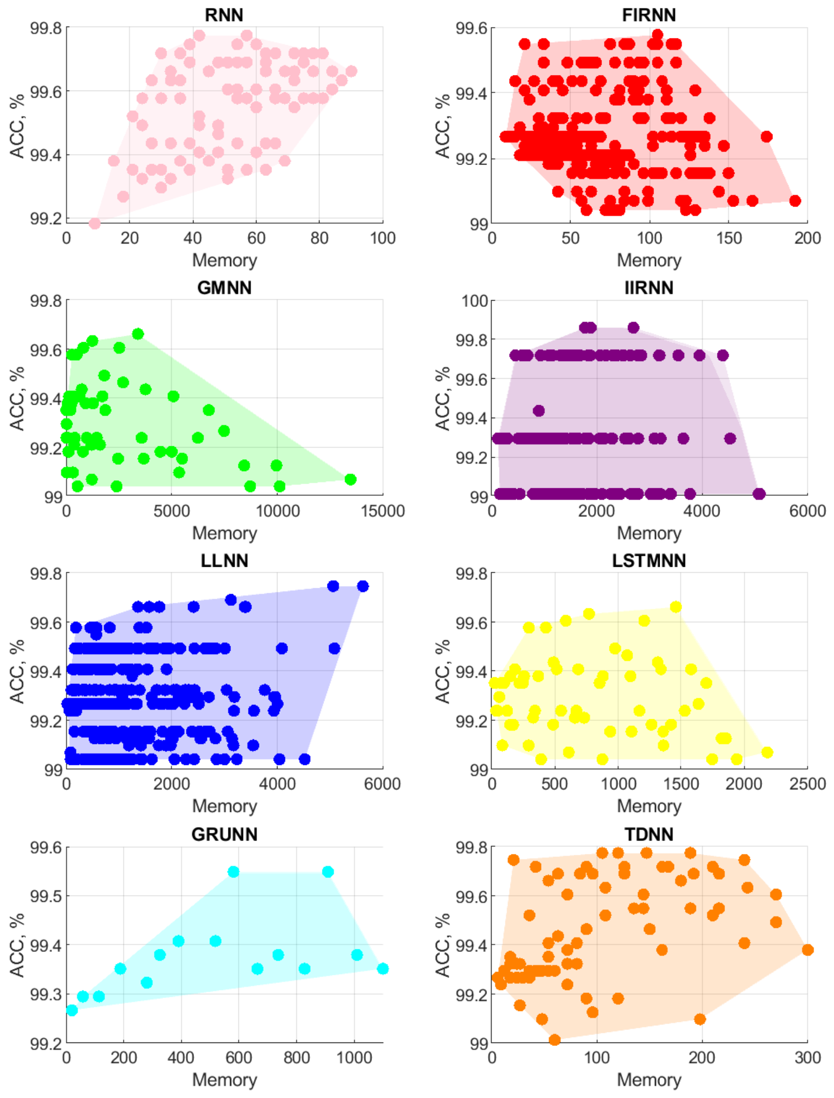


Fig. 3. Memory utilization of dynamic NN models, each achieving at least 99% accuracy. The graph shows the relationship between memory usage (sum of weights, delays, and coefficient-adjusted activation functions) and accuracy across different architectures.

in the number of inputs, delay elements, or hidden neurons. *The main objective* of the analysis is to identify memory allocation needs for each NNs architecture as also as to get insights on memory usage influence on the overall NNs accuracy.

Memory utilization in these NNs consists of several components. The total memory is the sum of the number of weights and the number of delays, with equal memory allocation needed for each weight and delay. Additionally, act. functions contribute to memory usage. They are evaluated as lookup tables, and their memory contribution is not directly equivalent to weights or delays. Therefore, the memory allocated for act. functions is multiplied by a specific coefficient to account for their different impact on overall memory usage.

In Fig. 3 RNN graph when memory allocation increases from approximately 0 to 80 units, the accuracy generally trends upwards from 99.2 to about 99.8%. This suggests that RNNs benefit from increased memory, enhancing their ability to learn and retain temporal dependencies, thereby improving performance.

The FIRNN graph shows that accuracy peaks around 100 units of memory usage and then declines. Accuracy ranges from 99.0% to 99.6%, indicating an optimal memory size for FIRNNs. Beyond this point, more memory doesn't lead to better accuracy and may even reduce performance. This suggests a critical balance in memory allocation for FIRNNs.

The GMNN graph shows a clear positive correlation between memory and accuracy up to a certain threshold. Memory usage ranges widely from 0 to 15,000 units, with accuracy improving from 99.0% to about 99.8%. This indicates that GMNNs use large memory capacities to manage detailed temporal data effectively, enhancing performance in tasks requiring comprehensive historical analysis.

The IIRNN graph shows a high but stable accuracy range from 99.0% to 99.86%, with memory usage from 0 to 6,000 units. Accuracy doesn't vary much with memory changes, suggesting IIRNNs maintain high performance across different memory levels. This stability indicates efficient memory usage in IIRNNs. A noticeable feature in the IIRNN graph is the distinct horizontal lines formed by data points at specific accuracy levels: 99.0%, 99.25%, and 99.75%. These lines suggest that certain configurations of IIRNNs consistently achieve these accuracy levels regardless of variations in memory usage. This pattern indicates that while memory allocation is crucial, there are other factors within the IIRNN architecture that strongly influence its accuracy, leading to these stable performance bands.

The LLNN graph shows a slight positive trend in accuracy with increasing memory usage, ranging from 0 to 6,000 units. Accuracy improves from 99.0% to about 99.8%, implying LLNNs can benefit from more memory, but less dramatically than other architectures. Moreover, there are noticeable horizontal lines at accuracy levels such as 99.1%, 99.3%, 99.4%, and 99.5%. These lines indicate that LLNNs also have configurations that consistently achieve these accuracy levels. The presence of these lines suggests that while memory usage impacts performance, certain LLNN configurations can maintain specific accuracy thresholds, highlighting a degree of robustness in their design.

The LSTMN graph shows memory usage from 0 to 2,500 units, with accuracy improving from 99.0% to about 99.8%. This positive correlation indicates that LSTMNs effectively use more memory to maintain long-term dependencies, which is crucial for tasks requiring extended temporal context.

The GRUNN graph shows a notable positive correlation between memory usage and accuracy, with memory ranging from 0 to 1,000 units. Accuracy increases from 99.2% to about 99.6%, showing that GRUNNs use more memory to improve performance.

This architecture's ability to manage and update memory states dynamically adds to its efficiency.

The TDNN graph shows a broader distribution of memory usage from 0 to 300 units, with accuracy ranging from 99.2% to 99.8%. There is a slight positive trend, indicating TDNNs benefit from more memory to some extent. However, the variability suggests other factors also affect their accuracy.

5 Conclusions

This study explored eight different models of small-scale dynamic neural networks to identify the least complex architectures capable of accurately classifying walking and running activities using accelerometer data. Based on the analysis of 440,000 distinct NN implementations, the following key points were observed:

1. Among the configurations tested, the IIRNN, especially IIRNN$_1$, achieved the highest accuracy of 99.86%. with the least computational demand in terms of bin. additions and bin. multiplications.
2. The TDNN demonstrated an excellent balance by providing a high accuracy of 99.27% while requiring lowest computational complexity in terms of bin. additions and bin. multiplications making it highly suitable for real-time applications.
3. Despite its higher computational demands, the GMNN exhibited strong performance, achieving 99.35% accuracy with the fewest number of act. functions (2).
4. Among the NN models analyzed, FIRNNs and TDNNs stand out for their ability to achieve high accuracy with minimal architectural complexity. FIRNNs perform optimally with 1 or 2 hidden neurons, 2 inputs, and 5 delays, while TDNNs excel with 1 or 2 inputs and 6 hidden neurons, making both networks highly suitable for tasks requiring efficient, real-time processing.
5. Notably, the IIRNN and LLNN graphs show distinct horizontal lines at certain accuracy levels, indicating that these networks achieve stable and consistent performance across a range of memory usages. This robustness makes them reliable choices for applications where maintaining high accuracy regardless of memory constraints is essential.

References

Akter, M., Ansary, S., Khan, M. A.-M., Kim, D. (2023). Human activity recognition using attention-mechanism-based deep learning feature combination, *Sensors* **23**(12), 15. https://doi.org/10.3390/s23125715

Al-Fraihat, D., Sharrab, Y., Alzyoud, F., Qahmash, A., Tarawneh, M., Maaita, A. (2024). Speech recognition utilizing deep learning: A systematic review of the latest developments, *Human-Centric Computing and Information Sciences* **14**, 19143 – 19165. https://doi.org/10.22967/HCIS.2024.14.015

Alessandrini, M., Biagetti, G., Crippa, P., Falaschetti, L., Turchetti, C. (2021). Recurrent neural network for human activity recognition in embedded systems using PPG and accelerometer data, *Electronics* **10**(14), 1–18. https://doi.org/10.3390/electronics10141715

Alhudhaif, A. (2024). A novel approach to recognition of alzheimer's and parkinson's diseases: Random subspace ensemble classifier based on deep hybrid features with a super-resolution image, *PeerJ Computer Science* **10**, e1862. https://doi.org/10.7717/peerj-cs.1862

Back, A. D., Tsoi, A. C. (1992). An adaptive lattice architecture for dynamic multilayer perceptrons, *Neural Computation* **4**(6), 922–931. https://doi.org/10.1162/neco.1992.4.6.922

Butkevičiūtė, E., Bikulčienė, L., Blazauskas, T., Žemaitytė, A. (2023). Cognitive checkup and mental training platform for elite athletes, *Baltic Journal of Modern Computing* **11**(2), 257–271. https://doi.org/10.22364/bjmc.2023.11.2.03

Campolucci, P., Uncini, A., Piazza, F., Rao, B. D. (1999). On-line learning algorithms for locally recurrent neural networks, *IEEE Transactions on Neural Networks* **10**(2), 253–271. https://doi.org/10.1109/72.750549

Cho, K., Merrienboer, B., Bahdanau, D., Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches, *Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)* p. 9. https://doi.org/10.3115/v1/W14-4012

Davidashvilly, S., Cardei, M., Hssayeni, M., Chi, C., Ghoraani, B. (2024). Deep neural networks for wearable sensor-based activity recognition in parkinson's disease: Investigating generalizability and model complexity, *BioMedical Engineering OnLine* **23**(1), 17. https://doi.org/10.1186/s12938-024-01214-2

de Vries, B., Principe, J. C. (1992). The gamma model–a new neural model for temporal processing, *Neural Networks* **5**(4), 565–576. https://doi.org/10.1016/S0893-6080(05)80035-8

Dentamaro, V., Gattulli, V., Impedovo, D., Manca, F. (2024). Human activity recognition with smartphone-integrated sensors: A survey, *Expert Systems with Applications* **246**, 1–22. https://doi.org/10.1016/j.eswa.2024.123143

Glorot, X., Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks, *in* Teh, Y. W., Titterington, M. (eds), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Vol. 9 of *Proceedings of Machine Learning Research*, PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 249–256.

Gomaa, W., Khamis, M. A. (2023). A perspective on human activity recognition from inertial motion data, *Neural Computing & Applications* p. 20463–20568. https://doi.org/10.1007/s00521-023-08863-9

Hamzacebi, C., Avni, H. E., Cakmak, R. (2019). Forecasting of Turkey's monthly electricity demand by seasonal artificial neural network, *Neural Computing & Applications* **31**(7), 2217–2231. https://doi.org/10.1007/s00521-017-3183-5

Hari, T., Watanabe, S., Zhang, Y., Chan, W. (2017). Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM, *18th Annual Conference of the International Speech Communication Association (Interspeech 2017), vols 1–6: Situated Interaction*, Interspeech, pp. 949–953. https://doi.org/10.21437/Interspeech.2017-1296

Helmi, A. M., Al-qaness, M. A. A., Dahou, A., Abd Elaziz, M. (2023). Human activity recognition using marine predators algorithm with deep learning, *Future Generation Computer Systems-the International Journal of Escience* **142**, 340–350. https://doi.org/10.1016/j.future.2023.01.006

Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory, *Neural Computation* **9**(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jiang, P., Li, R., Liu, N., Gao, Y. (2020). A novel composite electricity demand forecasting framework by data processing and optimized support vector machine, *Applied Energy* **260**, 1–15. https://doi.org/10.1016/j.apenergy.2019.114243

Kasparaitis, P., Antanavičius, D. (2023). Investigation of input alphabets of end-to-end lithuanian text-to-speech synthesizer, *Baltic Journal of Modern Computing* **11**(2), 285–296. https://doi.org/10.22364/bjmc.2023.11.2.05

Khan, I. U., Afzal, S., Lee, J. W. (2022). Human activity recognition via hybrid deep learning based model, *Sensors* **22**(1), 1–16. https://doi.org/10.3390/s22010323

Kosar, E., Barshan, B. (2023). A new CNN-LSTM architecture for activity recognition employing wearable motion sensor data: Enabling diverse feature extraction, *Engineering Applications of Artificial Intelligence* **124**, 1–15. https://doi.org/10.1016/j.engappai.2023.106529

Kumar, P., Chauhan, S., Awasthi, L. K. (2024). Human activity recognition (HAR) using deep learning: Review, methodologies, progress and future research directions, *Archives of Computational Methods in Engineering* **31**(1), 179–219. https://doi.org/10.1007/s11831-023-09986-x

Laktionov, I., Diachenko, G., Koval, V., Yevstratiev, M. (2023). Computer-oriented model for network aggregation of measurement data in iot monitoring of soil and climatic parameters of agricultural crop production enterprises, *Baltic Journal of Modern Computing* **11**(3), 500–522. https://doi.org/10.22364/bjmc.2023.11.3.09

Lawrence, S., Tsoi, A., Back, A. (1995). The gamma MLP for speech phoneme recognition, *in* Touretzky, D., Mozer, M., Hasselmo, M. (eds), *Advances in Neural Information Processing Systems*, Vol. 8, MIT Press, pp. 785–791.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, *in* Wallach, H., Larochelle, H., Beygelzimer, A., d'Alche Buc, F., Fox, E., Garnett, R. (eds), *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, Vol. 32 of *Advances in Neural Information Processing Systems*, pp. 1–14. 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada, DEC 08-14, 2019.

Li, Z., Liao, Y., Hu, B., Ni, L., Lu, Y. (2022). A financial deep learning framework: Predicting the values of financial time series with ARIMA and LSTM, *International Journal of Web Services Research* **19**(1), 1–15. https://doi.org/10.4018/IJWSR.302640

McClelland, J. L., Rumelhart, D. E. (1987). *Schemata and Sequential Thought Processes in PDP Models*, MIT press. https://doi.org/10.7551/mitpress/5236.003.0004

Navakauskas, D., Serackis, A., Matuzevičius, D., Laptik, R. (2014). *Specializuotos elektroninės intelektualiosios sistemos garsams ir vaizdams apdoroti. Teorija ir taikymai*, Vilnius Gediminas Technical University. https://doi.org/10.3846/2310-m

Navakauskienė, R., Navakauskas, D., Borutinskaitė, V., Matuzevičius, D. (2021). *Epigenetics and Proteomics of Leukemia: A Synergy of Experimental Biology and Computational Informatics*, Springer International Publishing. https://doi.org/10.1007/978-3-030-68708-3

Paliwal, K. (1991). A time-derivative neural net architecture - an alternative to the time-delay neural net architecture, *Neural Networks for Signal Processing Proceedings of the 1991 IEEE Workshop*, pp. 367–375. https://doi.org/10.1109/NNSP.1991.239505

Pham, T., Tran, T., Phung, D., Venkatesh, S. (2016). DeepCare: A deep dynamic memory model for predictive medicine, *Advances in Knowledge Discovery and Data Mining, Pakdd 2016, PT II*, Vol. 9652 of *Lecture Notes in Artificial Intelligence*, pp. 30–41. https://doi.org/10.1007/978-3-319-31750-2_3

Prechelt, L. (1998). Automatic early stopping using cross validation: Quantifying the criteria, *Neural Networks* **11**(4), 761–767. https://doi.org/10.1016/S0893-6080(98)00010-0

Serpush, F., Menhaj, M. B., Masoumi, B., Karasfi, B. (2022). Wearable sensor-based human activity recognition in the smart healthcare system, *Computational Intelligence and Neuroscience* **2022**, 1–23. https://doi.org/10.1155/2022/1391906

Sikder, N., Nahid, A.-A. (2021). KU-HAR: An open dataset for heterogeneous human activity recognition, *Pattern Recognition Letters* **146**, 46–54. https://doi.org/10.1016/j.patrec.2021.02.024

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). Attention is all you need, *in* Guyon, I., Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds), *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Vol. 30 of *Advances in Neural Information Processing Systems*, pp. 1–11. 31st Annual Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, DEC 04-09, 2017.

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., Koumoutsakos, P. (2020). Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics, *Neural Networks* **126**, 191–217. https://doi.org/10.1016/j.neunet.2020.02.016

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. (1989). Phoneme recognition using time-delay neural networks, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37**(3), 328–339. https://doi.org/10.1109/29.21701

Wan, E. A. (1990). Temporal backpropagation for FIR neural networks, *IJCNN International Joint Conference on Neural Networks, Vols 1–3*, Int Neural Network Soc, pp. A575–A580. International Joint Conf on Neural Networks (IJCNM-90), San Diego, CA, JUN 17-20, 1990.

Weerakody, P. B., Wong, K. W., Wang, G., Ela, W. (2021). A review of irregular time series data handling with gated recurrent neural networks, *Neurocomputing* **441**, 161–178. https://doi.org/10.1016/j.neucom.2021.02.046

Werbos, P. J. (1990). Backpropagation through time – what it does and how to do it, *Proceedings of the IEEE* **78**(10), 1550–1560. https://doi.org/10.1109/5.58337

Zhang, Z., Zhang, S., Chen, C., Yuan, J. (2024). A systematic survey of air quality prediction based on deep learning, *Alexandria Engineering Journal* **93**, 128–141. https://doi.org/10.1016/j.aej.2024.03.031

Received July 7, 2024 , accepted September 20, 2024