Human Activity Recognition Using Smartphones: A Comprehensive Study of Recurrent Neural Network Architectures

Bhavya Pandey, Akshat Jain, Parth Deshmukh

Abstract

Human Activity Recognition (HAR) has emerged as a critical research area with applications spanning healthcare monitoring, fitness tracking, and smart home systems. This study presents a comprehensive evaluation of recurrent neural network architectures for smartphone-based human activity recognition using the HAR dataset. We compare nine different models across three RNN families: Simple RNN, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), each implemented with multiple architectural complexities and optimization techniques. The dataset comprises accelerometer and gyroscope sensor data from 30 subjects performing various activities. Our experimental results demonstrate that GRU-based architectures achieve superior performance, with the GRU-3 model attaining the highest accuracy of 94% and F1-score of 94.1%. This research contributes to the development of efficient HAR systems by providing insights into the effectiveness of different recurrent architectures and optimization strategies for temporal sensor data classification.

Introduction

Human Activity Recognition using smartphones has gained significant importance due to the ubiquitous nature of mobile devices equipped with various sensors. The ability to classify human activities from sensor data has potential to transform multiple domains, including healthcare monitoring for elderly patients, fitness tracking applications, and smart home systems.

Traditional HAR approaches relied heavily on handcrafted features and classical machine learning algorithms, which often struggled to capture the complex temporal dependencies inherent in sensor data [2]. Recent advances in deep learning, particularly recurrent neural networks, have shown remarkable success in modeling sequential data, making them naturally suited for HAR tasks.

The motivation for this study comes from the growing need for efficient, lightweight HAR systems that can operate effectively on mobile devices while maintaining high accuracy. Unlike complex ensemble methods or computationally intensive models that achieve accuracies approaching 99%, this research focuses on finding the optimal balance between model complexity and performance using simpler, more deployable architectures.

The dataset used is the HAR dataset obtained from the University of California Irvine Machine Learning Repository. It contains sensor recordings from 30 healthy volunteers of varying ages performing six different activities while wearing a smartphone on their waist. The data consists of sensor data from the accelerometer and gyroscope in the phone, and is labelled for one of 6 activities- Walking, Walking upstairs, Walking downstairs, Sitting, Standing and Laying.

Background

Human Activity Recognition (HAR) has seen extensive research due to its applications in health monitoring, fitness tracking, and smart environments. Traditional approaches relied on handcrafted features derived from accelerometer and gyroscope data, followed by classification using models like SVMs or Random Forests [14]. While effective in some cases, these methods required domain expertise and lacked the ability to model complex temporal dynamics.

The shift toward deep learning has significantly advanced HAR. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), especially LSTM models, have shown superior performance by learning features directly from raw sensor data [15][16]. Hybrid architectures combining CNNs with LSTMs have also been explored to capture both spatial and temporal dependencies [17]. However, the growing complexity of these models has raised concerns about computational costs, especially for real-time or mobile applications.

Recent research has addressed this by suggesting lightweight and efficient architectures suitable for edge deployment [18], as well as optimization strategies like pruning and quantization. Our work builds upon these foundations by evaluating deep learning models on the HAR dataset, balancing predictive performance with computational efficiency.

Approach

The approach used in this study has been to analyse different recurrent neural network models and compare their performance and efficiency in classifying the activity using only the raw sensor data from the Human Activity Recognition (HAR) dataset. Identical baseline, enhanced and advanced versions of Simple RNN, GRU and LSTM models have been used, and compared on their ability to identify the activity being performed using accuracy and F1-score.

1. Data Acquisition & Representation

We utilized the HAR Dataset, which contains sensor signals from smartphones carried by individuals performing six different activities: Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, and Laying. The raw signals from accelerometers and gyroscopes along X,Y and Z axes (body_acc, body_gyro, and total_acc) were extracted and stacked to form input tensors. The final input shape was (n_samples, 128, 9), where 128 is the number of time steps and 9 represents the different signal channels.
Activity labels were read and converted into integer values ranging from 0 to 5. These were then one-hot encoded to facilitate training with categorical cross-entropy loss.

Signal types

Nine channels per timestamp:

INPUT_SIGNAL_TYPES = $\{\,\mathrm{body\_acc}_{x,y,z},\ \mathrm{body\_gyro}_{x,y,z},\ \mathrm{total\_acc}_{x,y,z}\}.$

Load into arrays

$$X_{\text{train}} \in \mathbb{R}^{N\times T\times F}, \quad X_{\text{test}} \in \mathbb{R}^{M\times T\times F}$$

Where,

T=128 timesteps per sample (2.56 s at 50 Hz),

F=9 channels,

N=7352,

M=2947 examples in train/test.

Labels

y∈{1,…,6} are one-hot encoded to $(\mathbf{y}_{\text{cat}})_i = \begin{cases} 1, & i = y, \\ 0, & \text{else.} \end{cases}$

2. Dataset & Hyperparameters

In our study, we explored multiple deep learning architectures, each with varying biases toward sequential sensor data. TensorFlow's tf.data.Dataset API was used to build efficient training and testing datasets. The datasets were cached, shuffled, batched, and prefetched to optimize performance during training. Cosine Annealing Learning Rate schedule was implemented to gradually reduce the learning rate during training, which helps escape local minima during training.

Dataset pipeline- uses batching and prefetching for throughput: (X,y)→Dataset→batch(128).

Common hyperparameters:

- Sequence length: 128
- Feature dim: 9
- Classes: 6
- Batch size =128
- Epochs =50 (With early stopping)
- Initial learning rate =2.5×10^−3
- L2-regularization weight =1.5×10−3
- Dropout rate p=0.2

3. Loss, Regularization & Optimization

The models were trained using the categorical cross-entropy loss function. The Adam optimizer was chosen for its adaptive learning capabilities. Regularization techniques such as dropout and L2 loss were employed to prevent overfitting.

- **Data loss**: Categorical cross-entropy (from logits):

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \mathbf{z}) = -\sum_{c=1}^{6} y_c \, \log\Big(\text{softmax}(z)_c\Big)$$

- **L2 penalty** on all trainable weights θ:

$$\mathcal{L}_{\ell_2} = \lambda \, \|\theta\|_2^2$$

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\ell_2}.$$

- **Total loss**:

- **Optimizer**: Adam

$$
\begin{aligned}
m_t &= \beta_1\, m_{t-1} + (1-\beta_1)\, \nabla_\theta \mathcal{L}, \\
v_t &= \beta_2\, v_{t-1} + (1-\beta_2)\, (\nabla_\theta \mathcal{L})^2, \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \\
\theta &\leftarrow \theta - \eta\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}.
\end{aligned}
$$

- **Learning-rate schedule**: Cosine Annealing

$$\eta(t) = \eta_{\min} + \tfrac{1}{2}(\eta_0 - \eta_{\min})\Big[1 + \cos\!\big(\tfrac{\pi\, t}{E}\big)\Big], \quad \eta_{\min} = 10^{-6}.$$

- **Early stopping** (patience = 3) restores best weights, and occasionally a Reduce-LR-on-Plateau (factor = 0.5) is applied.

4. Model Architectures

All models take input X∈R^(T×F) and produce logits z∈R^6. After the final dense layer, a softmax yields class probabilities.

Simple RNN Variants

RNN-1

Two stacked Simple RNN layers:

$$\mathbf{h}_t^{(1)} = \tanh(W^{(1)} x_t + U^{(1)} h_{t-1}^{(1)} + b^{(1)})$$

return seq → Dropout →

$$\mathbf{h}_t^{(2)} = \tanh(W^{(2)} h_t^{(1)} + U^{(2)} h_{t-1}^{(2)} + b^{(2)})$$

final state → Dropout → Dense(6).

RNN-2

Same as RNN-1, but each RNN output is **LayerNormalized** before dropout:

$$\mathrm{LN}(\mathbf{h}) = \gamma \frac{\mathbf{h} - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta.$$

RNN-3

Input noise: add N(0,0.12) to each feature.
Three Simple RNN layers (128→64→32 units), each with **recurrent_dropout** (0.2), **LayerNorm**, and Dropout.

4.2. LSTM Variants

Replace every Simple RNN cell above with an LSTM cell. For an LSTM layer:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i),\\
\mathbf{f}_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f),\\
\mathbf{o}_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o),\\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c),\\
c_t &= \mathbf{f}_t \odot c_{t-1} + \mathbf{i}_t \odot \tilde{c}_t,\\
h_t &= \mathbf{o}_t \odot \tanh(c_t).
\end{aligned}
$$

LSTM-1: two layers (128→64), Dropout.

LSTM-2: add LayerNorm around each layer's output.

LSTM-3: three layers (128→64→32) with GaussianNoise and recurrent_dropout (0.2), plus LayerNorm + Dropout.

4.3. GRU Variants

Substitute GRU cells, defined by:

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z),\\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r),\\
\tilde{h}_t &= \tanh(W x_t + U(r_t \odot h_{t-1}) + b),\\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.
\end{aligned}
$$

GRU-1: two layers (128→64), Dropout.

GRU-2: same with preceding LayerNorm calls.

GRU-3: 3 layers with 128, 64 and 32 units, recurrent dropout and Gaussian Noise

5. Evaluation

Model performance was evaluated using following metrics: Accuracy, Precision, Recall, and F1-Score. These metrics provide a comprehensive view of performance, especially in the multi-class classification setting.

After training each model, predictions $\hat{y}_i = \arg\max_c \ \mathrm{softmax}(z_i)_c$ are compared to true labels.

Accuracy: $\frac{1}{N} \sum_i \mathbf{1}(\hat{y}_i = y_i)$.

Macro-Precision: $\frac{1}{C} \sum_c \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}$.

Macro-Recall: $\frac{1}{C}\sum_c \frac{\mathrm{TP}_c}{\mathrm{TP}_c+\mathrm{FN}_c}$.

`Macro-F1: $\frac{1}{C}\sum_c \frac{2\,P_c\,R_c}{P_c+R_c}$.

Results are collated in a table for easy comparison of all eight architectures.

Results

Dataset Description

The dataset used for this study is the HAR dataset obtained from the University of California Irvine Machine Learning Repository. It contains sensor recordings from 30 healthy volunteers aged 19-48 years performing six activities while wearing a Samsung Galaxy S II smartphone on their waist. The dataset includes:

- **Activities**: Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, Laying.
- **Sensors**: 3-axial accelerometer and gyroscope data.
- **Sampling**: 50Hz with 2.56-second sliding windows (128 readings per window).
- **Features**: 9 signal types (body acceleration, total acceleration, angular velocity for x, y, z axes).
- **Dataset Split**: 7,352 training samples, 2,947 test samples.

Experimental Configuration

All models were implemented using TensorFlow with consistent hyperparameters:

- **Batch Size**: 128
- **Epochs**: 50 (with early stopping)
- **Learning Rate**: 0.0025
- **Regularization**: L2 regularization (λ = 0.0015)
- **Dropout Rate**: 0.2
- **Optimizer**: Adam with optional gradient clipping and cosine annealing.

Model Architectures

RNN Models

RNN-1 (Baseline): A two-layer SimpleRNN architecture with 128 and 64 units respectively, incorporating dropout layers and L2 regularization. This model serves as the baseline for comparison.

RNN-2 (Enhanced): Building upon RNN-1, this model adds layer normalization after each RNN layer and implements cosine annealing learning rate scheduling instead of plateau-based reduction.

RNN-3 (Advanced): A more complex RNN variant featuring three layers (128, 64, 32 units), Gaussian noise injection for data augmentation, recurrent dropout, and gradient clipping to prevent exploding gradients.

RNN Models
RNN Model 1: 2 SimpleRNN layers (128 → 64), each followed by Dropout, final Dense layer for classification.
RNN Model 2: Same as Model 1, but with Layer Normalization after each SimpleRNN before Dropout.
RNN Model 3: 3 SimpleRNN layers (128 → 64 → 32), each followed by Layer Norm + Dropout, with Gaussian Noise before first layer.

GRU Models

GRU-1 (Baseline): A two-layer GRU architecture mirroring the RNN-1 structure but utilizing GRU cells for better gradient flow and memory retention.

GRU-2 (Enhanced): Incorporates layer normalization and cosine annealing learning rate scheduling while maintaining the two-layer GRU structure.

GRU-3 (Advanced): GRU model with three layers, Gaussian noise, recurrent dropout, and gradient clipping.

GRU Models
GRU Model 1: 2 GRU layers (128 → 64), each followed by Dropout, final Dense layer for classification.
GRU Model 2: Same as Model 1, but with Layer Normalization after each GRU layer before Dropout.
GRU Model 3: 3 GRU layers (128 → 64 → 32), each followed by Layer Norm + Dropout, with Gaussian Noise before first layer.

LSTM Models

LSTM-1 (Baseline): Two-layer LSTM architecture with 128 and 64 units, L2 regularization and dropout layers.

LSTM-2 (Enhanced): Adds layer normalization and cosine annealing learning rate scheduling to the baseline LSTM.

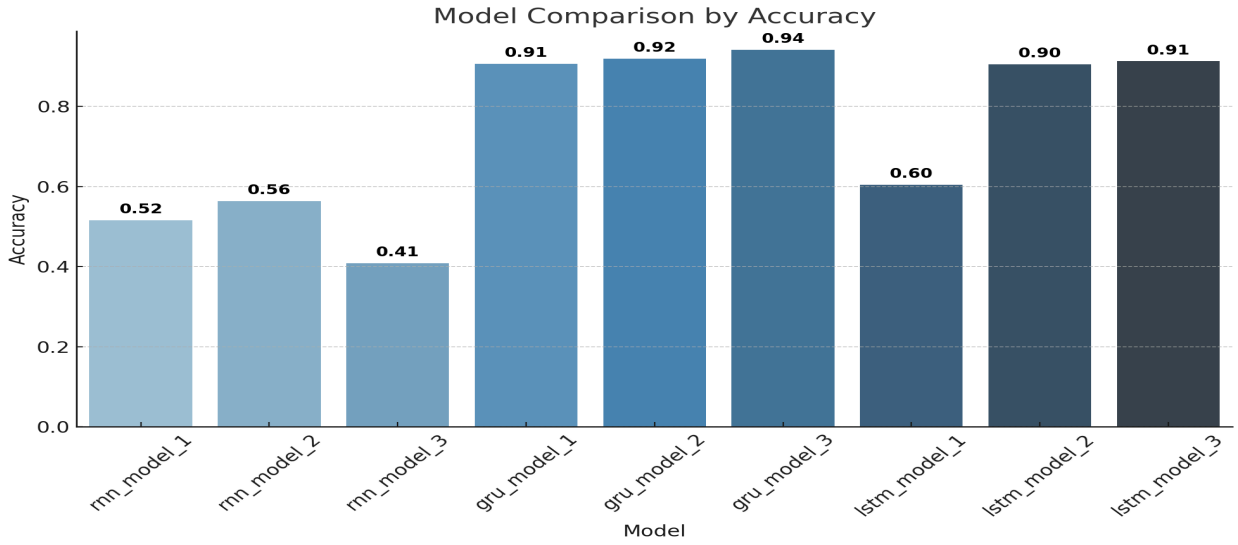
LSTM-3 (Advanced): The most complex LSTM variant incorporating three layers, noise injection, recurrent dropout, and gradient clipping techniques.

LSTM Models
LSTM Model 1: 2 LSTM layers (128 → 64), each followed by Dropout, final Dense layer for classification.
LSTM Model 2: Same as Model 1, but with Layer Normalization after each LSTM layer before Dropout.
LSTM Model 3: 3 LSTM layers (128 → 64 → 32), each followed by Layer Norm + Dropout, with Gaussian Noise before first layer.

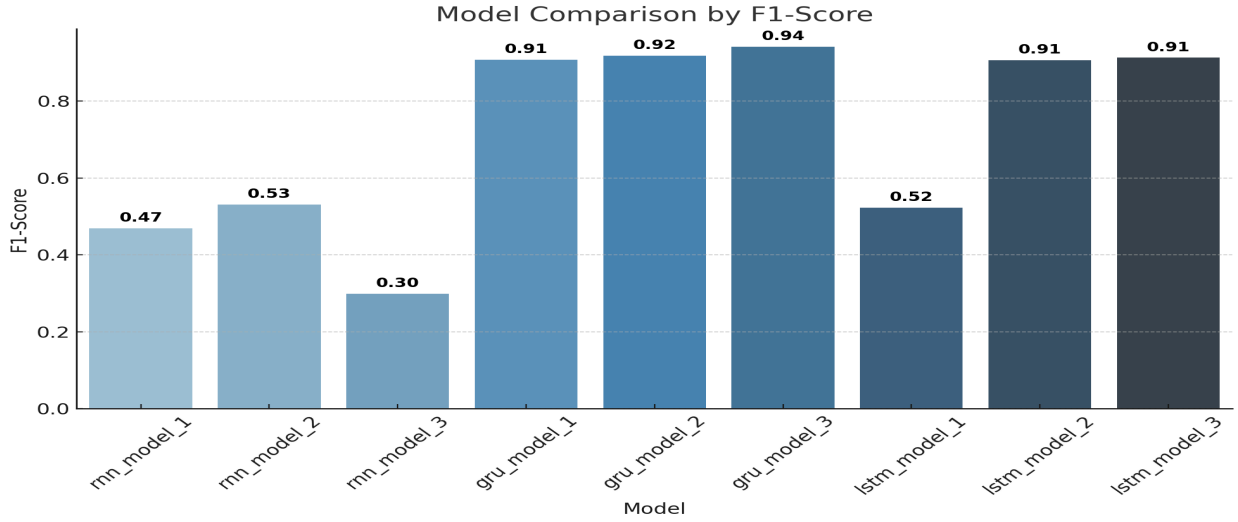
Performance Metrics-

Model	Accuracy	Precision	Recall	F1-Score
rnn_model_1	0.515779	0.520531	0.508037	0.469105
rnn_model_2	0.563963	0.552046	0.55221	0.530966
rnn_model_3	0.40889	0.262555	0.38666	0.299079
gru_model_1	0.906345	0.907459	0.908484	0.907321
gru_model_2	0.918901	0.920138	0.919671	0.918072
gru_model_3	0.940957	0.94121	0.941845	0.941112
lstm_model_1	0.604683	0.568444	0.57887	0.522843
lstm_model_2	0.904988	0.905919	0.907873	0.906381
lstm_model_3	0.912453	0.914721	0.913081	0.913058

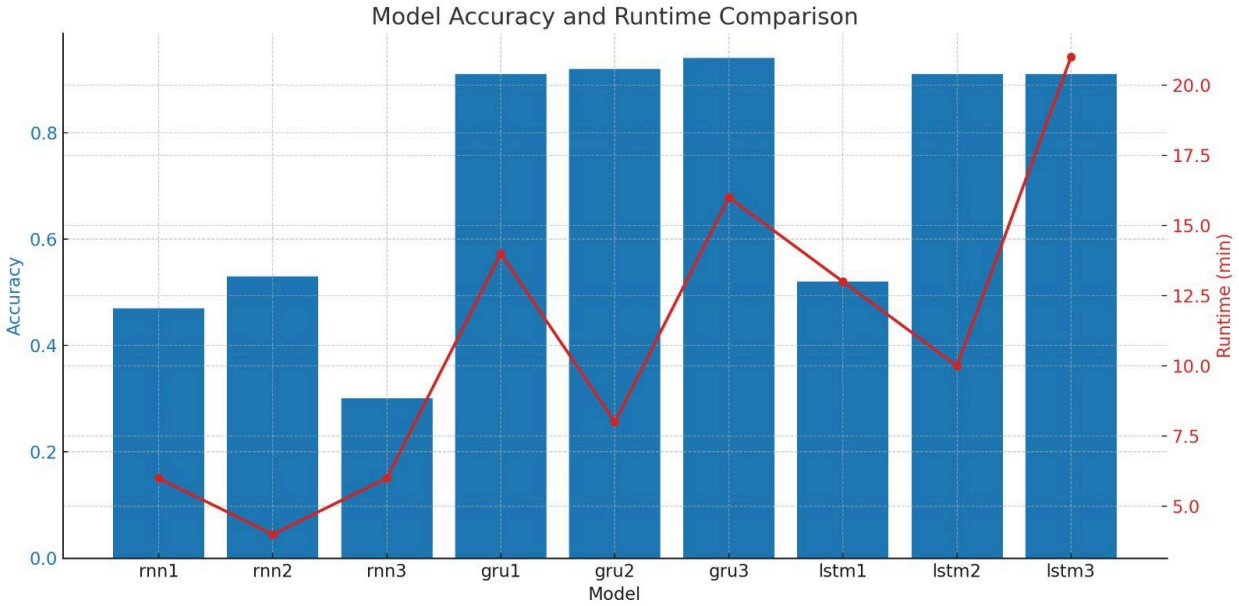
Accuracy-



F1-Score-



Runtime vs Accuracy-



Discussion

Model Performance Overview:

The comparative evaluation of nine deep learning models, comprising RNN, LSTM, and GRU variants revealed clear performance differences. GRU-based models consistently outperformed both LSTM and standard RNN architectures across all evaluation metrics. This trend aligns with existing research, which highlights the effectiveness of gated architectures like GRUs and LSTMs in processing sequential data. In contrast, plain RNNs, due to the vanishing gradient problem, struggle to retain long-term dependencies, making them less effective for tasks like Human Activity Recognition (HAR), where temporal context is critical [3].

GRU Superiority

GRU models performed better overall, with accuracy scores ranging from 90.6% to 94%. This strong performance shows the GRU model's ability to balance efficiency with effective temporal modeling. Unlike LSTMs, GRU employs a simpler gating mechanism using only reset and update gates which allows for faster training while maintaining performance. Recent research confirms that GRU models are often more suitable for HAR tasks due to their ability to efficiently model long-range dependencies without the overhead of more complex architectures.

Effect of Architectural Enhancements

Model performance also benefitted from architectural enhancements such as layer normalization, cosine annealing learning rate schedules, and regularization techniques. These strategies contributed to performance gains. For instance, GRU-2 and GRU-3 exhibited incremental improvements over GRU-1, indicating that such optimizations can fine-tune already strong model architectures. These enhancements played a crucial role in mitigating overfitting and improving generalization on the test set.

Limitations of RNN and LSTM Models

Standard RNN models were the weakest performers, with accuracies between 40.8% and 56.3%, emphasizing their limited capacity to model complex temporal dynamics without gating mechanisms. LSTM models, while significantly better than RNNs, did not outperform their GRU. Their accuracies ranged from 60.4% to 91.2%. This underperformance suggests that the more complex memory cell structure of LSTMs may not offer a practical advantage over GRUs in HAR applications. Overall, GRUs appear to provide the optimal trade-off between simplicity and performance for this task.

Comparison with State-of-the-Art

Our results compare favorably with recent HAR studies. While some recent works report accuracies exceeding 98%[7], these often employ more complex architectures, ensemble methods, or hybrid approaches combining CNNs with RNNs. Our study demonstrates that simple GRU architectures can achieve competitive performance (94%) with significantly less parameters and computational requirements.

Recent benchmarking studies have shown that CNN-based models [12] often achieve superior performance on HAR tasks, with some reporting accuracies over 99%. However, our focus on pure RNN architectures shows the applications where temporal modeling is prioritized and computational resources are constrained.

Practical Implications

The superior performance of GRU models has important implications for practical HAR deployment. GRUs offer several advantages:

1. **Computational Efficiency**: Less memory requirements compared to LSTMs.
2. **Training Stability**: Better gradient flow than simple RNNs.
3. **Parameter Efficiency**: Fewer parameters than equivalent LSTM architectures.
4. **Real-time Capability**: Suitable for mobile device deployment.

Optimization Strategies

The study reveals that architectural enhancements have varying impacts:

- **Layer Normalization**: Consistently improves stability and performance.
- **Cosine Annealing**: Provides better learning rate scheduling than plateau reduction.
- **Gaussian Noise**: Acts as effective regularization for temporal data.
- **Gradient Clipping**: Essential for deeper RNN architectures.

Limitations and Future Work

Several limitations should be acknowledged:

1. **Dataset Scope**: Evaluation limited to UCI HAR dataset; broader validation needed.
2. **Activity Complexity**: Focus on basic activities; complex activities remain challenging.
3. **Subject Variability**: Limited analysis of cross-subject generalization.
4. **Real-time Performance**: No evaluation of actual deployment constraints.

Future research directions include:

- **Hybrid Architectures**: Combining RNNs with CNNs for spatial-temporal feature extraction.
- **Attention Mechanisms**: implemented attention for improved temporal modelling.
- **Transfer Learning**: Use pre-trained models for new domains.
- **Edge Deployment**: Optimize models for mobile and IoT devices.

Conclusion

Our study shows that GRU based models provide the best performance for smartphone-based human activity recognition. The GRU-3 model, with advanced regularization and optimization techniques, resulted in the highest accuracy of 94% while maintaining computational efficiency suitable for mobile deployment.

The research contributes to the HAR field by providing systematic comparison of RNN architectures and demonstrating that sophisticated gating mechanisms are crucial for effective temporal modeling in sensor data. These results support the development of efficient HAR systems that balance accuracy with computational constraints, making them suitable for real-world applications in healthcare, fitness, and smart environment domains.

Well-designed GRU models provide an excellent compromise between performance and efficiency. This makes them particularly suitable for applications where computational resources are limited and real-time processing is required. The approach and findings presented in this project provide a foundation for future HAR research and practical system development, contributing to the broader goal of creating intelligent, responsive environments that can understand and adapt to human activities.

References

1. Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A Public Domain Dataset for Human Activity Recognition Using Smartphones. IEEE.5

2. Kulsoom, Farzana, et al. "A review of machine learning-based human activity recognition for diverse applications." Neural Computing and Applications 34.21 (2022): 18289-18324.

3. Nazari, Farhad, et al. "Comparison of deep learning techniques on human activity recognition using ankle inertial signals." 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2022.

4. Khan, Murad, and Yousef Hossni. "A comparative analysis of LSTM models aided with attention and squeeze and excitation blocks for activity recognition." Scientific Reports 15.1 (2025): 3858.

5. Mekruksavanich, Sakorn, and Anuchit Jitpattanakul. "Lstm networks using smartphone data for sensor-based human activity recognition in smart homes." Sensors 21.5 (2021): 1636.

6. Hossain, Md Meem, et al. "Benchmarking Classical, Deep, and Generative Models for Human Activity Recognition." arXiv preprint arXiv:2501.08471 (2025).

7. Zhou, Haotian, et al. "Efficient human activity recognition on edge devices using DeepConv LSTM architectures." Scientific Reports 15.1 (2025): 13830.

8. Chen, Kaixuan, et al. "Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities." ACM Computing Surveys (CSUR) 54.4 (2021): 1-40.

9. Gu, Fuqiang, et al. "A survey on deep learning for human activity recognition." ACM Computing Surveys (CSUR) 54.8 (2021): 1-34.

10. Vrigkas, Michalis, Christophoros Nikou, and Ioannis A. Kakadiaris. "A review of human activity recognition methods." Frontiers in Robotics and AI 2 (2015): 28.

11. Ramasamy Ramamurthy, Sreenivasan, and Nirmalya Roy. "Recent trends in machine learning for human activity recognition A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1254.

12. Khatun, Mst Alema, et al. "Deep CNN-LSTM with self-attention model for human activity recognition using wearable sensor." IEEE Journal of Translational Engineering in Health and Medicine 10 (2022): 1-16.

13. Hossain, Md Meem, et al. "Benchmarking Classical, Deep, and Generative Models for Human Activity Recognition." arXiv preprint arXiv:2501.08471 (2025).

14. Anguita, D., et al. (2013). A Public Domain Dataset for Human Activity Recognition Using Smartphones. ESANN.

15. Ordóñez, F. J., & Roggen, D. (2016). Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. Sensors.

16. Hammerla, N. Y., et al. (2016). Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables. IJCAI.

17. Jiang, W., & Yin, Z. (2015). Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks. ACM ICMI.

18. Ha, S., et al. (2015). Multi-modal Convolutional Neural Networks for Activity Recognition. IEEE Sensors Applications Symposium.