Graph Neural Networks for Human Activity Recognition in Smart Environments

David-Dumitru Ţigău*, Ali Mahmoudi*, Fetze Pijlman†, Tanir Ozcelebi*
*Department of Mathematics and Computer Science, Eindhoven University of Technology
†Research, Signify
Eindhoven, The Netherlands
d.d.tigau@student.tue.nl, a.mahmoudi@tue.nl, fetze.pijlman@signify.com, t.ozcelebi@tue.nl

***Abstract*—Human Activity Recognition (HAR) in smart environments enables assisted living, health monitoring, and home automation. Traditional HAR methods often rely on handcrafted features or shallow models that do not generalize across different sensor layouts. Sequence-based deep learning models, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs), improve temporal modeling but typically treat sensor streams as independent channels. As a result, these models do not explicitly encode spatial relationships between sensors or the structural constraints imposed by the environment, limiting their ability to model sensor interactions driven by the environment's physical structure. We study Graph Convolutional Networks (GNNs) for HAR using ambient sensor data, modeling sensors as nodes and their spatio-temporal interactions as edges. Our graph construction uses domain-informed room adjacency and temporal co-activation, with shared node and edge encoders. We evaluate a Graph Convolutional Network (GCN), a Graph Attention Network (GAT), and a hybrid GAT–Long Short-Term Memory model (GAT-LSTM) on two CASAS smart homes (HH111 and HH101). Sparse, domain-informed graphs consistently outperform fully connected graphs, and longer temporal windows improve robustness. GAT-LSTM attains the best classification performance, while GCN provides the most efficient baseline. Overall, combining spatial attention with temporal modeling yields a strong framework for HAR while highlighting remaining challenges for multi-resident activities and data-driven room adjacency.**

***Index Terms*—graph neural networks, recurrent neural networks, human activity recognition, smart homes, spatio-temporal modeling, temporal sequence learning**

I. Introduction

Graph Neural Networks (GNNs) have emerged as a promising class of models that naturally incorporate relational and structural information in data. In the context of ambient Human Activity Recognition (HAR), GNNs allow sensors to be modeled as graph nodes and their spatio-temporal interactions as edges, enabling the system to learn context-aware representations. This paradigm shift moves beyond isolated feature extraction toward a more holistic understanding of sensor behavior.

This paper explores the use of GNNs for HAR in single-resident smart homes with non-intrusive ambient sensors. We model the sensor network as a spatio-temporal graph and evaluate three key architectures: Graph Convolutional Network (GCN), Graph Attention Network (GAT), and a hybrid of Graph Attention Network with Long Short-Term Memory (GAT-LSTM). The central objective of this study is to analyze how graph construction strategies, temporal context windows, and GNN architecture choices influence recognition performance and computational efficiency in single-resident smart homes. Each model is evaluated on its ability to accurately classify activities of daily living (ADLs) while maintaining computational efficiency.

We evaluate our approach on two CASAS smart homes, HH111 and HH101 [1], which provide timestamped sensor activations with activity labels. Graphs combine spatial priors (room adjacency) with temporal co-activation patterns between sensors. Node features capture semantic and statistical properties, and edge features encode temporal correlation and overlap.

Our experimental results show that sparse, domain-informed graphs outperform fully connected graphs in both accuracy and training time. Furthermore, the GAT-LSTM architecture consistently yields the highest classification performance, demonstrating the benefit of combining spatial attention with temporal modeling. We also analyze how graph design, dataset structure, and multi-resident activity classes affect model generalizability and performance.

The contributions of this paper are as follows:

- We propose a framework for graph-based HAR using ambient sensor data, integrating spatial and temporal dynamics via GCN, GAT, and GAT-LSTM architectures.
- We design and evaluate an edge-aware graph construction strategy that incorporates sensor type, location, and co-activation statistics.
- We conduct extensive experiments on two real-world smart home datasets and assess the impact of graph topology, model architecture, and temporal windowing on HAR performance.
- We demonstrate that GNN-based models, particularly when coupled with temporal sequence modeling, achieve superior performance over standard graph models in classifying ADLs.

Specifically, we address the following research question: *How do graph construction strategies, temporal context windows, and GNN architectures influence recognition performance and efficiency in single-resident smart homes?*

Code is available online.[1]

II. Related Work

Ambient sensor-based HAR offers a non-intrusive alternative to wearable or vision-based systems that preserves privacy while enabling continuous monitoring in smart environments [2]. However, sensor data is often sparse, noisy, and context-dependent, making it difficult for rule-based or shallow ML methods to generalize across households [3]. These approaches typically rely on handcrafted features or fixed heuristics and do not scale well to environments with different sensor configurations or activity patterns.

To address these challenges, recent work has turned to deep learning methods capable of extracting temporal and semantic patterns directly from raw data. Long Short-Term Memory (LSTM) networks have shown success in modeling temporal dependencies in activity sequences, particularly for ambient sensor streams [4]. Similarly, unsupervised models like Hidden Markov Models (HMMs) have been explored to reduce reliance on annotated data [5]. Despite their promise, these approaches often treat sensor data as flat or sequential inputs, failing to account for the spatial relationships between sensors distributed across different rooms. This limitation has led to the increasing adoption of GNNs in HAR, which offer a natural way to model relational structures.

GNNs have demonstrated strong performance in various domains, including social networks and recommendation systems [6]. Mondal et al. [7] applied GCNs to smartphone-based activity data and achieved high classification accuracy by leveraging spatial correlations. Raj et al. [8] extended this approach by integrating GCNs with GRUs to model spatio-temporal patterns in motion capture datasets. However, these approaches focus on wearable or motion-capture settings and do not represent the physical layout of ambient sensors in a home.

Recent work has further explored the use of GNNs for activity recognition in smart environments. Su and Chen [15] propose a GNN-based approach for inferring activities of daily living by modeling human-object interactions, focusing on relational structures between entities involved in activities. While effective for capturing interaction patterns, their approach assumes explicit object-level representations and does not address the challenges of sparse, event-driven ambient sensor data in smart homes.

GATs further enhance representational capacity by allowing models to dynamically weigh node interactions based on learned attention coefficients [9]. This mechanism is particularly valuable in heterogeneous environments where sensor importance varies. GATs have shown improved robustness within the evaluated setting over GCNs in settings with noisy or irregular graph structures [10]. However, empirical evidence suggests their effectiveness can vary depending on the dataset and structural consistency.

[1]https://github.com/D4v321/graph_neural_networks_for_human_activity_recognition_in_smart_environments

To better model temporal dynamics in graph-structured data, hybrid architectures combining GNNs with Recurrent Neural Networks (RNNs) have emerged. In traffic forecasting, Zhu et al. [11] used a GAT-LSTM model to jointly capture spatial and temporal patterns, outperforming both standalone GAT and LSTM models. Orji et al. [12] applied a similar architecture to electricity load forecasting and reported substantial gains in prediction accuracy. These results suggest that separating spatial and temporal learning into modular components can be an effective design strategy.

In the context of ambient HAR, relatively few studies have applied GNN-based architectures to smart home sensor networks. Plötz [13] proposed a graph-guided model for co-activation patterns, demonstrating performance gains on CASAS datasets. However, the challenge of constructing meaningful graph topologies remains. Most studies rely on handcrafted or fully connected graphs, which can either miss important structure or introduce noise. This paper builds on these insights by evaluating multiple GNN architectures, including GCN, GAT, and GAT-LSTM, on real-world ambient HAR datasets. Our approach incorporates domain-informed graph construction, temporal modeling, and comprehensive performance analysis to identify scalable and generalizable strategies for HAR in smart environments.

Despite growing interest in graph-based approaches for HAR, most existing methods rely on wearable or vision-based data or employ heuristically defined graph structures that do not explicitly reflect the physical layout of smart environments. Moreover, while temporal modeling has been extensively explored through sequence-based architectures, fewer studies have investigated the explicit separation of spatial and temporal learning in ambient sensor settings. This work addresses these gaps by evaluating domain-informed graph construction strategies and a hybrid spatial-temporal architecture for activity recognition using ambient smart home sensors.

III. Methodology

A. Research Approach

The process began with a literature review to identify limitations in traditional HAR systems and the potential advantages of GNN-based modeling. Two real-world datasets (HH111 and HH101) from the CASAS project were then selected as independent testbeds. A standardized preprocessing pipeline was developed to filter sensor events, normalize timestamps, and map activity labels into broader categories.

Graphs were constructed from overlapping temporal windows, where nodes represent sensors and edges encode spatial and temporal interactions. A manually created room adjacency matrix guided the main set of experiments, with a data-driven adjacency variant explored as an auxiliary component.

B. Datasets

We evaluate on two single-resident smart home datasets from the CASAS project: HH111 and HH101. Both consist of timestamped binary motion and door sensor activations plus

light sensors, annotated with daily activities. To standardize activity labels across the CASAS datasets and reduce noise, the original fine-grained annotations were consolidated into 16 consistent activity classes: Relax, Sleep, Leave Home, Other Activity, Cook, Bed Toilet Transition, Personal Hygiene, Dress, Work, Watch TV, Wash Dishes, Enter Home, Read, Eat, Wake Up, and Entertain Guests.

This mapping groups semantically related sub-activities into unified classes (e.g., *Cook Breakfast*, *Cook Lunch*, and *Cook Dinner* $\rightarrow$ *Cook*), while resolving dataset-specific labeling inconsistencies. Sensor data was filtered to include Motion, Door, and Light events, and 'Ignore' labels and ambient light intensity readings were excluded to focus on intentional human-environment interactions.

Graphs were split chronologically into 65% training and 35% test at the graph level, replicating a realistic scenario where models are trained on past sequences and evaluated on future ones. Within the training partition, 10% was further set aside as a validation split for hyperparameter tuning. HH111 contains more sensors and more variable daily patterns, whereas HH101 has fewer sensors and a more routine structure, providing complementary evaluation scenarios. The two datasets thus complement each other: HH111 provides stable training conditions, while HH101 introduces structural and temporal variations that test the models' generalizability. Each dataset is processed independently and split chronologically into training and test sets to reflect forward-in-time prediction.

C. Preprocessing Pipeline

Raw sensor logs from CASAS were standardized using a reproducible preprocessing pipeline. The procedure filtered events to retain only binary activations from motion, door, while continuous light intensity sensors and 'Ignore' flagged events were discarded. Each event was parsed into seven fields: timestamp, sensor ID, area, location, binary state, sensor type, and label.

The cleaned event streams were stored in CSV format and served as the input for temporal segmentation and graph construction. This standardized pipeline ensured consistent preprocessing across both HH111 and HH101, while preserving each dataset's structural differences.

D. Graph Construction

Each dataset was segmented into sliding windows of 60 or 300 seconds, with a 10-second stride. For each window, a spatio-temporal graph was created in which nodes represent sensors and edges capture spatial proximity and temporal co-activations. Each graph is labeled with the activity associated with the final event in the window. The use of overlapping windows combined with the final-event labeling was intentionally designed to support dense and continuous activity predictions. Labeling each window based on the final event further aligns the prediction with the most recent observed activity, making the approach suitable for runtime deployment scenarios that require up-to-date activity recognition.

TABLE I: Summary of node and edge features used in graph construction.

Node features	Sensor type; room location (one-hot); last state; activation frequency; time since last activation; ON duration (min/max); ON–OFF cycles; variance of intervals; total ON time; hour-of-day (sine/cosine)
Edge features	Temporal correlation (asymmetric co-activation); temporal overlap (Jaccard similarity of active time buckets)

1) Node features: captured both categorical metadata and temporal statistics. Categorical features included sensor type and room location, one-hot encoded for consistency. Temporal features included activation dynamics such as last recorded state, activation frequency, time since last activation, ON duration, and cyclical encodings of the hour of day.

2) Edge features: represented the strength and type of interaction between sensors. Two measures were used, as described below:

(i) *Temporal correlation*

$$\text{temporal\_correlation}_{i \to j} = \frac{N_{i,j}}{N_i}$$

$$\text{temporal\_correlation}_{j \to i} = \frac{N_{i,j}}{N_j}$$

where:

- N_i is the number of activations of sensor i during the current window
- N_j is the number of activations of sensor j during the current window
- $N_{i,j}$ is the number of times sensors i and j were activated within a 5-second interval

(ii) *Temporal overlap*

- $\mathcal{B}_i \subseteq \{1, 2, \ldots, B\}$ be the set of bucket indices where sensor i is active
- $\mathcal{B}_j \subseteq \{1, 2, \ldots, B\}$ be the set of bucket indices where sensor j is active

Then, the temporal overlap is computed as the Jaccard similarity between these two sets:

$$\text{temporal\_overlap}_{i,j} = \frac{|\mathcal{B}_i \cap \mathcal{B}_j|}{|\mathcal{B}_i \cup \mathcal{B}_j|}$$

This metric ranges from 0 (no co-activation) to 1 (identical active periods).

Table I provides a compact summary of all node and edge features used in graph construction.

3) Graph topology: was constructed using a hybrid strategy. Within-room sensors were fully connected, reflecting dense local co-activity. Between rooms, edges were created only if the rooms were adjacent according to the room adjacency matrix and if their sensors had historically co-activated within 10 seconds at least once in the dataset. The choice of the co-activation interval limit is motivated by the assumption that walking pathways within a home typically do not take longer than 10 seconds between sensors in close proximity. This design retained physically meaningful transitions while

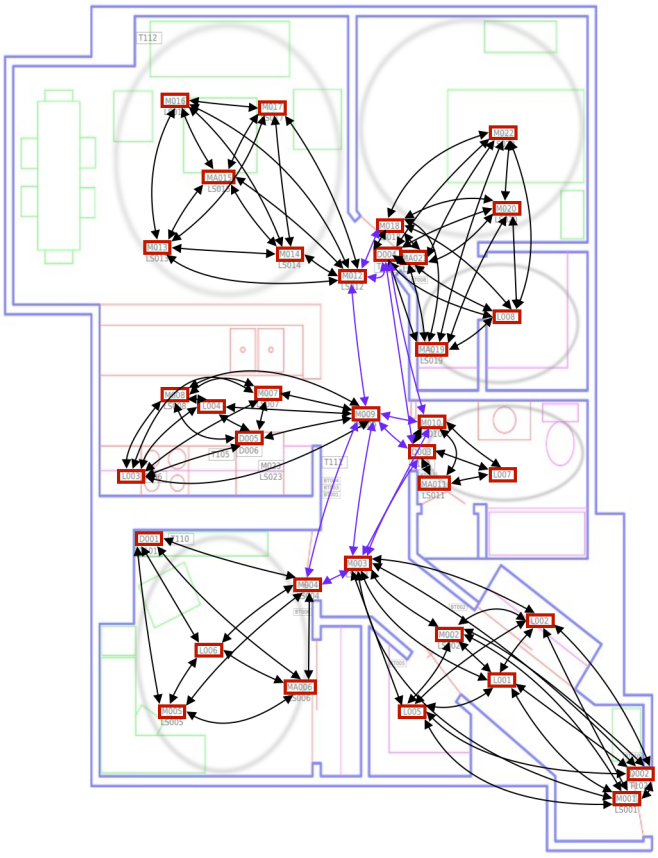
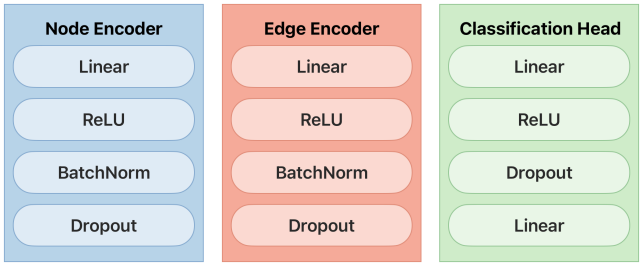


Fig. 1: Floorplan of the CASAS HH111 apartment with overlaid sensors (red rectangles) and constructed edges (intra-room in black, inter-room in purple). The floorplan background originates from the CASAS dataset [1], with sensor connections added for this study.

reducing the number of edges compared to a fully connected graph. A fully connected variant was also implemented as a baseline for comparison.

E. Room Adjacency

Spatial connectivity between sensors was defined using a room adjacency matrix, which specifies whether two rooms are directly connected. This matrix guided the creation of inter-room edges during graph construction, complementing the fully connected intra-room strategy. Two variants were considered: (i) a manual adjacency matrix used in all main experiments, and (ii) a data-driven adjacency matrix explored only as a proof-of-concept. For reliability and interpretability, the manual adjacency matrices were adopted in all primary experiments. Fig. 1 provides a schematic overview of how the adjacency matrix is used to define the edges between the nodes in the HH111 dataset. While the figure offers a conceptual visualization, the final edge set is determined by sequential sensor activations across rooms, subject to a temporal threshold. As such, the actual graph structure may deviate from this representation.



(a) Node Encoder (b) Edge Encoder (c) Classification Head

Fig. 2: Reusable encoder and classification modules used across all models.

1) Manual adjacency: was derived directly from the architectural floor plans provided with the HH111 and HH101 datasets. Two rooms were considered adjacent if a direct passage was possible between them.

2) Data-driven adjacency: was explored as an auxiliary proof of concept. This method inferred adjacency from historical co-activation patterns of motion sensors, recording transitions between rooms when sensors fired within a 30-second interval. This approach yielded plausible spatial layouts without requiring floor plans, but it occasionally omitted valid transitions and proved sensitive to manual threshold settings.

F. Model Architectures

The models were implemented with a modular design, reusing a Node Encoder, Edge Encoder, and Classification Head across all architectures (Fig. 2). Hyperparameter optimization was performed independently per dataset and architecture.

1) Graph Convolutional Network: The GCN serves as the baseline model, extended with edge-aware feature enrichment prior to graph convolution (Fig. 3). Node and edge features are first processed independently through their respective encoders (Fig. 2), ensuring both modalities are projected into a shared hidden space.

Each edge feature e_{ij} is transformed into an embedding via a shared Edge Encoder:

$$e'_{ij} = \text{EdgeEncoder}(e_{ij}) \tag{1}$$

For each node i, encoded edge features are aggregated from incoming and outgoing connections:

$$\text{inc}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} e'_{ji}, \quad \text{out}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} e'_{ij} \tag{2}$$

The hidden node representation is then enriched with the aggregated edge information:

$$\tilde{h}_i^{(l)} = h_i^{(l)} + \text{inc}_i + \text{out}_i \tag{3}$$

The enriched node features are then updated using a normalized graph convolution with self-loops:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\tilde{d}_i \tilde{d}_j}} W \tilde{h}_j^{(l)}\right) \quad (4)$$

where $\tilde{d}_i$ denotes the degree of node i in the graph with self-loops.

After convolution, node embeddings are pooled into a graph-level representation via global mean pooling. This vector is passed to the Classification Head to produce the final activity prediction.

2) Graph Attention Network: The GAT extends the GCN by introducing an attention mechanism that dynamically weights neighbor contributions during message passing (Fig. 4). As in the GCN, node and edge features are first processed by their respective encoders (Fig. 2), ensuring that all inputs lie in a common hidden space.

For each node, attention coefficients are computed from the concatenated embeddings of the node, its neighbors, and the associated edge features. These coefficients are normalized with a softmax function and used to weight the contribution of each neighbor during aggregation. A multi-head attention scheme is employed to increase robustness, enabling the model to learn multiple complementary relational patterns in parallel.

The resulting node embeddings are passed through stacked attention layers, each followed by Batch Normalization, ReLU activation, and Dropout. As with the GCN, a global mean pooling operation produces a graph-level embedding, which is passed to the Classification Head for activity classification.

3) Graph Attention Network with LSTM: The Graph Attention Network with LSTM hybrid model is designed to capture both spatial and temporal dependencies in sensor data (Fig. 5). Each temporal window is first encoded into a graph-level embedding using the GAT encoder described in the previous architecture (see Fig. 4). Node and edge features are processed through the shared encoders (Fig. 2), passed through stacked attention layers, and pooled into a single vector representation.

These graph-level embeddings, each corresponding to a sliding temporal window (60s or 300s with a stride of 10s), are then arranged in chronological order to form fixed-length sequences. The sequence length is treated as a hyperparameter and is optimized independently for each dataset using Optuna. During hyperparameter tuning, sequence lengths between 5 and 50 are explored. The resulting sequence of graph embeddings serves as the input to the Long Short-Term Memory (LSTM) network, enabling the model to capture temporal dependencies across consecutive windows. The final hidden state of the LSTM is passed to the Classification Head to produce activity predictions.

Both the graph attention and recurrent components used in this work follow their standard formulations as introduced in prior literature. The attention mechanism in the GAT layers is defined according to Veličković et al. [9], and the LSTM component follows the formulation proposed by Hochreiter and Schmidhuber [14]. No modifications are made to the mathematical definitions of GCN, GAT, or LSTM themselves. Instead, the contribution of this work lies in its integration into a unified and modular architecture tailored to HAR from ambient sensor data. This includes the explicit use of node and edge encoders, a consistent block-based design shared across models, and a graph construction strategy that preserves spatial and temporal structure.

G. Experimental Design

The experimental setup was designed to ensure fair and real-world comparisons across datasets and model architectures while addressing class imbalance and temporal dependencies.

1) Class imbalance: Skewed activity distributions were handled with class-weighted cross-entropy loss, assigning weights inversely proportional to class frequencies. Oversampling was explored but discarded due to higher computational burden and convergence issues.

2) Data splits: Each dataset was split chronologically into 65% training and 35% test, preserving temporal order and reflecting a forward-in-time prediction setting. The test set was held out and not used during hyperparameter tuning or model selection.

Hyperparameter tuning was performed on the training portion only. For this purpose, 10% of the training data was set aside as the validation set, while the remaining 90% was used for training during each trial. Model configurations were evaluated based on their performance on this validation split, and early stopping was applied to prevent overfitting.

After selecting the best hyperparameters, K-fold cross-validation was applied on the entire training data to obtain a more robust estimate of the model's performance. The training data was partitioned into multiple folds, and in each iteration, a model was trained on a subset of the data and validated on the remaining fold. Early stopping was applied within each fold using its corresponding validation split.

Finally, the model was trained on the full training data using the final hyperparameters. All final performance results reported in this paper are based on the held-out test set.

3) Hyperparameter tuning: For each dataset and architecture, Bayesian optimization with Optuna searched over learning rate, dropout, weight decay, depth, hidden size, and (for GAT/GAT-LSTM) attention heads and LSTM size. Configurations were selected based on validation accuracy.

4) Training procedure: Models were trained with Adam for up to 100 epochs, using early stopping (patience=3). Training was performed in mini-batches using PyTorch Geometric's DataLoader.

5) Evaluation metrics: We report test accuracy, macro F1-score (chosen because it is a relevant metric for datasets presenting class imbalance), training time per epoch, and graph creation overhead to quantify accuracy-efficiency trade-offs. In all results tables, *CV Acc.* refers to the average classification accuracy obtained on the validation set during cross-validation or hyperparameter tuning, while *Test Acc.* and *Test F1* are reported on the held-out test set.

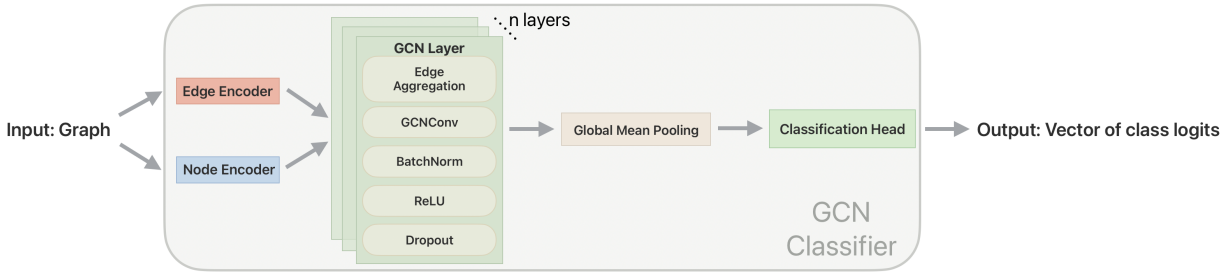


Fig. 3: GCN architecture using node and edge encoders, stacked convolutional layers, and a classification head.

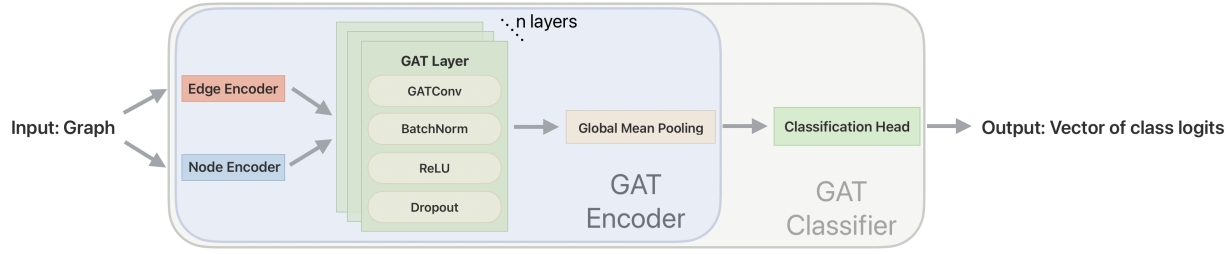


Fig. 4: GAT architecture with multi-head attention applied to node and edge embeddings, followed by pooling and a classification head.

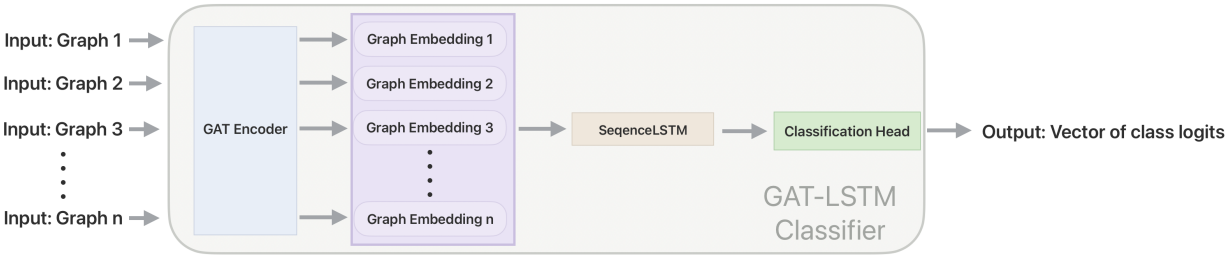


Fig. 5: GAT-LSTM hybrid architecture combining spatial graph encoding with sequential temporal modeling.

H. Implementation Details

Experiments were run in Google Colab Pro using Python 3.11, Jupyter Notebooks, and NVIDIA T4 GPUs for training, with CPU instances for preprocessing and graph construction. Models were implemented in PyTorch Geometric with Optuna for hyperparameter tuning and standard libraries (Pandas, NumPy) for preprocessing and feature engineering. Graphs were cached for reuse, and we fixed random seeds to ensure reproducibility.

IV. Results

A. Room Adjacency Matrix

We first evaluated the impact of the room adjacency strategy on graph construction and classification performance. Table II reports results for the GCN baseline when using manually defined adjacency matrices versus data-driven ones.

The data-driven adjacency approach yielded comparable or slightly higher test scores than manual adjacency (e.g., HH111 F1: 48 vs. 46) but was sensitive to threshold settings and occasionally excluded plausible transitions, reducing stability across runs. It also reduced file sizes and processing time.

TABLE II: GCN performance using manual vs. data-driven room adjacency.

Dataset	Strategy	CV Acc. (%)	Test Acc. (%)	Test F1 (%)
HH111	Manual	$\mathbf{74 \pm 3}$	54	46
	Data-driven	72 ± 2	**56**	**48**
HH101	Manual	67 ± 1	71	46
	Data-driven	$\mathbf{69 \pm 1}$	**72**	**49**

Because of this instability, all subsequent experiments use manual adjacency matrices derived from floor plans, providing consistent and interpretable graph structures.

B. Graph Construction Strategy

We next compared domain-informed sparse graphs, constructed using manual adjacency, against fully connected graphs where all sensors were linked regardless of spatial or temporal plausibility. Table III summarizes classification performance for the GCN baseline.

Sparse graphs consistently achieved higher macro F1-scores, reduced file size and shorter creation times than fully connected graphs, confirming that restricting edges to physically

TABLE III: GCN performance: manual adjacency (sparse) vs. fully connected graphs.

Dataset	Strategy	CV Acc. (%)	Test Acc. (%)	Test F1 (%)
HH111	Manual	$\mathbf{74 \pm 3}$	**54**	**46**
	Fully	71 ± 1	53	39
HH101	Manual	67 ± 1	**71**	**46**
	Fully	$\mathbf{69 \pm 2}$	69	43

TABLE IV: GCN performance by temporal context window.

Dataset	Window (s)	CV Acc. (%)	Test Acc. (%)	Test F1 (%)
HH111	60	69 ± 3	**55**	42
	300	$\mathbf{74 \pm 3}$	54	**46**
HH101	60	65 ± 3	69	**46**
	300	$\mathbf{67 \pm 1}$	**71**	**46**

TABLE V: Performance comparison across architectures.

Dataset	Model	CV Acc. (%)	Test Acc. (%)	Test F1 (%)
HH111	GCN	74 ± 3	54	46
	GAT	74 ± 2	56	44
	GAT-LSTM	$\mathbf{77 \pm 3}$	**57**	**48**
HH101	GCN	67 ± 1	71	46
	GAT	69 ± 2	74	58
	GAT-LSTM	$\mathbf{74 \pm 2}$	**75**	**58**

meaningful and frequently co-activated sensors reduces noise and improves classification.

Based on this, sparse, domain-informed graphs were adopted for all subsequent experiments.

C. Temporal Context Window Length

We compared short (60-second) and long (300-second) sliding windows with a stride of 10 seconds using the GCN baseline (Table IV). Longer 300-second windows delivered higher or more stable macro F1-scores, particularly on HH111 (46 vs 42), by better capturing activity transitions. Short windows reduced processing time and storage, but often lacked sufficient context to distinguish similar activities. We therefore adopted the long context windows as the default configuration for subsequent experiments.

Based on this trade-off, all subsequent experiments adopted 300-second windows as the default configuration.

D. Architecture Comparisons

We next compared the three model architectures GCN, GAT, and GAT-LSTM using the same preprocessing pipeline, 300-second windows, and manual adjacency graphs. Table V reports classification performance across both datasets.

GAT-LSTM consistently achieved the highest test performance, with macro F1-scores of 48 on HH111 and 58 on HH101. GAT provided gains over GCN on HH101 but showed a slight reduction in F1-score on HH111. The GCN remained the most efficient baseline, providing competitive accuracy with lower training cost.

Table VI summarizes the average training time per epoch, tuned depth, and hidden dimension size selected for each model. Training efficiency was influenced primarily by the tuned depth rather than the architecture alone. For example, GAT on HH111 required nine layers and averaged 287 seconds per epoch, while GCN trained with a single layer in only 22 seconds.

TABLE VI: Training resources and selected depth.

Dataset	Model	Time (s/epoch)	Layers	Hidden Dim
HH111	GCN	**22**	1	252
	GAT	287	9	120
	GAT-LSTM	297	2	150
HH101	GCN	146	7	141
	GAT	**120**	3	256
	GAT-LSTM	129	2	126

E. Performance Across Datasets

A comparison of results between HH111 and HH101 highlights how dataset characteristics affect model performance. As shown in Table V, HH111 yielded more stable cross-validation accuracy, reflecting its denser temporal coverage and more balanced activity distribution. For example, GCN achieved $74\% \pm 3$ CV accuracy on HH111 compared to $67\% \pm 1$ on HH101.

In contrast, HH101 produced higher test accuracy but lower macro F1-scores, indicating difficulties with minority activities. The dataset contains fewer sensors, lacks light switch devices, and has a different spatial layout and activity distribution.

Despite these challenges, the relative ranking of architectures was consistent across datasets: GAT improved robustness by weighting sensor interactions, and GAT-LSTM achieved the strongest classification performance. These results suggest that while absolute performance is constrained by dataset structure, the proposed architectures show promising generalization across the evaluated datasets.

F. Confusion Matrix Analysis

To further analyze class-level performance, Fig. 6 presents confusion matrices for the GAT-LSTM model using a 300s temporal context and a 10s prediction stride. Overall, the model achieves strong diagonal dominance, indicating high per-class accuracy for most activities.

Activities with distinctive sensor activation patterns, such as *Enter Home*, *Leave Home*, *Personal Hygiene*, *Sleep*, and *Work* are classified with high accuracy. In contrast, confusion is more prominent between semantically similar or low-activity classes, such as *Relax*, *Read*, and *Other Activity*.

Additionally, activities with overlapping sensor usage patterns, such as *Cook* and *Wash Dishes*, exhibit confusion, reflecting the inherent ambiguity of ambient sensor data. These patterns are consistent across both datasets, although HH101 shows slightly higher confusion for certain classes, such as Entertain Guests and *Work* which lack samples in the training split. Thus, *Work* is predicted as *Other Activity*, for example.

V. Conclusion

This study evaluated GCN, GAT, and GAT-LSTM architectures for ambient sensor-based HAR using sparse, domain-informed graph structures. Graphs based on room adjacency

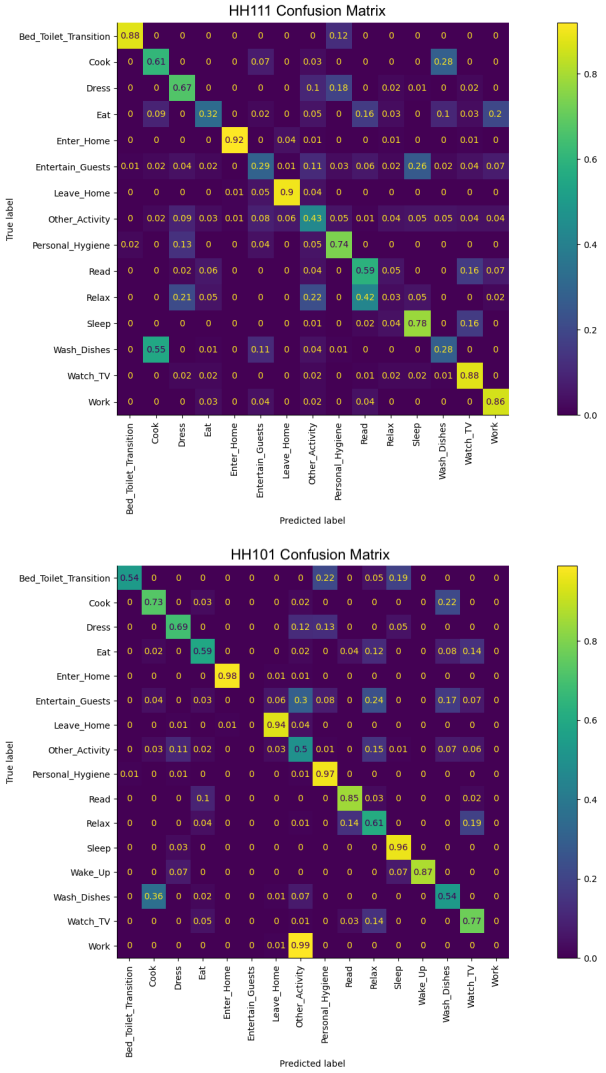


Fig. 6: Confusion matrices for the GAT-LSTM model with a 300s context window and 10s stride on HH111 (top) and HH101 (bottom).

and temporal sensor co-activation outperformed fully connected baselines, indicating that simple spatial priors can reduce noise, improve efficiency, and enhance interpretability. Within the evaluated setting, longer 300-second context windows were more robust than 60-second windows because they captured extended activity transitions. Among the tested models, GAT-LSTM achieved the highest recognition performance, suggesting that combining spatial attention with temporal modeling provides measurable benefits over purely sequential or fully connected designs.

These results have several implications for HAR system design in smart environments. They underline the value of modeling both spatial structure and temporal context, since architectures that explicitly represent these dependencies performed better than simpler alternatives. They also show that domain-informed graph construction can improve performance while preserving computational efficiency.

The study also highlights several limitations and directions for future work. Data-driven adjacency estimation was unstable across runs, and the models did not generalize to multi-resident scenarios, as illustrated by the unseen Entertain Guests activity in HH101. In addition, all experiments were conducted offline, so real-time responsiveness, deployment-time computational performance, scalability to edge devices, and energy efficiency remain untested. Future research should therefore examine generalization across a wider range of environments, including multi-resident households and more diverse sensor configurations; investigate more reliable or adaptive graph construction methods, including fully data-driven approaches; develop models that can distinguish interactions among multiple residents; and evaluate deployment-oriented settings with attention to latency, resource constraints, continuous operation, and real-world scalability.

Acknowledgment

This work is part of the IntelLight project (TKI HTSM) with the support of Signify.

References

[1] D. Cook, “CASAS Smart Home Datasets,” Washington State University, 2025. [Online]. Available: https://casas.wsu.edu/datasets/

[2] L. Hicks, A. Ruiz-Garcia, V. Palade, and I. Almakky, “Self-supervised transformers for activity classification using ambient sensors,” arXiv preprint arXiv:2011.12137, 2020.

[3] B. Yuan, J. Herbert, and Y. Emamian, “Smartphone-based activity recognition using hybrid classifier,” in Proc. 4th Int. Conf. Pervasive Embedded Comput. Commun. Syst., 2014.

[4] D. Liciotti, M. Bernardini, L. Romeo, and E. Frontoni, “A sequential deep learning application for recognising human activities in smart homes,” Neurocomputing, vol. 396, pp. 501–513, Jul. 2020.

[5] R. Gómez-Ramos, J. Duque-Domingo, E. Zalama, and J. Gómez-García-Bermejo, “An unsupervised method to recognise human activity at home using non-intrusive sensors,” Electronics, vol. 12, no. 23, p. 4772, 2023.

[6] J. Zhou et al., “Graph neural networks: A review of methods and applications,” AI Open, vol. 1, pp. 57–81, 2020.

[7] R. Mondal, D. Mukherjee, P. K. Singh, V. Bhateja, and R. Sarkar, “A new framework for smartphone sensor-based human activity recognition using graph neural network,” IEEE Sensors J., vol. 21, no. 10, pp. 11461–11468, May 2020.

[8] M. S. Raj, S. N. George, and K. Raja, “Leveraging spatio-temporal features using graph neural networks for human activity recognition,” Pattern Recognit., vol. 146, Art. no. 110301, 2024.

[9] P. Veličković et al., “Graph attention networks,” arXiv preprint arXiv:1710.10903, 2017.

[10] K. Fountoulakis, A. Levi, S. Yang, A. Baranwal, and A. Jagannath, “Graph attention retrospective,” J. Mach. Learn. Res., vol. 24, no. 246, pp. 1–52, 2023.

[11] T. Zhu, M. J. L. Boada, and B. L. Boada, “Adaptive graph attention and long short-term memory-based networks for traffic prediction,” Mathematics, vol. 12, no. 2, p. 255, 2024.

[12] U. Orji, Ç. Güven, and D. Stowell, “Enhanced load forecasting with GAT-LSTM: Leveraging grid and temporal features,” arXiv preprint arXiv:2502.08376, 2025.

[13] T. Plötz *et al.*, “Using graphs to perform effective sensor-based human activity recognition in smart homes,” *Sensors*, vol. 24, no. 12, p. 3944, 2024.

[14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] P. Su and D. Chen, “Adopting graph neural networks to analyze human–object interactions for inferring activities of daily living,” *Sensors*, vol. 24, no. 8, p. 2567, 2024.