Investigating the GRUGCN model for Human Activity Recognition and Prediction

Kishalan Pather
pthkis001@myuct.ac.za
University of Cape Town
Cape Town, South Africa

ABSTRACT

Human Activity Recognition (HAR) Prediction (HAP) have become prominent research domains within machine learning with applications in healthcare, surveillance, and smart homes. However, effectively modeling the complex spatio-temporal dependencies from time series data remains a significant challenge. Traditional deep learning models often struggle to explicitly capture underlying relationships between different sensor placements. This paper investigates the performance of an STGNN model, specifically the GRUGCN model which is designed to explicitly model spatial and temporal dynamic correlations. We conduct a comprehensive evaluation of the GRUGCN model on the publicly available dataset, CASAS. Our findings demonstrate that the standard GRUGCN model does not consistently outperform traditional deep learning models. We show, however, that its effectiveness can be enhanced with the addition of adaptive mechanisms that dynamically construct the graph structure for activity recognition and prediction.

CCS Concepts

• Computing methodologies →Artificial Intelligence; Neural networks.

Keywords

Human Activity Recognition, Deep Learning, Time Series, Spatial-Temporal Graph Neural Networks

1 INTRODUCTION

Sensor based Human Activity Recognition (HAR) and Prediction (HAP) are emerging fields in research. Due to the addition of IoT devices, wearables and ambient sensors, that have become a part of our daily lives. By interpreting data from these sensors, we can develop useful applications ranging from remote health monitoring for the elderly to creating energy-efficient smart home environments based on past behaviour. Despite this potential, accurately classifying and predicting human activities from sensor data presents significant challenges as the data is complex, consisting of multivariate time series streams from a network of sensors in a smart home.

Earlier approaches to Human Activity Recognition relied heavily on handcrafted feature extraction techniques combined with traditional machine learning methods, such as k-Nearest Neighbors (k-NN) and Hidden Markov Models (HMMs). While these models provided some success, they struggled to capture complex and dynamic activities, limiting their effectiveness in real-world applications [6].

The advancement of Deep Neural Networks (DNNs) improved on these results, however models like CNNs and LSTMS often process these streams of data in isolation. While these networks effectively capture temporal dependencies, they largely neglect the spatial relationships between these sensors. For example, an LSTM model may learn the sequence of a kitchen sensor followed by a dining room sensor, but it fails to understand the inherent physical distance and connection between these two locations.

This issue is remedied with the implementation of Spatio-Temporal Graph Neural Networks (STGNNs) which are designed to fully capture both spatial and temporal dependencies, often leading to better performance. This paper directly addresses this challenge of modeling these complex dependencies by implementing and evaluating a specific STGNN: the Gated Recurrent Unit - Graph Convolutional Network (GRUGCN) [3].
While HAR is a well-studied area, this paper extends the evaluation to Human Activity Prediction (HAP), a less explored and more complex task. Forecasting the next activity, rather than only identifying the current one, represents a significant research gap, as it requires the model to learn not just patterns but also the causal and transitional logic of human behavior.

Since the GRUGCN model takes in a predefined, static map of sensor relationships, which may not always be the most optimal way to model these connections. We also explore how performance is impacted by implementing a more dynamic approach. We investigate an adaptive method where the model learns the most important sensor relationships directly from the data, allowing it to discover non-obvious connections and potentially improve results.

Building upon this evaluation, we also conduct an analysis on the base GRUGCN model's performance. We conduct a detailed error analysis using confusion matrices to identify problematic activity classes and systematic misclassification patterns. Furthermore, as done in recent STGNN studies, we extract and visualize the learned relationships between sensors in our adaptive model. Through this process, we can gain insights into the model's decision-making process and uncover patterns that may deepen our understanding of the HAR/HAP problem.

The remainder of this paper is structured as follows: Section 2 provides the Background and Related Work. Here we discuss how the problem is formulated, previous DNN work for HAR/HAP and STGNN work for HAR/HAP and an overview of the GRUGCN model. In Section 3, we discuss the materials and methods. We go over the datasets selected, our preprocessing pipeline, how we set our baseline models, and the GRUGCN model implementations. In section 4 we discuss the results from our experiments for HAR and HAP respectively. We also conduct an analysis and provide explanations for our findings. Finally in section 5, we summarise our findings and observations, conclude our work and discuss potential limitations.

Research Questions

1. How does the STGNN model, GRUGCN perform against traditional DNN models and the established GWN for the tasks of human activity recognition and prediction?
2. Does implementing an adaptively learned adjacency matrix mechanism to the GRUGCN model improve its performance?

2 BACKGROUND AND RELATED WORK

2.1 Problem Formulation

HAR is framed as a supervised multi-class classification problem, where the model receives a sequence of sensor readings $S = \{S_1, \dots, S_n\}$ with each $S_1 = \{S_i, 0, \dots, S_{i,t}\}$ representing the time-series data from sensor i from $time = 0$ to $time = t$. More concisely, the input takes the form:

$$S = \begin{pmatrix} S_{1,0} & S_{2,0} & \cdots & S_{n,0} \\ \vdots & \vdots & \ddots & \vdots \\ S_{1,t} & S_{2,t} & \cdots & S_{n,t} \end{pmatrix}$$

The goal is to learn a function $F\ (S) = \hat{A} = \{\hat{A}0, \dots, \hat{A}t\}$ where

$\hat{A}_i$ denotes the model's classification of the activity that took place at time i, from a set of possible activities $A = \{A_1, \dots, A_k\}$ (e.g., "cooking").

The model F is trained to minimize the loss $L(F\ (S), A^*)$, where $A^* = \{A_0, \dots, A_t\}$ is the ground truth activity sequence, and L measures the discrepancy between the model's classifications and the actual activities.

HAP focuses on forecasting future activities based on recent sensor patterns. The input remains the recent sequence of sensor readings $S = \{S_1, \dots, S_n\}$, and the output is a predicted sequence of future activities $\hat{A}_{future} = \{\hat{A}_{t+1}, \dots, \hat{A}_{t+h}\}$, where h is the prediction horizon.

Our research has trained and evaluated CNN, LSTM, and STGNN models to learn the most accurate function F for classifying and predicting human activities from sensor data.

2.2 Deep Neural Networks for HAR and HAP

2.2.1 Convolutional Neural Networks (CNN)

CNNs are one of the most researched deep learning techniques. They are renowned for their effectiveness in tasks such as image classification [2]. By applying convolutional filters, they act as powerful feature extractors capable of capturing local temporal patterns within sensor data. However, a significant limitation of standard CNNs is that they process the multivariate time-series sensor data as a simple array, ignoring the spatial relationship between different sensors in a smart home.

2.2.2 Long Short-Term Memory Model (LSTM)

LSTMs [4] are a specialised type of recurrent neural network developed to overcome the vanishing gradient problem and effectively model long-range dependencies like in sequential data. The core component of an LSTM is its memory cell, which is controlled by three distinct gates: an input, an output, and a forget gate. These gates work together to regulate the flow of information, enabling the cell to selectively retain relevant information for long periods and discard irrelevant data. This mechanism allows LSTMs to excel at capturing the temporal dynamics such as the "when" of an activity sequence. However, like CNNs, their architecture does not inherently model the "where", the spatial structure and relationships within the sensor network.

2.3 Spatio-Temporal Graph Neural Networks for HAR and HAP

To overcome the limitations of traditional DNNs, recent research has shifted towards methods that can learn both spatial and temporal dependencies simultaneously. STGNNs are designed for this purpose, explicitly capturing both spatial and temporal dependencies making them highly suitable for multi-sensor applications [12] While STGNNs have been most notably used in domains such as traffic flow prediction [11], their utility extends directly to the field of HAR and HAP.

the most notable STGNN model in this domain is GraphWaveNet (GWN) [9]. Originally developed for traffic forecasting, its principles are directly applicable to any spatio-temporal modeling task, including sensor-based HAR.

GWN integrates Graph Convolutional Networks (GCNs) to capture spatial features with Temporal Convolutional Networks (TCNs) for temporal patterns. The GCN component contains an adaptive adjacency matrix that allows the model to learn non-obvious relationships between nodes or sensors directly from the data. This makes GWN an efficient architecture that has become a significant benchmark in the field.

2.4 STGNN Problem Formulation

To apply the STGNN models we must explicitly define the graph structure and signals that propagate though it.

We model the smart home sensor network as a graph $G = (V, E, A)$ where:

- Nodes (V): The set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ represents the N sensors in the smart home. Each node corresponds to a unique sensor.
- Edges (E): The set of edges E represents the spatial relationships or functional correlations between sensors.
- Adjacency Matrix (A): These relationships are encoded in an adjacency matrix $A \in R^{N \times N}$. An entry $A_{ij} = 1$ signifies a connection from sensor i to sensor j.

With this framework, the HAR and HAP tasks are defined as learning a function F that maps the historical graph signals and the graph structure to a set of activity labels. For HAR, the goal is to predict the current activity label:

$$y^t = F(X; A)$$

For HAP, the goal is to forecast a sequence of future activities over a horizon H:

$$(y^{t+1}, \ldots, y^{t+H}) = F(X; A)$$

2.5 GRUGCN Architecture

The GRUGCN model integrates graph convolutional networks (GCN) to capture spatial dependencies and gated recurrent units (GRU) to capture temporal dynamics.

The spatial component: Graph Convolutional Network (GCN)
At each time step, the GCN component receives a graph representation of the sensor states. It performs graph convolutions by aggregating feature information from a node's immediate neighbors, as defined by the underlying graph structure. In its standard implementation, the GCN component relies on a fixed, static adjacency matrix which represents the predetermined relationships between sensors in the smart home.

The temporal component: Gated Recurrent Unit (GRU)
The spatially aware feature representation generated by the GCN is then passed to the GRU layer. The GRU, an advanced variant of a Recurrent Neural Network, uses gating mechanisms to effectively control the flow of information across time steps and update its hidden state. This process enables the model to learn and remember temporal patterns from the sequence of spatially filtered inputs.

This process repeats for each time step in the sequence allowing the GRUGCN model to learn the evolution of spatial patterns over time. These two combined components make the model a strong candidate for tasks were both spatial and temporal dynamics are important such as sensor based HAR and HAP.

3 MATERIALS AND METHODS

3.1 Datasets

We selected the Aruba and Milan datasets as they represent two contrasting environments within the CASAS project [10]. Aruba provides a large, long-duration dataset with more balanced activities, enabling models to learn robust temporal patterns. In contrast, Milan is smaller, noisier, and includes unique challenges such as the presence of a pet, making it ideal for testing model robustness under sparse and noisy conditions. Together, they offer complementary perspectives: one emphasizing stability and scale, the other stressing adaptability under difficult conditions. Larger CASAS datasets were not chosen due to their multi-resident setups, which introduce ambiguity beyond the scope of this project. The CASAS datasets used in this study are publicly available and anonymised, with all personally identifiable information removed thus ethical clearance was not needed.

The Milan dataset was collected over several months. This environment was equipped with 33 sensors including motion, door and item sensors and has 16 distinct activity classes such as "Cook," "Work," "Sleep," "Relax," and "Bed to Toilet.'

The Aruba dataset contains data over a period of nearly eight months and has 39 sensors including motion sensors, magnetic door sensors and temperature sensors. The dataset has 12 distinct activity classes that are annotated such as “Meal Preparation," "Eating," "Wash Dishes," "Sleeping," and "Housekeeping.”

Table 1: Key Differences between Aruba and Milan Datasets

Property	Aruba	Milan
Duration	~8 months of data	~2 months of data
Dataset Size	Large (hundreds of thousands of sensor events)	Small (tens of thousands of events)
Number of Activities	12	16
Sensor Count & Layout	39 sensors, spread across larger home (distinct sensor patterns for each room)	33 sensors, compact apartment (high overlap between kitchen, dining, and living areas)
Activity Distribution	More balanced (common activities like *Sleeping, Preparing Breakfast, Bed-to-Toilet* appear many times)	Highly imbalanced (many rare activities like *Eating, Watching TV* appear only a few times)
Sensor Ambiguity	Low–moderate (clear boundaries between areas of activity)	High (same sensors triggered by multiple activities, e.g., *Cooking* vs. *Eating*)

Annotation Quality	Generally consistent across long-term experiment	More noisy and inconsistent (e.g., *Other Activity* used often)
Class Balance	Moderate imbalance but sufficient data for all classes	Severe imbalance; some classes < 20 samples

3.2 Preprocessing and Partitioning

3.2.1 Time series segmentation

A forward fill imputation strategy was first applied to ensure that each sensor maintained a value at every timestamp, thus propagating the last known state forward.

To prepare the data for model input, a sliding-window approach was utilised to segment the continuous stream of data into fixed-length sequences. Each window consisted of a predefined number of events (e.g. 20 events, later tuned through hyperparameter tuning). A single ground truth Y was assigned to each window X based on the activity label that occurred most frequently. In the case of a tie, the most recently occurring activity within the window was selected as the label.

3.2.2 Feature engineering

Our STGNN models were adapted to incorporate three node features for each sensor at every event:

- Sensor State (V): The current state of the sensor (e.g., ON/OFF, temperature).
- Global elapsed time (Tg): The time since the last event from any sensor in the network, which captures the overall level of activity in the environment.
- Sensor elapsed time (Ts): The time since a given sensor was last triggered, indicating the recency of interaction with that particular sensor.

3.2.3 Data partitioning and normalisation

We employed a forward-chaining cross-validation method, as it preserves the temporal order of our time series data and avoids the leakage of future data into past predictions.

The dataset was chronologically divided into K folds. For each iteration $k \in \{1, \ldots, K-1\}$. , the model was trained on folds 1 through K and evaluated on the subsequent fold, $K+1$. This expanding window strategy simulates a realistic scenario where the model is periodically retrained on accumulated historical data to predict future events.

Within each cross-validation split, the training data was further partitioned for hyperparameter tuning. Specifically, the first 85% of the training window was used for model training while the remaining 15% was used for validation to mitigate overfitting. Continuous sensor values (e.g., temperature) were standardised using Z-score normalisation.

3.2.4 Adjacency Matrix construction

Since the STGNN models require adjacency matrices to capture spatial dependencies of the smart-home sensor network, a static, undirected graph was constructed for each dataset. The graph is represented by an adjacency matrix: $A \in R^{N \times N}$ where N denotes the total number of sensors. An entry A_{ij} was set to 1 if sensors i and j were located in the same room (based on the dataset floorplan), and 0 otherwise. This approach assumes that sensors sharing a physical space exhibit stronger spatial relationships.

3.3 Baseline CNN, LSTM and GWN Implementations

To provide a fair benchmark for evaluating the performance of the GRUGCN model, we implemented and trained three baseline models: a Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) network, and a GraphWaveNet (GWN) model. The CNN and LSTM models were chosen as they represent architectures from prior deep learning research in the human activity recognition and prediction domain. The GWN model was selected as it offers the closest STGNN comparison to GRUGCN. For each baseline, hyperparameter tuning was conducted separately on each dataset to ensure optimal performance. The final hyperparameters for these models can be found in tables 2-5 in the appendix.

3.4 GRUGCN Model Implementation

3.4.1 Hyperparameter optimisation (HPO)

Hyperparameters are parameters that are set prior to training and influence model learning and generalisation performance [8]. To optimise these parameters, we employed Optuna, a python framework for hyperparameter optimisation. The objective function was designed to maximise the weighted F-1 score on a held-out validation set.

Each trial was trained for up to 300 epochs with early stopping applied if the validation F-1 score failed to improve for a predefined number of epochs.

Due to our forward-chaining cross validation strategy, hyperparameters were optimised independently for each fold. After the best parameters were identified, the model was retrained on the full training data of the fold before being evaluated on the unseen test data.

Separate sets of optimal hyperparameters were obtained for the Aruba and Milan datasets due to their inherent differences. A full list of hyperparameters, their search spaces and corresponding optimal values is provided in tables 2-5 in the appendix.

3.4.2 Model training and Evaluation

Following HPO, the best performing configurations were selected for final training and evaluation. For the STGNN models (GRUGCN and GWN), a fixed adjacency matrix was initialised as previously described. Separate models were trained for both the Aruba and Milan datasets to account for the unique properties of each environment.

A three-fold forward-chaining cross-validation was applied, ensuring models were always tested on chronologically later data. For each fold, models were trained on the respective training set, after which predictions were generated on the unseen test set.

Predictions, ground-truth labels, and timestamps were saved in CSV format for subsequent analysis.
We evaluated our models using the weighted F-1 score, which accounts for class imbalance by weighting each class contribution by its prevalence in the dataset. Final performance scores were reported as the mean F-1 score across all three folds.

3.4.3 Adaptive GRUGCN

As an extension to our work, we investigated adaptive learning mechanisms for the GRUGCN model architecture to evaluate whether performance could be improved beyond the fixed adjacency matrix. Specifically, we explored two approaches for implementing a dynamic adjacency mechanism:

a) Node Embeddings

Inspired by GraphWaveNet, this method introduced two learnable embedding matrices, with each sensor assigned a unique vector in both matrices. During a forward pass, the dot product between embeddings generated an $N \times N$ relationship score matrix across all sensor pairs. These scores were passed through ReLU and Softmax functions to produce a normalized, learnable adjacency matrix representing latent functional relationships. This matrix was combined with the static adjacency matrix to form a hybrid graph that captured both physical proximity and data-driven dependencies, subsequently used in the GCN layers for message passing.

b) Attention-Based Mechanism

In this implementation, we extended it with an attention based adaptive adjacency mechanism inspired by Graph Attention Networks (GAT) [7] and the adaptive adjacency approaches in GraphWaveNet [9] and AGCRN [1]. The model learns a data-driven graph structure during training. To achieve this, each sensor's hidden state (produced by the GRU, which encodes its recent temporal dynamics) is projected into two different embedding spaces: a query space and a key space. At each forward pass, the model computes similarity scores between every pair of sensors by taking the dot product between queries and keys. This results in an (N, N) score matrix that reflects the contextual influence of one sensor on another for the current activity window. These scores are then normalized using a SoftMax function, producing a dynamic adjacency matrix whose edge weights vary depending on the activity context. The dynamic adjacency matrix is then combined with the original fixed adjacency matrix. The amount to which they are combined are determined by the 'mixed_static' hyperparameter that is tuned for and can be found in the hyperparameter tables 2-5 in the appendix.

3.5 Infrastructure

All experiments, including training and hyperparameter optimisation for all baseline and STGNN models were conducted using the facilities provided by the University of Cape Town's ICTS High Performance Computing team [5].

4 RESULTS & ANALYSIS

This section presents our results of the comparative performance of our three baseline models against the GRUGCN model. The evaluation is based on the average F-1 score and the F-1 score from fold 3 for the Milan and Aruba datasets. We then present a comprehensive analysis of the GRUGCN model performance for both the Milan and Aruba datasets. The evaluation is based on confusion matrices generated from the model's prediction test set. This analysis is structured into three parts: an overview of the model's aggregate performance, a per class examination of performance metrics and finally a qualitative analysis of the model's misclassification patterns.
We start with human activity recognition and then prediction.

4.1 Human Activity Recognition Results

4.1.1 Results

For the Milan dataset, The GRUGCN model outperforms the other three baselines, albeit marginally. With an average F-1 score of 0.5533 as seen in table 6.
For the Aruba dataset, The LSTM model performed the best with an average F-1 score of 0.82. The CNN model also performs strongly with an F-1 score of 0.80 however the GRUGCN model (0.7478) which were competitive on the Milan dataset was comparatively lower than the non-graph-based baselines.

There is a significant difference in the overall performance between the two datasets. This is due to the Milan dataset being sparser as it only lasts for two months compared to the Aruba dataset that is taken over eight months. The Milan dataset also has a dog that could have created additional noise in the dataset.

The GRUGCN model only outperformed the baselines on the Milan dataset but could not do this for the Aruba dataset. This could be due to the activities being more strongly defined by their temporal sequence than by the complex spatial relationships and thus, the LSTM model performed the best in this case. The GRUGCN model still fell short compared to GWN and this is likely due to not having an adaptive learning mechanism as GWN does.

4.1.2 Confusion Matrices analysis

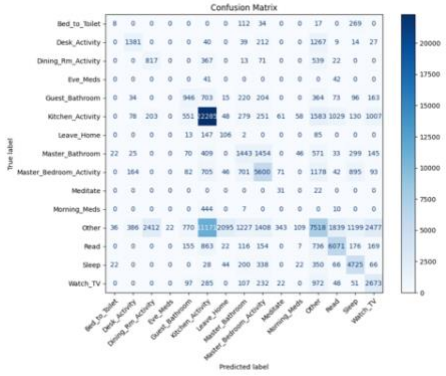

Figure 1: Confusion matrix for milan (HAR)

Table 6: Human Activity Recognition (HAR) results for the Milan and Aruba Datasets

Dataset	Milan	
Metric	Average F-1	Fold 3 F-1
CNN	0.5 (± 0.0453)	0.5436
LSTM	0.53 (±0.0398)	0.5115
GWN	0.53 (±0.0288)	0.5142
GRUGCN	0.5533 (±0.0475)	0.5590
GRUGCN(Node Embed.)	0.4035 (±0.0923)	0.5599
GRUGCN(Attention)	**0.5637 (**±0.0412)	**0.5661**

Dataset	Aruba	
Metric	Average F-1	Fold 3 F-1
CNN	0.8 (±0.0062)	0.81
LSTM	**0.82 (±0.0050)**	**0.82**
GWN	0.77 (±0.0021)	0.77
GRUGCN	0.7478 (±0.013)	0.7419
GRUGCN(Node Embed.)	0.7625 (±0.01)	0.7688
GRUGCN(Attention)	0.7687 (±0.013)	0.7938

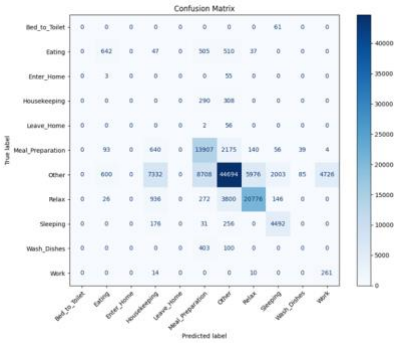

Figure 2: Confusion Matrix for Aruba (HAR)

a) Aggregate performance

GRUGCN achieves an overall accuracy of 52.29% and 67.61% for Milan and Aruba respectively. This was calculated by dividing the sum of all correct predictions by the total number of predictions in Figure 1 and Figure 2 respectively.

b) Per class performance

The respective classification reports can be found in the appendix in table 8 and 9.

For the Milan dataset, The GRUGCN models best identified classes are sleep, kitchen activity and read with F-1 scores of 0.689, 0.685, and 0.684 respectively. These are all typically long duration activities that are mostly stationary with distinct features. The model finds it easier to identify these types of activities. Conversely, the model performs extremely poorly on short, transitional activities. The activities: 'Morning_meds' and 'Eve_meds' were never identified correctly and thus have an F-1 score of 0.00. Similarly, 'Bed_to_Toilet' has a near zero F-1 score of 0.030, further indicating that the model's biggest challenge is transition activities.

For the Aruba dataset, a similar trend can be observed as the model performs its best with high frequency, long-duration activities like 'relaxing', 'sleeping', and 'meal_preparation' with F-1 scores of 0.786, 0.767, and 0.676 respectively. Once again, activities 'Bed_to_Toilet', 'Enter_Home', and 'Wash_Dishes' indicate the model's critical failure to identify short, transitional activities in the dataset. These activities were never once correctly identified.

c) Misclassification report

Looking at the off-diagonal values reveals notable patterns in the model's errors. In the Milan dataset, the model has a strong tendency to misclassify a wide range of activities as 'Kitchen_Activity' or 'Other'. Other was incorrectly predicted 11171 times for the true instance of 'Kitchen_Activity'. This could be due to the sheer frequency of 'Other'. This could in turn cause 'Other' to act as a catch-all for ambiguous kitchen-related movement.

A similar case can be seen in the Aruba dataset. The model seems to struggle to distinguish it with other high frequency activities as a staggering 8708 instances of 'Other' is misclassified for 'Meal_preparation'. There is also a lot of confusion between activities in similar contexts. In the Milan dataset, 'Master_Bedroom_Activity' is confused with 'Master_Bathroom' (1454 predictions) and vice-versa (1227 predictions).

This indicates that the model struggles to differentiate between more specific activities that are in the same spatial proximity. Likewise, in the Aruba dataset, 'Relax' is often mislabeled as 'Other' (3800 instances) suggesting that stationary activities are difficult to distinguish from periods of general inactivity.

4.2 Human Activity Prediction Results

4.2.1 Results

For the Milan dataset, as seen in table 7, The GRUGCN model's F-1 score of 0.3187 was the lowest among the evaluated models lagging behind GWN (0.41), LSTM (0.40) and CNN (0.39). However, it achieved the highest F-1 score on fold 3 suggesting that its ability to generalise across all data folds was weaker than the other models.

On the Aruba dataset, the GRUGCN model performance was consistently lower than the baselines with an average F-1 score of 0.4714. Its fold 3 F-1 score of 0.4901 outperformed GWN (0.43),

Table 7: Human Activity Prediction (HAP) results for the Milan and Aruba Datasets

Dataset	Milan	
Metric	Average F-1	Fold 3 F-1
CNN	0.39 (±0.05)	0.32
LSTM	0.40 (±0.05)	0.32
GWN	**0.41 (±0.05)**	0.34
GRUGCN	0.3187 (±0.0961)	0.3734
GRUGCN(Node Embed.)	0.2926 (±0.1187)	**0.3770**
GRUGCN(Attention)	0.4036 (±0.033)	0.3457

Dataset	Aruba	
Metric	Average F-1	Fold 3 F-1
CNN	**0.55 (±0.01)**	0.52
LSTM	**0.55 (±0.02)**	**0.55**
GWN	0.51 (±0.04)	0.43
GRUGCN	0.4714 (±0.0239)	0.4901
GRUGCN(Node Embed.)	0.4783 (±0.0256)	0.5033
GRUGCN(Attention)	0.4900 (± 0.01)	0.5003

the other graph-based model but it was surpassed by the LSTM (0.55) and CNN (0.52) models.

Once again, both purely temporal and convolutional models outperform both graph-based models in the Aruba dataset. This suggests that the architectural complexity of these graph-based models, designed to capture spatial relationships, did not provide a predictive advantage on the Aruba dataset.

The GRUGCN model appears to be highly sensitive to the underlying data. It can outperform the baselines for fold 3 in the Milan dataset yet underperforms on average. This suggests the model can only excel when clear, complex spatio-temporal patterns are available to learn. Otherwise, its complexity may be a disadvantage compared to simpler models like the LSTM and CNN models in the Aruba dataset.

The GRUGCN model's performance in HAP contrasts with its results for HAR where it was the top performer on the Milan dataset. The inherent uncertainty of forecasting future actions appears to diminish the GRU-GCN's architectural advantage, making it harder for the model to leverage spatial information effectively unless the predictive signals are exceptionally strong and clear.

4.2.2 Confusion Matrices Analysis

a) *Aggregate performance*

The GRUGCN model achieves an accuracy of 39.6% and 48.8% for the Milan and Aruba datasets respectively in figure 3 and 4. This is comparably lower than HAR however expected due to the increased complexity of HAP.

b) Per class performance

Full classification reports can be found in tables 10 and 11 in the appendix attached.

c) *Misclassification report*

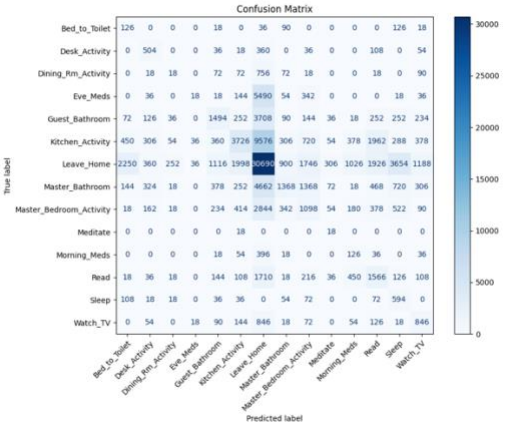

Figure 3: Confusion Matrix for Milan (HAP)

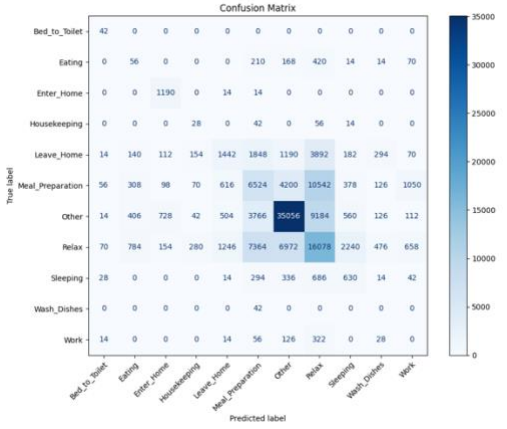

Figure 4: Confusion matrix for Aruba (HAP)

In the Milan dataset, the model's predictions are dominated by a single activity: 'Leave_Home'. This class acts as a "prediction sink" as the model predicts it as the future activity for a vast number of unrelated current states. For example, the model predicts that 9576 instances of 'Kitchen_Activity' and 3708 instances of 'Guest_Bathroom' were incorrectly predicted to be followed by

'Leave_Home'. This suggests the model has over learned a pattern where many activities precede leaving. This causes the model to default to this prediction instead of identifying more nuanced transitions. The precision for nearly all other classes is consequently extremely low.

In the Aruba dataset, 'Other' serves as the catch all future state. A notable 12,474 true instances of 'Meal_Preparation' and 7,344 true instances of 'Relax' were incorrectly predicted to be followed by 'Other'. The model seems to tend to opt for a safe generation prediction of 'Other'.

In both datasets, the model developed a strong bias towards predicting a single, high frequency future state. i.e. 'Leave_Home' and 'Other' in the Milan and Aruba datasets respectively. This indicates that the model is not learning the complex, nuanced transitions between activities but is rather defaulting to the most statistically probably outcome.

The model is particularly good at predicting when a long duration activity like 'Relax' in the Aruba dataset will continue but struggles with a change in state which is the core challenge of activity prediction.

There is a significant drop in accuracy and F-1 scores in prediction compared to the task of recognition. This demonstrates the substantially higher difficulty of prediction. This uncertainty of future actions leads the model to make safe, overly generalised forecasts, however recognition benefits from the immediate, available data of the current state.

4.3 Adaptive GRUGCN Results

As part of our extension, this section presents a comparative analysis of the base model GRUGCN compared to two enhanced variants: one incorporating node embeddings and another using an attention mechanism.

4.3.1 Adaptive GRUGCN for HAR

For the Milan dataset the attention based GRUGCN provided the best results with an average F-1 score of 0.5637. Conversely, the inclusion of node embeddings proved to be detrimental to the model's overall performance with an average F-1 score of 0.4035. However, its fold 3 F-1 score is comparable with a value of 0.5599. This suggests that while the node embeddings could be tuned for a specific data partition, they severely hindered the model's ability to generalize across the entire dataset.

For the Aruba dataset, the attention-based model once again yielded improved results over both the standard model with an average F-1 score of 0.7687. Unlike on the Milan dataset, the node embeddings variant also provided an improvement. This indicates that node embeddings were more beneficial in the context of the Aruba dataset.

The performance of node embeddings appears to be dataset-dependent, yielding better results on the Aruba dataset compared to the Milan dataset. This discrepancy may stem from the inherent differences between the two datasets; specifically, the Milan dataset is sparser and noisier, while the Aruba dataset contains a larger volume of data.

4.3.2 Adaptive GRUGCN for HAP

For the Milan dataset, the attention-based variant demonstrated superior performance over the traditional GRUGCN model with an average F-1 score of 0.4036. This variant also displayed the greatest stability with a low standard error of (±0.033). Once again, the inclusion of node embeddings was detrimental to the model's overall performance with the lowest average F-1 score 0.2926 and the highest variance (±0.1187). As seen in previously in HAR, the node embeddings variant's fold 3 F-1 score achieved the highest F-1 score of 0.3770 for HAP as well.

On the Aruba dataset, the attention based GRUGCN model once again yielded improved results, outperforming the standard model with an F-1 score of 0.49. In contrast to its performance on the Milan dataset, the node embeddings variant, also provided a slight improvement over the baseline GRUGCN. This finding indicates that the learned node embeddings were more beneficial and effective within the context of the Aruba dataset.

4.3.3 Adaptive GRUGCN heatmap analysis for HAR

To further understand how the models learn spatial dependencies, this subsection analyzes and compares the adjacency matrices of the best-performing variant, the attention-based GRUGCN, with the GraphWaveNet baseline.

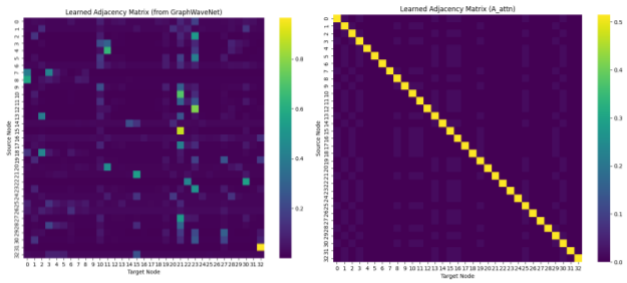

Figure 5: Heatmaps for GWN vs GRUGCN (attention) Milan

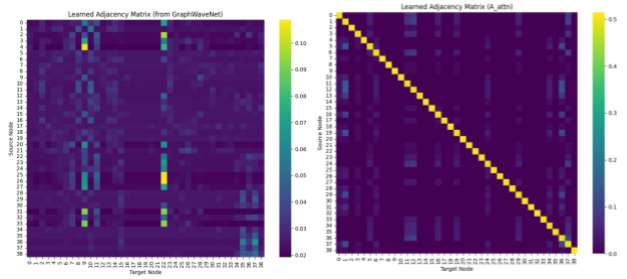

Figure 6: Heatmaps for GWN vs GRUGCN (attention) Aruba

The heatmaps for the attention based GRUGCN model displays an unambiguous pattern, a bright diagonal. The off-diagonal values

are almost uniformly dark, except for some slightly lighter areas. Overall, this indicates that the model has chosen a near-zero influence between different nodes and values its own past information as the most critical predictor for its future state.

The GWN heatmaps are characterized by a sparse, scattered pattern of influence. The diagonals are not consistently strong, indicating that self-influence is not the model's primary focus. Instead, a few bright spots off the diagonal line indicate the model has learned a few specific, valuable relationships between distant nodes.

The two models exhibit contrasting strategies. In Milan, GRUGCN achieved a higher F-1 score than GraphWaveNet, suggesting its approach was more effective in that context. However, on Aruba, GRUGCN performed worse, indicating that GraphWaveNet's strategy may be better suited for that dataset.

5 DISCUSSION & CONCLUSIONS

This study investigated the effectiveness of the GRUGCN model for Human Activity Recognition (HAR) and Human Activity Prediction (HAP) in smart home environments, compared with CNN, LSTM, and GraphWaveNet baselines.

The results show that GRUGCN achieved marginal improvements on the Milan dataset for HAR, particularly on longer-duration activities, but failed to consistently outperform simpler temporal models like the LSTM baseline. For the Aruba dataset, the CNN and LSTM models performed more reliably. This shows that the GRUGCN model is more sensitive to dataset properties like sensor density and noise. For HAP, the GRUGCN model lagged behind both the CNN and LSTM models on average and displayed a strong tendency to default to majority-class predictions.

Our evaluation showed that the incorporation of adaptive mechanisms, particularly the attention-based variant, consistently improved performance across both datasets for HAR and HAP. The node embeddings approach had mixed success as it improved results on Aruba but were detrimental for Milan. These improvements confirm static graph structures like the traditional GRUGCN model limit its effectiveness and adaptive learning can better capture dynamic relationships between sensors.

This study was also limited to two CASAS datasets which were both single resident environments. Broader evaluation and larger multi-resident datasets would be required to validate the generalisability of our findings.

Overall, our work demonstrates that while GRUGCN shows some promise for HAR and HAP, it does not consistently outperform simpler temporal baselines. Models that integrate attention or other adaptive mechanisms appear to be better suited to capture the complexity of real-world human activity patterns, particularly as prediction tasks remain an open challenge in this domain.

REFERENCES

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive Graph Convolutional Recurrent Network forTraffic Forecasting. Retrieved from http://arxiv.org/abs/2007.02842

[2] Kaixuan Chen, Lina Yao, Dalin Zhang, Bin Guo, and Zhiwen Yu. 2019. Multi-agent Attentional Activity Recognition. Retrieved from http://arxiv.org/abs/1905.08948

[3] Jianfei Gao and Bruno Ribeiro. 2023. On the Equivalence Between Temporal and Static Graph Representations for Observational Predictions. Retrieved from http://arxiv.org/abs/2103.07016

[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Computation 9, 8: 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[5] ICTS High Performance Computing team. 2023. High Performance Computing at University of Cape Town. https://hpc.uct.ac.za.

[6] Mohammed Elnazer Abazar Elmamoon and Ahmad Abubakar Mustapha. 2025. A Comparative Study of Deep Learning Models for Human Activity Recognition. Cloud Computing and Data Science: 79–93. https://doi.org/10.37256/ccds.6120256264

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. Retrieved from http://arxiv.org/abs/1710.10903

[8] Jia Wu, Xiu Yun Chen, Hao Zhang, Li Dong Xiong, Hang Lei, and Si Hao Deng. 2019. Hyperparameter optimization for machine learning models based on Bayesian optimization. Journal of Electronic Science and Technology 17, 1: 26–40. https://doi.org/10.11989/JEST.1674-862X.80904120

[9] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. Retrieved from http://arxiv.org/abs/1906.00121

[10] Washington State University. 2025. CASAS Smart Home Project Datasets. https://casas.wsu.edu/datasets/.

[11] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. Retrieved from http://arxiv.org/abs/1801.07455

[12] Assaad Zeghina, Aurélie Leborgne, Florence Le Ber, and Antoine Vacavant. 2024. Deep learning on spatiotemporal graphs: A systematic review, methodological landscape, and research opportunities. Neurocomputing 594. https://doi.org/10.1016/j.neucom.2024.127861

APPENDICES

Table 2: Hyperparameters used for CNN, LSTM, GWN and GRUGCN models for the Milan dataset HAR

Hyperparameters	Search Space	Optimal Hyperparameters					
		CNN	LSTM	GWN	GRUGCN (Normal)	GRUGCN (node-embeddings)	GRUGCN (attention)
Window size	[10-100]	40	40	10	80	20	50
Stride percentage	[0.2-1]	0.3059218305954031	0.9286875162060546	0.3658365458890759	0.2849323498829033 4	0.2403928338342755 7	0.2186589468972951 2
Batch size	[32,64,128]	128	128	32	128	32	32
Learning rate	[0.0001 - 0.01]	0.0003138804586149 8274	0.0003274152600693986	0.0012253920568570877	0.0022662045646897 79	0.0001651065923369 3272	0.0020299532833769 54
Weight decay	[0.000001-0.001]	4.9819618 93920149e -05	9.6450185 48540572e -05	1.9947677 571734607 e-06	5.9335331 45764349e -05	2.6418278 046213356 e-05	6.3517899 68828102e -06
Early stopping patience	[15-25]	16	19	22	24	25	20
Filters1	[32,64,128]	128	-	-	-	-	-
Filters2	[64,128,256]	256	-	-	-	-	-
Kernel_size	[3,5,7]	5	-	-	-	-	-
Activation_fn	['relu','leaky_relu']	leaky_relu	-	-	-	-	-
Hidden size	[32-256]	-	96	32	32	64	32
Ff_size	[128, 256, 512]	-	-	256	-	-	-
N_layers	[1-5]	-	-	4	3	2	1
Emb_size	[10,20,40]	-	-	-	-	10	-
topk	[8, 10, 12, 15]	-	-	-	-	-	8
Mixed_static	[0.0-1]	-	-	-	-	-	0.3237292 34321161

Table 3: Hyperparameters used for CNN, LSTM, GWN and GRUGCN models for the Aruba dataset HAR

Hyperparameters	Search Space	Optimal Hyperparameters					
		CNN	LSTM	GWN	GRUGCN (Normal)	GRUGCN (node-embeddings)	GRUGCN (attention)
Window size	[10-100]	40	70	10	60	80	100
Stride percentage	[0.2-1]	0.36 970941286 996223	0.3345815 106342040 5	0.4695924 355422541 6	0.5264680 546107309	0.2001067 352776531 3	0.2182431 033758911 5
Batch size	[32,64,128]	64	32	128	32	32	32
Learning rate	[0.0001 - 0.01]	0.0001437 129365952 2506	0.0001340 214332898 955	0.0001011 265628346 5014	0.0006837 902413850 405	0.0013197 971663913 434	0.0010852 790199596 197

Weight decay	[0.000001-0.001]	0.0001824921892277345	3.581844936755959 7e-05	9.200066355306309e-06	7.874297835108202e-05	6.028453972917 6e-06	2.748790907287318 2e-06
Early stopping patience	[15-25]	21	21	16	18	22	18
Filters1	[32,64,128]	128	-	-	-	-	-
Filters2	[64,128,256]	128	-	-	-	-	-
Kernel_size	[3,5,7]	5	-	-	-	-	-
Activation_fn	[‘relu’,’leaky_relu’]	Leaky_relu	-	-	-	-	-
Hidden size	[32-256]	-	96	32	64	32	32
Ff_size	[128, 256, 512]	-	-	128	-	-	-
N_layers	[1-5]	-	-	5	2	1	2
Emb_size	[10,20,40]	-	-	-	-	10	-
topk	[8, 10, 12, 15]	-	-	-	-	-	8
Mixed_static	[0.0-1]	-	-	-	-	-	0.4660466815936153 4

Table 4: Hyperparameters used for CNN, LSTM, GWN and GRUGCN models for the Milan dataset HAP

Hyperparameters	Search Space	Optimal Hyperparameters					
		CNN	LSTM	GWN	GRUGCN (Normal)	GRUGCN (node-embeddings)	GRUGCN (attention)
Window size	[10-100]	20	10	20	90	30	10
Stride percentage	[0.2-1]	0.283446051271579	0.4590650099900955	0.20140742573801032	0.206917117729562	0.2501452719153233	0.897886858474856
Batch size	[32,64,128]	32	32	32	32	32	32
Learning rate	[0.0001 - 0.01]	0.0001793417080389 5634	0.0001617065680088524	0.0012253920568570877	0.0008311525621073043	0.0002240161796697806	0.0001692638611732 3042
Weight decay	[0.000001-0.001]	0.0003910203221204539	1.594281502960981 6e-05	2.049819262701 43e-06	2.526759849739073e-05	4.748315977045662e-06	0.0002002420820809 7415
Early stopping patience	[15-25]	24	24	15	22	22	18
Filters1	[32,64,128]	128	-	-	-	-	64
Filters2	[64,128,256]	256	-	-	-	-	-
Kernel_size	[3,5,7]	7	-	-	-	-	-
Activation_fn	[‘relu’,’leaky_relu’]	relu	-	-	-	-	-
Hidden size	[32-256]	-	128	48	32	64	64
Ff_size	[128, 256, 512]	-	-	256	-	-	-
N_layers	[1-5]	-	-	5	2	1	4
Emb_size	[10,20,40]	-	-	-	-	30	-
topk	[8, 10, 12, 15]	-	-	-	-	-	12
Mixed_static	[0.0-1]	-	-	-	-	-	0.9494695

							462953908

Table 5: Hyperparameters used for CNN, LSTM, GWN and GRUGCN models for the Aruba dataset HAP

Hyperparameters	Search Space	Optimal Hyperparameters					
		CNN	LSTM	GWN	GRUGCN (Normal)	GRUGCN (node-embeddings)	GRUGCN (attention)
Window size	[10-100]	70	70	10	20	90	10
Stride percentage	[0.2-1]	0.38858146749800293	0.20066544517157567	0.45959410479978080	0.73864263395638790	0.39314143711139570	0.63581562140864520
Batch size	[32,64,128]	128	32	32	64	32	128
Learning rate	[0.0001 - 0.01]	0.00048763199134896050	0.00032741526006939860	0.00012174128947549716	0.00073586215750774240	0.00324319481034849100	0.00056104277913555800
Weight decay	[0.000001-0.001]	4.160014021904976e-05	0.000894823052777054100	0.000970412468749749100	5.171391680822943e-06	5.362867922645050e-06	5.993721738557602e-06
Early stopping patience	[15-25]	20	19	20	23	21	20
Filters1	[32,64,128]	64	-	-	-	-	-
Filters2	[64,128,256]	256	-	-	-	-	-
Kernel_size	[3,5,7]	5	-	-	-	-	-
Activation_fn	[‘relu’,’leaky_relu’]	relu	-	-	-	-	-
Hidden size	[32-256]	-	160	64	32	32	32
Ff_size	[128, 256, 512]	-	-	128	-	-	-
N_layers	[1-5]	-	-	5	1	1	1
Emb_size	[10,20,40]	-	-	-	-	20	-
topk	[8, 10, 12, 15]	-	-	-	-	-	12
Mixed_static	[0.0-1]	-	-	-	-	-	0.90074454286165030

Table 8: HAR Classification report for Milan

Activity	Precision	Recall	F1-Score	Support
Bed_to_Toilet	0.091	0.018	0.030	440
Desk_Activity	0.668	0.462	0.546	2989
Dining_Rm_Activity	0.238	0.447	0.311	1829
Eve_Meds	0.000	0.000	0.000	83
Guest_Bathroom	0.352	0.336	0.344	2818
Kitchen_Activity	0.594	0.809	0.685	27563
Leave_Home	0.045	0.300	0.078	353
Master_Bathroom	0.323	0.319	0.321	4517
Master_Bedroom_Activity	0.562	0.585	0.573	9577
Meditate	0.059	0.585	0.107	53

Morning_Meds	0.000	0.000	0.000	461
Other	0.495	0.228	0.312	33012
Read	0.654	0.717	0.684	8469
Sleep	0.602	0.806	0.689	5861
Watch_TV	0.392	0.596	0.473	4487
Accuracy	-	-	0.523	102512
Macro Avg	0.338	0.414	0.344	102512
Weighted Avg	0.525	0.523	0.500	102512

Table 9: HAR Classification report for Aruba

Activity	Precision	Recall	F1-Score	Support
Bed_to_Toilet	0.000	0.000	0.000	61
Eating	0.471	0.369	0.414	1741
Enter_Home	0.000	0.000	0.000	58
Housekeeping	0.000	0.000	0.000	598
Leave_Home	0.000	0.000	0.000	58
Meal_Preparation	0.577	0.815	0.676	17054
Other	0.860	0.603	0.709	74124
Relax	0.771	0.800	0.786	25956
Sleeping	0.665	0.907	0.767	4955
Wash_Dishes	0.000	0.000	0.000	503
Work	0.052	0.916	0.099	285
Accuracy	-	-	0.676	125393
Macro Avg	0.309	0.401	0.314	125393
Weighted Avg	0.780	0.676	0.710	125393

Table 10: HAP Classification report for Milan

Activity	Precision	Recall	F1-Score	Support
Bed_to_Toilet	0.040	0.304	0.070	414
Desk_Activity	0.259	0.452	0.329	1116
Dining_Rm_Activity	0.042	0.016	0.023	1134
Eve_Meds	0.167	0.003	0.006	6156
Guest_Bathroom	0.372	0.223	0.279	6714
Kitchen_Activity	0.515	0.200	0.289	18594
Leave_Home	0.503	0.647	0.566	47448
Master_Bathroom	0.411	0.135	0.204	10098
Master_Bedroom_Activity	0.188	0.173	0.180	6354
Meditate	0.031	0.500	0.059	36
Morning_Meds	0.056	0.184	0.086	684
Read	0.227	0.344	0.273	4554
Sleep	0.094	0.589	0.162	1008
Watch_TV	0.250	0.370	0.298	2286
Accuracy	-	-	0.396	106596
Macro Avg	0.225	0.296	0.202	106596
Weighted Avg	0.416	0.396	0.374	106596

Table 11: HAP Classification report for Aruba

Activity	Precision	Recall	F1-Score	Support

Bed_to_Toilet	0.176	1.000	0.300	42
Eating	0.033	0.059	0.042	952
Enter_Home	0.521	0.977	0.680	1218
Housekeeping	0.049	0.200	0.078	140
Leave_Home	0.375	0.154	0.219	9338
Meal_Preparation	0.324	0.272	0.296	23968
Other	0.730	0.694	0.711	50498
Relax	0.390	0.443	0.415	36322
Sleeping	0.157	0.308	0.208	2044
Wash_Dishes	0.000	0.000	0.000	42
Work	0.000	0.000	0.000	560
Accuracy	-	-	0.488	125124
Macro Avg	0.250	0.373	0.268	125124
Weighted Avg	0.506	0.488	0.491	125124