Transformers encoders to extract temporal-spatial context of a building

Somaya Ahmadi, Longwood University

Somaya Ahmadi is currently an undergraduate student majoring in computer science at Longwood University. She has experience in web development and machine learning projects.

Dr. Sanish Rai, Longwood University

Dr. Sanish Rai is an Assistant Professor in the Department of Computer Science at Longwood University. He received his Ph.D. in Computer Science from Georgia State University. His teaching and research interests include machine learning, artificial intelligence, simulation and modeling, and computing education. He is actively involved in curriculum development and undergraduate research.

©American Society for Engineering Education, 2026

Abstract

Occupancy estimation in buildings is important for energy saving, safety, evacuation planning and occupant comfort. As energy demands and carbon emissions are rising, smart occupancy estimation is one of the solutions for managing resources. This project focuses on temporal in each room and spatial among rooms occupancy estimation using time series sensor data such as HVAC, lighting, plug load and weather conditions.

We use transformer encoder only. Transformers are used widely in learning long-term dependencies in sequence and their ability to model both temporal and spatial relationships through their self-attention heads that focus on multiple places at the same time. Each room has its temporal encoder to learn time series patterns in each room and a shared spatial encoder that learn dependencies across rooms. The spatial context is important when some sensor data is missing, as the model can rely on information from related rooms then make accurate predictions. Transformers are highly parallelizable and efficient with GPU`s. Our model achieved 92% accuracy demonstrating good performance.

I. Introduction

Building occupancy estimation is predicting how many people are in a room or a building at a specific time [2,7]. It is useful for improving safety, security, energy efficiency, evacuation planning, and occupant comfort [3,4]. This project focuses on spatial occupancy estimation using machine learning, aiming to predict how many people are present in several rooms at the same time and shows their temporal and spatial relationships using environmental sensor data such as HVAC, plug loads, fans, light, and weather conditions to find patterns without using invasive technologies to avoid privacy concerns.

The model used in this project is based on transformer neural networks which are good at analyzing time-series data and learning complex patterns in multiple spaces [5]. We implemented encoder-only transformer models. Temporal transformer encoders applied to each room separately to learn time-based patterns and a shared Spatial Transformer Encoder to get dependencies between the rooms.

Advancements in machine learning and the increase in use of sensors have made it possible to collect data from buildings. The Improvements in Internet of Things (IoT) enable us to monitor indoor environments and analyze sensor data in various ways [3]. The Building Internet of Things (BIoT) has been very useful in reducing energy consumption and costs. Building occupancy aids in the improvement of the energy management systems, allowing the reduction of energy consumption while maintaining occupant comfort [3].

Globally growing population, global warming and rising living standards caused the world to demand energy. Now the focus is on discovering methods to use less energy and maintain quality

of life. Research shows that buildings are responsible for approximately 40% of the world`s energy consumption and releases 36% of total carbon emissions [5, 6].

Unlike other traditional machine learning algorithms like RNN, LSTM, Decision Tree, and MLP, transformers can better handle long sequences and learn patterns both temporal and spatial using their self-attention head to predict the number of occupants and learn the spatial patterns [5]. At the same time transformers are highly parallelizable and thus efficient to run in GPU [5].

In this project we trained a transformer model using a dataset containing environmental sensor records for five rooms. Our goal is to develop an occupancy estimation framework that predicts the number of people and deploys it on dedicated hardware, such as the Raspberry Pi. Using transformer encoders allows us to extract the temporal dependency among the sensor data and the spatial dependency between the rooms. Thus, the model is temporal and spatial context aware and much better at estimation than other models. Our experiment results show transformers can achieve 92% accuracy.

The rest of the paper is organized into various sections as follows. Section III presents the related work, and section IV presents the information about the dataset and data preprocessing. Section V presents model design, and VI presents the experiments and results. Finally, we conclude in Section VII.

II. Related works

Occupancy estimation has been an interest to researchers mainly for energy saving [4]. It can be widely used to improve HVAC systems and lights to be more efficient. This is used in larger occupancy buildings such as schools, offices, and factories. For example, people have achieved a reduction in energy usages as high as 40% [10]. Environmental sensors have been successful in retrieving accurate occupancy information [4]. With advancements in IoT, occupancy estimation can be applied in homes to save energy, enhance comfort, and improve safety [11].

Researchers use different models for occupancy estimation such as k-nearest-neighbors (K-NN), support vector machines (SVM), random forests (RF), Logistic Regression (LR) [12], and deep neural networks such as Multilayer Perceptron (MLP) [13], Convolutional Neural Networks (CNN) [14], and Recurrent Neural Networks (RNN) [15,16]. More recently, Transformer models have been used for occupancy prediction because of their strong ability to capture patterns in time-series data. One study introduced a model OPTnet, a Transformer-based occupancy prediction network [5]. The researchers collected data from six zones in an office building in Hebei, China, using environmental sensors such as CO_2, temperature, humidity, and fan control status. They compared the Transformer model with other machine learning models like random forest, decision tree, XGBoost, and LSTM. Their results showed that OPTnet outperformed the others, achieving high accuracy. They achieved accuracy of 96.7% in Zone 1 (Week 2, 1-minute resolution) and 94.5% in Zone 4 (Week 1, 30-minute

resolution). The model also had the lowest mean squared error (MSE) in most zones, such as 0.033 in Zone 1 and 0.030 in Zone 7. This demonstrates that Transformers can effectively learn both short- and long-term patterns using self-attention.

Another study tested hybrid models that combine Bidirectional LSTM (Bi-LSTM) and Transformer layers to predict occupancy using low-resolution smart meter data [8]. SERT is a transformer-based model and SST-ANN is an artificial neural network that skips transformers but uses triplet encoding that represents each input as (time, variable, value) let the model learn temporal and spatial dependencies. In SERT, these encoded inputs are passed through transformer encoder and SST-ANN use a simple feed-forward layer. They did experiments that SERT outperformed the other models [9].

By looking at these previous studies, we applied an encoder-only Transformer model for this project to estimate number of occupants. The main goal of this project is to build an accurate, non-intrusive occupancy estimation model that captures both temporal and spatial dependencies across rooms. This allows the model to still make good predictions even if some data is missing.

III. Data Description

A. Dataset Information:

This study uses the ROBOD dataset [1], which contains environmental sensor data collected from five rooms. These rooms include two lecture halls of different sizes, one office space for administrative staff, one office space for researchers, and a library space accessible to all students as shown in Figure 1. A total of 181 days of data was collected at a sampling resolution of every 5 minutes. Room 1, Room 2, and Room 3 each contributed 29 days of data, while Room 4 and Room 5 contributed 47 days.The dataset includes timestamp information stored as strings, occupancy counts, and presence information stored as integers, and all other sensor readings stored as floating-point numbers. The dataset provides ground truth occupancy data, which was collected by manually counting the number of people and using surveillance camera.

B. Dataset Preprocessing:

Before training, the dataset was preprocessed. Missing values were handled using forward and backward filling methods. A total of 23 environmental features were normalized. The common timestamp and features across all the five rooms were selected. The processed data was then divided into input sequences for training, with each sequence containing 60-time steps and an overlap of 59 steps between sequences.

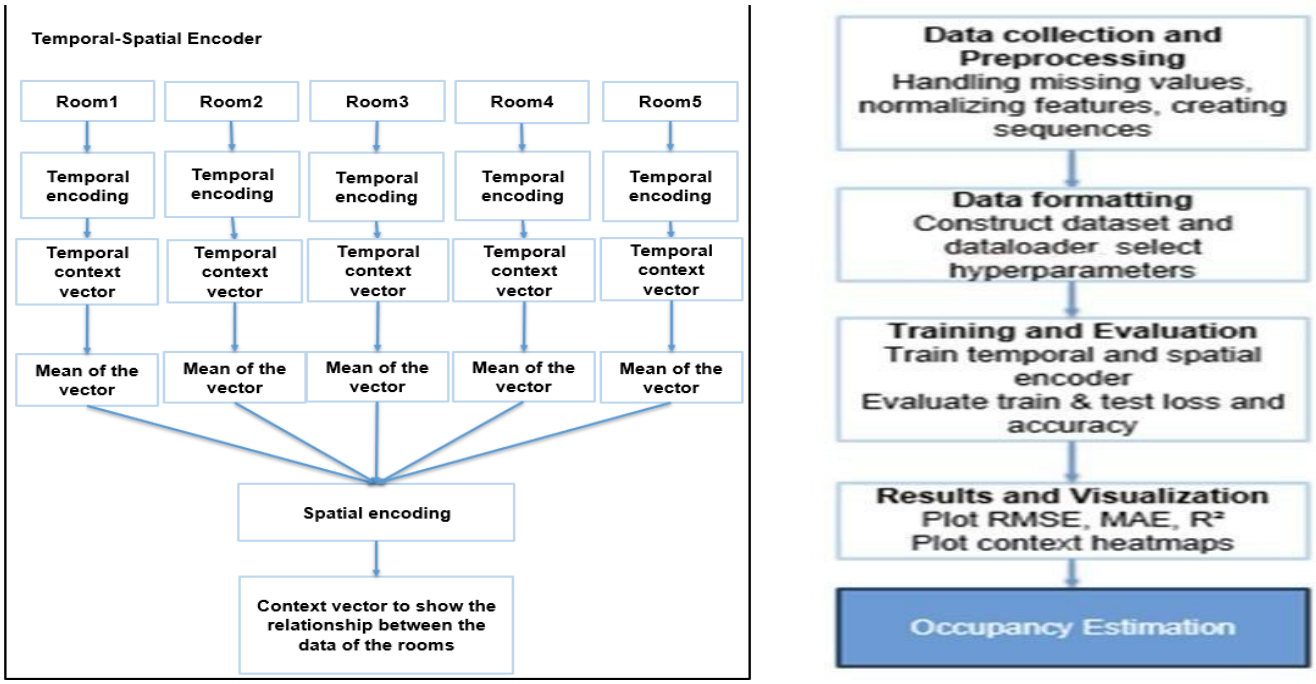

Figure 2. a) Temporal-Spatial Encoder framework b) Model training and evaluation process

IV. Model Design

Framework explanation:

We built a deep learning model to understand time-series patterns and relationships between rooms using transformer layers that contain two parts shown in Figure2.a. This is a hierarchical model that temporal dependencies are captured first using individual temporal encoders per room, and spatial relationships are learned on top of the temporal features using a spatial encoder. This allows richer occupancy prediction by combining both time-series patterns and spatial context among rooms.

The figure2.b shows the model training and evaluation process. The first part of the model is the temporal transformer encoder, which inherits from a PyTorch base class for neural network models. This class has hyperparameters such as hidden dimensions, number of heads and transformer layers. Input dimension is the number of input features, that is, 23 features per timestamp; the hidden dimension is the size of internal hidden states used by a transformer after changing the input into the expected format. A larger hidden dimension enables the model to learn more complex representations; however, it increases memory consumption. The number of attention heads focuses on how many parallel attention mechanisms are used and focuses on different parts of time sequences simultaneously to learn patterns better. The number of layers shows how many transformer-encoded layers are stacked and it enables the model to learn deeper patterns in predicting the number of occupants. This model includes a linear transformation layer for projecting the input features to a suitable format be able to process by transformer. Each time step contains 23 input features, and the internal hidden dimension is set to 64, this layer transforms the input shape from (batch size, sequence length, 23) to (batch size, sequence length, 64). The

intermediate feedforward layer is set to twice the hidden size to give the model more ability to learn complex pattens. This helps improve accuracy without using too much memory or slowing it down too much. The dropout rate of 10 percent means that during the training it avoids 10 percent of neurons from training which is useful for avoiding overfitting. The first batch equals true tells the model that the input format is structured as (batch size, sequence length, number of features). Self-encoder stacks transformer encoder layers and each layer learn temporal patterns. The forward method defines how data flows through the model during execution. First the input features are passed through a linear projection layer, and then the projected data is passed by encoder layers which learn the temporal context of input sequences. Lastly, it outputs temporal occupancy estimation, which now contains richer representations of the time-series input with encoded temporal context.

The second part of the model is implemented in the multi room transformer model class. After each room`s time-series data is processed by its own temporal transformer model stored in the self-room encoders, the model uses temporal heads to predict number of occupants per room. These temporal heads are stored in self-temporal heads, and each room has its own neural network. Each of these networks takes the output sequence from transformer and flattens it into a single vector from (batch size, sequence length, hidden dimension) to (batch size, sequential length*hidden dim). This prepares the data to be fed into a linear layer. The vector passes through two linear layers with a ReLU activation: the first layer reduces the size to 64 neurons, and the second value is occupant count for specific room. ReLU prevents negative values from flowing into the output layer for example if we are counting number of people, it cannot be negative.

The spatial encoder looks at all the rooms together and learns how the rooms affect each other using self-attention mechanism. After learning both temporal and spatial patterns, the model does prediction using forward (). The input x has the shape (batch size, number of rooms, sequential length, input dimensional). Each batch has time series data for the rooms. We initialized two empty lists named room contexts to store summary of vectors for each room and temporal prediction to store individual room predictions. For every room it extracts time series data from all rooms that have the shape (batch size, sequential length, input dimensional). The data is passed through a temporary transformer encoder.

The encoded outputs are pooled using mean pooling to create a vector containing temporal context. These vectors are then combined and passed through spatial encoder that learns dependencies between rooms. The output of spatial encoder is passed through a final layer named spatial output head that produces spatial predictions for each room. These predictions have both each room`s history and its relationship to other rooms. Simultaneously, the model uses temporal heads to predict each room based on its temporal encoder output. In the end, the model returns both temporal and spatial predictions for all the rooms.

We divided the data from each room into training and testing sets. To organize the data, we created a class called multi room dataset using PyTorch. This class collects sensor readings and occupancy numbers from five rooms. Each input has the shape (number of rooms, sequence length, number of features), and the labels are the real number of people in each room. We used PyTorch's data loader to send the data to the model in small batches. The training data was shuffled to help the model learn better. This setup helped the model get the data it needed during training and testing.

To train Transformer models for occupancy prediction in individual rooms, we defined a function called train and plot single room. This function is used for training, evaluating, and visualizing the performance of our model based on time-series sensor data from a single room. From the room sequences, the training and testing inputs X_train, X_test, y_train, y_test. To efficiently train the model in batches, a tensor dataset is created and passed into a PyTorch data loader.

The model used for each room consists of two main components. The first is a custom temporal transformer encoder, which captures temporal patterns across the input sequence using multi-head self-attention and stacked encoder layers. The second component is a prediction head which flattens the encoder's output and feeds it, followed by a ReLU activation that ensures non-linearity and avoids negative outputs. The final linear layer produces a single output, representing the predicted number of occupants for that room. In the training loop, the model parameters are updated using the Adam optimizer to adjust the weights so that it makes better prediction. At the same time, training accuracy is measured by comparing the rounded predictions. After each epoch, the model is evaluated. At the end of training, final predictions on the test set are computed, and the first 100 predictions are saved for visualization. Alongside the predicted values, the actual test labels are also saved. Additionally, three key evaluation metrics are calculated: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). These metrics help measure how accurately the model predicts occupancy and how much variation is between actual and predicted data. The results are printed in a summary text file. After all models are trained, their performance is visualized using three custom plotting functions. The first function called plot temporal loss subplots, generates side-by-side plots comparing training and test loss over epochs for each room. This helps identify overfitting or underfitting. The second function plot called temporal accuracy subplots, shows how well the model performed in predicting the correct number of occupants across time. The third function called temporal actual vs predicted subplots, visualizes the predicted versus actual occupancy for the first 100 test samples for each room, making it easier to see how closely the model tracks the real data.

To train and evaluate the spatial (multi-room) Transformer model, first, we initialized data tracking dictionaries for each room to store the training and testing losses, accuracies, and predictions. Then, we used the train multi room function, which handles both training and evaluation of the full spatial Transformer model that takes all room data together.

Inside this function, we defined the optimizer as Adam and used Mean Squared Error (MSE) as our loss function. For each batch, we passed the input tensor (shaped as batch size × number of rooms × sequence length × feature dimension) to the model, which returned predictions from the spatial head. We recorded training loss and accuracy by comparing rounded predictions to true values. After training, gradient tracking was disabled to reduce memory usage and speed it up. At the end of each epoch, we printed both training and testing metrics to monitor performance. Once training was complete, we extracted the first 100 predictions and ground truths for each room and stored them in dictionaries. We also calculated RMSE (Root Mean Squared Error), MAE

(Mean Absolute Error), and R^2 (coefficient of determination) to evaluate how well the spatial model predicted occupancy across rooms. These metrics were printed and saved in a summary file. We then visualized performance using grouped subplots. The function plot multiroom loss subplots showed the loss for training and testing per room. Similarly, plot multiroom accuracy subplots displayed accuracy, and plot multiroom actual vs predicted subplots compared actual vs. predicted values for the first 100 samples. These plots were saved to the results directory. To understand how rooms dependency, we plotted heatmap dependency using plot room dependency heatmap function to visualize how much rooms affect each other based on learned spatial representations.

V. Experiments

We performed many experiments and used these hyperparameters for both the temporal (per-room) and spatial (multi-room) Transformer models. The model used an input sequence with a length of 60. We split 25% of the data for testing and used 75% for training. The models were trained for 30 epochs.

For the temporal Transformer models, which were trained separately for each room, we used a hidden dimension size of 64, with 4 attention heads and 3 encoder layers. A dropout rate of 0.1 was applied to prevent overfitting. The temporal encoders learned time-based patterns from the input features to predict the number of occupants. The table below presents the performance of the temporal Transformer models trained for each room individually, we used three key metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2).

Rooms	RMSE	MAE	R^2
Room1 (R1)	0.7645	0.2504	0.9793
Room2 (R2)	0.8487	0.3060	0.9396
Room3 (R3)	0.5894	0.2803	0.9439
Room4 (R4)	0.7828	0.5002	0.9482
Room5 (R5)	0.6718	0.3177	0.9392

Overall, the temporal models performed well across all rooms, with R^2 values consistently above 0.93, confirming that the models were effective in learning temporal occupancy patterns.

The plot shows test loss and train loss over epochs. It helps us understand how well the model is learning shown below in Figure3. If both losses decrease steadily and stay close together, it means the model is learning properly without overfitting.

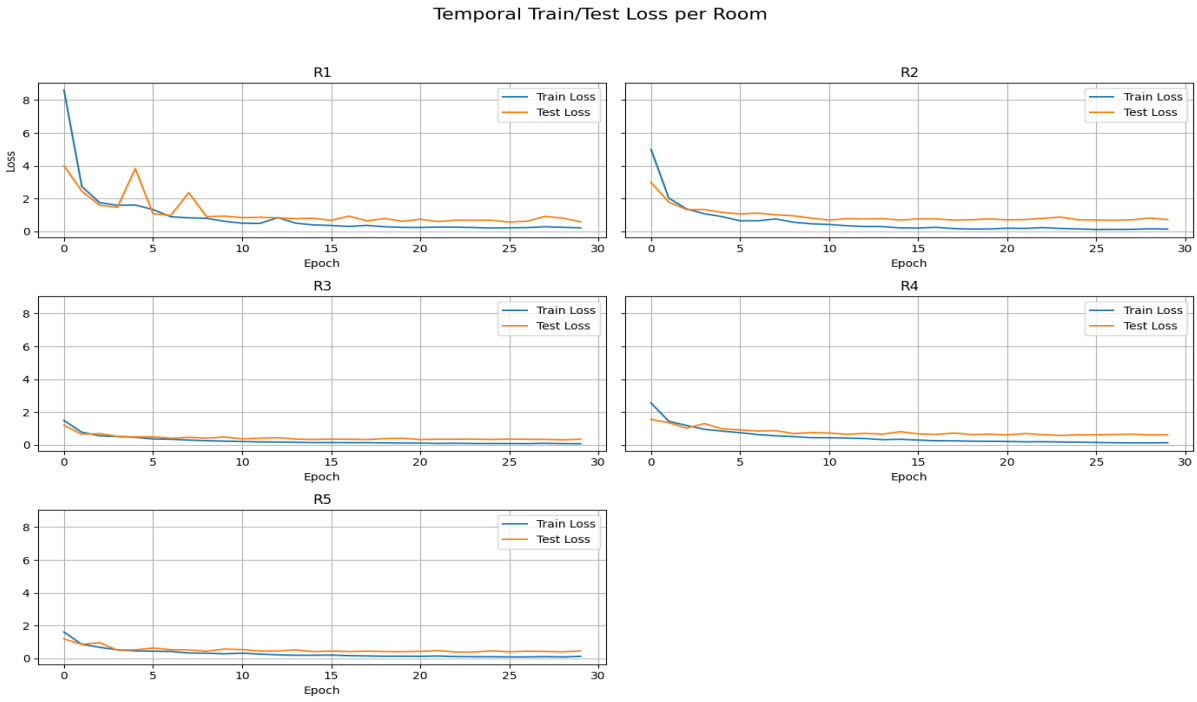

Figure 3. Temporal train vs test loss over epochs

The plot of temporal training and testing accuracy shown in Figure 4 shows how well the model performs in predicting occupancy over time. An increase in test accuracy means the model is learning and the model is improving its ability to predict occupancy.

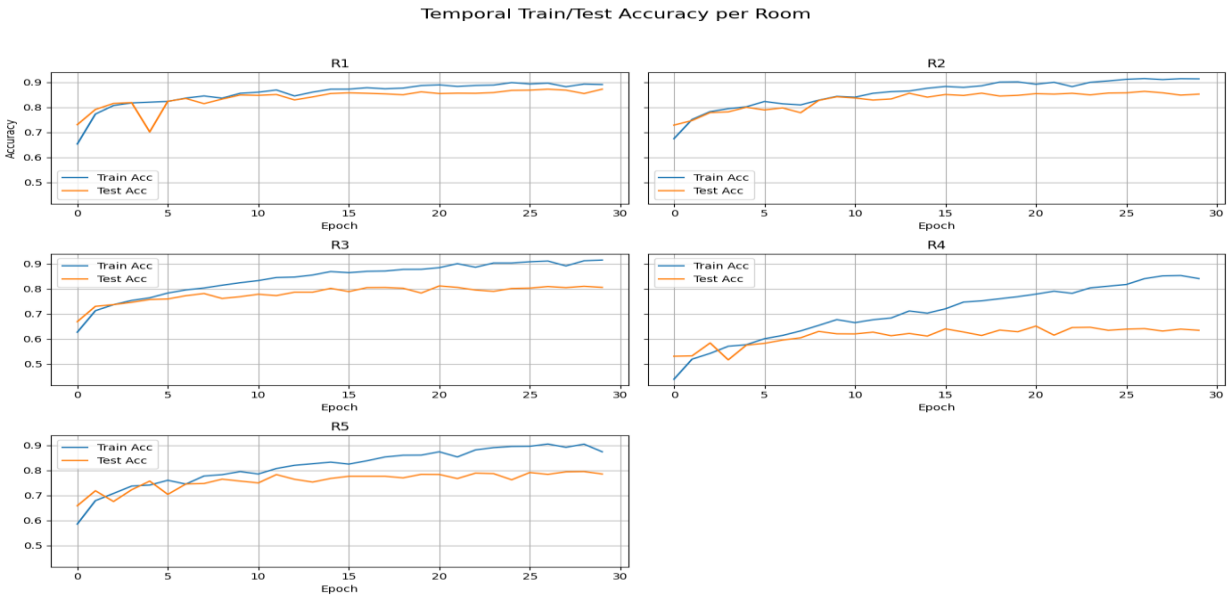

Figure 4. Temporal train vs test accuracy over epochs

The plot comparing temporal actual vs. predicted values shows how closely the model's predictions match shown in Figure 5. An overlap between the two lines means that the model is predicting the actual occupancy and if both lines be closer means the model is learning.

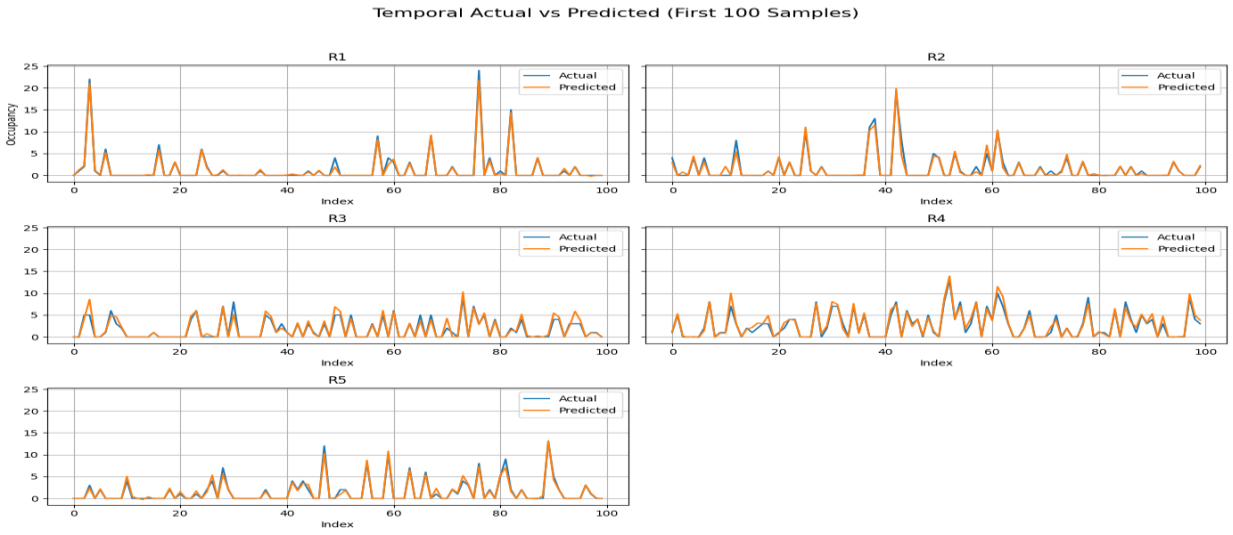

Figure 5. Temporal actual vs predicted 100 values

The spatial Transformer model used a hidden dimension of 32, 2 attention heads, and a single Transformer encoder layer with a dropout rate of 10 percent. This model took the encoded outputs from all five rooms (R1 to R5), learned the spatial dependencies across them using self-attention, and produced spatial predictions for each room simultaneously. This setup allowed the model to learn both temporal patterns within each room and spatial relationships between rooms.

We measured how well the spatial Transformer model performed across all rooms using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) shown in the below table. These metrics helped us evaluate how accurately the model predicted the number of people in each room while also capturing the relationships between rooms.

Rooms	RMSE	MAE	R^2
Room1 (R1)	1.0163	0.3649	0.9635
Room2 (R2)	1.4141	0.5537	0.8323
Room3 (R3)	0.5975	0.3229	0.9423
Room4 (R4)	0.8898	0.5867	0.9331
Room5 (R5)	0.6577	0.3417	0.9417

The results show that the model performed well in most rooms, especially in Room 1, 3, 4 and 5. However the Room 2 had the lowest R^2 score of 0.83 and the highest RMSE and MAE, suggesting it was more challenging for the model to accurately learn occupancy patterns in that room. Despite this, the overall performance across rooms remains strong, showing that the model captures both temporal patterns from each room and spatial relationships among rooms.

The spatial plots help us understand how the multi-room Transformer model performed across all rooms shown in Figure 6. The train and test loss plot show whether the model is learning effectively across epochs for all rooms together. If both losses decrease, it means training is stable.

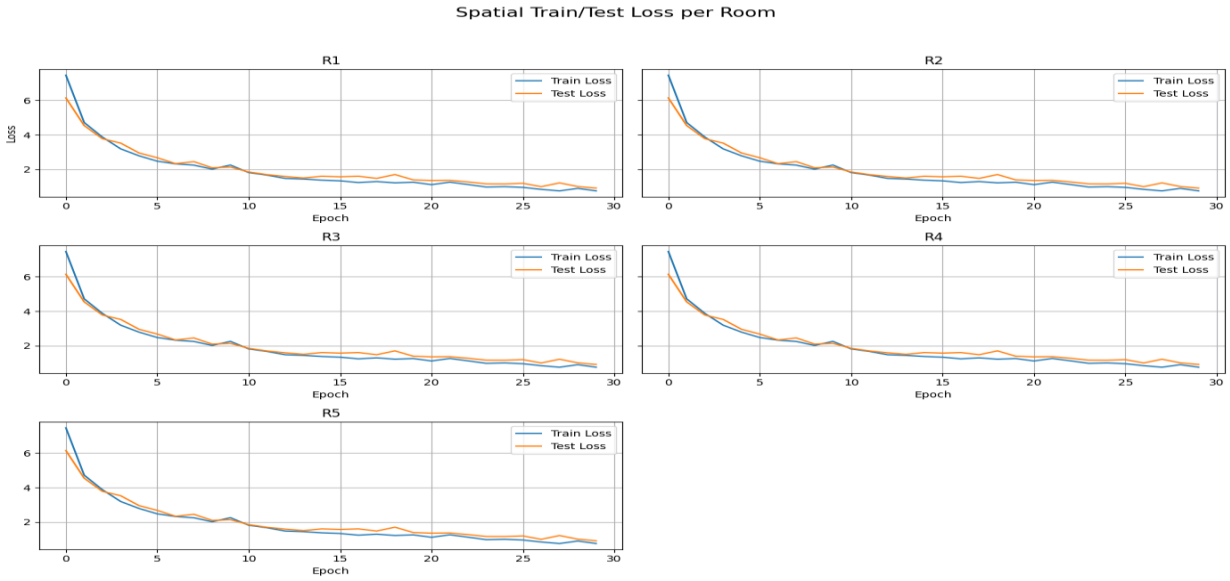

Figure 6. Spatial train vs test loss over epochs

The spatial accuracy plot in Figure 7 compares the model`s occupancy prediction accuracy across all rooms. Consistent accuracy between training and testing indicates strong performance.

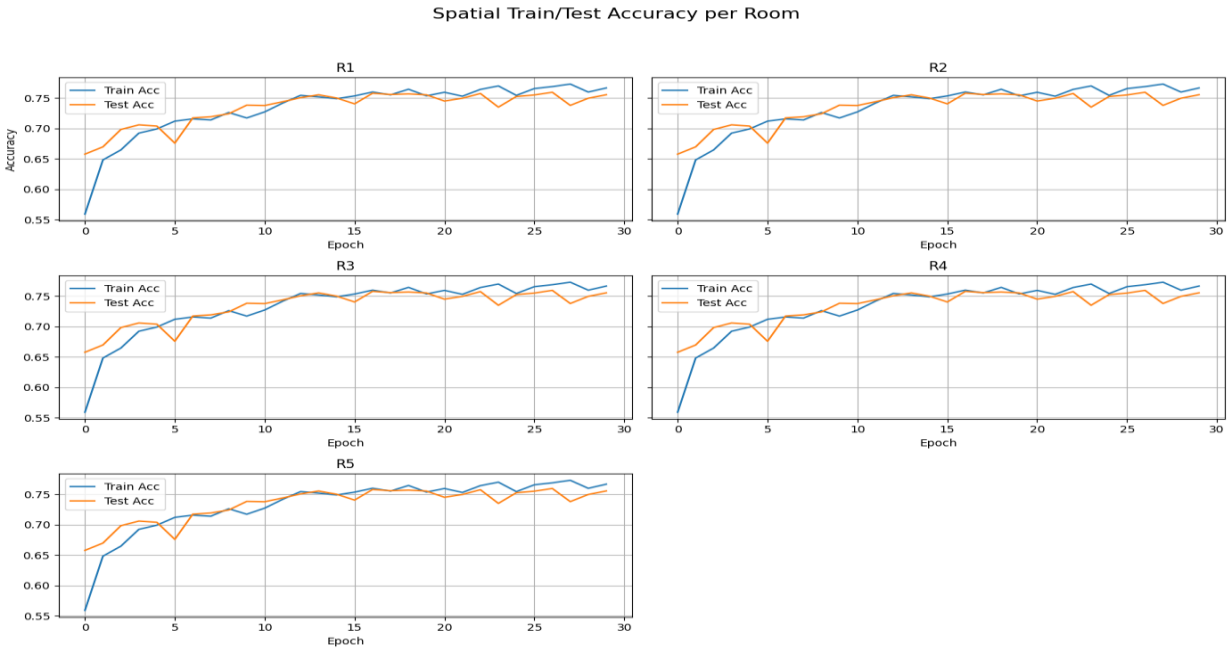

Figure 7. Spatial train vs test accuracy

Finally, the spatial actual vs predicted plot in Figure 8 visually compares the model`s predicted number of occupants to the actual values for each room, helping us see if the model can closely follow real occupancy patterns over time.

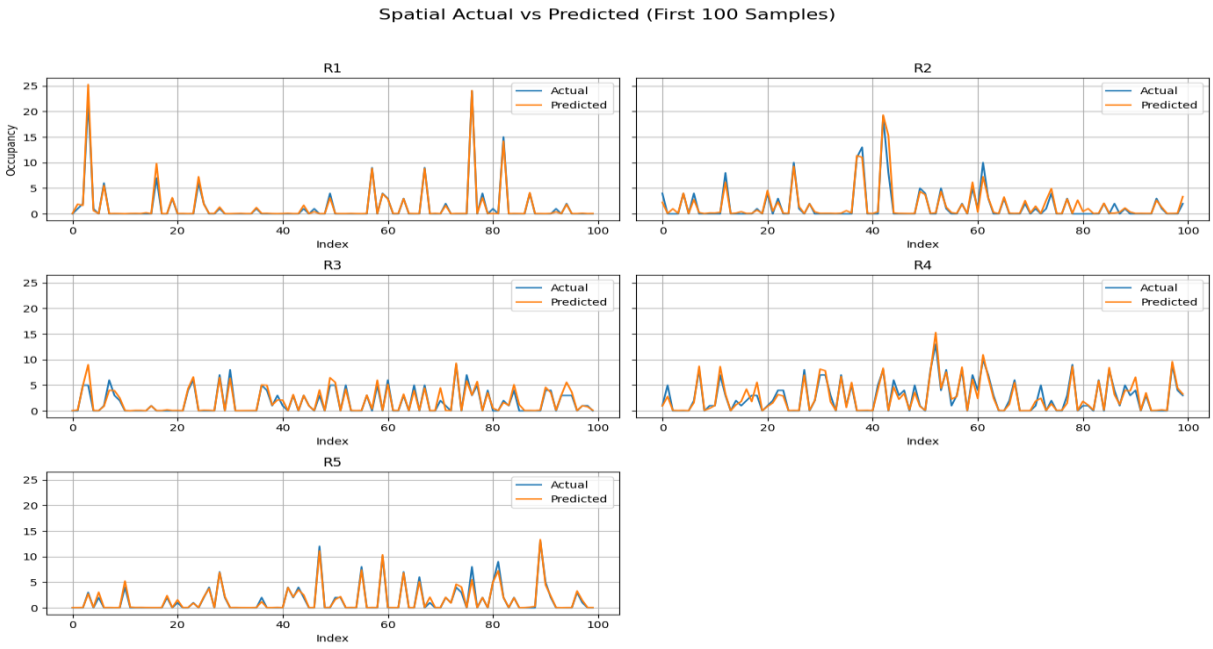

Figure 8. Spatial actual vs predicted first 100 samples

VI. Conclusion

In this project, we built a transformer-based occupancy prediction model that uses both temporal and spatial encoders to capture time sequence-based patterns within each room and the relationships among rooms. The temporal transformer encoder enables each room to learn occupancy estimation, while the spatial encoder allowed the model to understand how rooms are dependent on each other. This model is helpful when sensor data is missing or incomplete, as the model can still estimate occupancy using patterns from other rooms.

Despite the complexity of the model, it is highly parallelizable and runs efficiently on a GPU, making it suitable for larger datasets and real-time applications. During training, the spatial transformer achieved an average accuracy of 92%, showing strong overall performance. However, Room 2 showed lower accuracy R^2 about 83%, which suggests the need for further investigation. Importantly, even if accuracy is slightly lower than simpler models, the spatial-temporal model preserves valuable context from both time and space and making it possible for estimating occupancy when certain sensors fail or data is missing.

For future work, we plan to find better hyperparameters to improve prediction accuracy, explore multi-step forecasting (predicting several future time steps at once), test model deployment on an IOT device such as a Raspberry Pi.

This project also has value for engineering and computer science education. It was done as part of an undergraduate research project and used basic ideas such as machine learning, time-series data, environmental sensors, and GPU training. The work involved building and testing a model using real building data and Transformer models. It shows how machine learning can be used in engineering to solve real energy problems.

References

[1] Z. D. Tekler, E. Ono, Y. Peng, S. Zhan, B. Lasternas, and A. Chong, “ROBOD, room-level occupancy and building operation dataset,” *Building Simulation*, vol. 15, no. 12, pp. 2127–2137, Dec. 2022, doi: 10.1007/s12273-022-0925-9

[2] W. Wang, J. Chen, and T. Hong, “Occupancy prediction through machine learning and data fusion of environmental sensing and WiFi sensing in buildings,” *Automation in Construction*, vol. 94, pp. 104–114, Oct. 2018, doi: 10.1016/j.autcon.2018.07.007

[3] A. N. Sayed, Y. Himeur, and F. Bensaali, “Deep and transfer learning for building occupancy detection: A review and comparative analysis,” *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105254, Oct. 2022, doi: 10.1016/j.engappai.2022.105254

[4] Z. Chen, C. Jiang, and L. Xie, “Building occupancy estimation and detection: A review,” *Energy and Buildings*, vol. 169, pp. 260–270, Jun. 2018, doi: 10.1016/j.enbuild.2018.03.084

[5] I. Qaisar, K. Sun, Q. Zhao, T. Xing, and H. Yan, “Multi-sensor-based occupancy prediction in a multi-zone office building with Transformer,” *Buildings*, vol. 13, no. 8, p. 2002, Aug. 2023, doi: 10.3390/buildings13082002

[6] J. E. Marchelina, S. -Y. Chou, V. F. Yu, A. Dewabharata, V. C. Sugiarto and I. Karijadi, "Two-Stages Occupancy Number Detection Based on Indoor Environment Attributes By Utilizing Machine Learning Algorithm," *2019 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, New Taipei, Taiwan, 2019, pp. 38-43, doi: 10.1109/iFUZZY46984.2019.9066241

[7] B. Yang, F. Haghighat, B. C. M. Fung, and K. Panchabikesan, “Season-based occupancy prediction in residential buildings using machine learning models,” *e-Prime – Advances in Electrical Engineering, Electronics and Energy*, vol. 1, p. 100003, 2021, doi: 10.1016/j.prime.2021.100003

[8] X. Liang and H. Wang, "Hybrid Transformer-RNN Architecture for Household Occupancy Detection Using Low-Resolution Smart Meter Data," *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, Singapore, Singapore, 2023, pp. 1-6, doi: 10.1109/IECON51785.2023.10312340

[9] A. Shoari Nejad, R. Alaiz-Rodríguez, G. D. McCarthy, B. Kelleher, A. Grey, and A. Parnell, "SERT: A Transformer Based Model for Spatio-Temporal Sensor Data with Missing Values for Environmental Monitoring," *arXiv preprint arXiv:2306.03042v2*, 2023. [Online]. Available: https://arxiv.org/abs/2306.03042

[10] F. Wang, Q. Feng, Z. Chen, Q. Zhao, Z. Cheng, J. Zou, Y. Zhang, J. Mai, Y. Li, H. Reeve, Predictive control of indoor environment using occupant number detected by video data and CO2 concentration, Energy and Buildings 145 (2017) 155–162

[11] J. T. Giger and M. Markward, “The Need-to-Know Caregiver Perspectives Toward Using Smart Home Technology,” Journal of Social Work in Disability & Rehabilitation, vol. 10, no. 2, pp. 96–114, Apr. 2011, doi: https://doi.org/10.1080/1536710x.2011.571529

[12] M. S. Aliero et al., “Non-Intrusive Room Occupancy Prediction Performance Analysis Using Different Machine Learning Techniques,” Energies, vol. 15, no. 23, p. 9231, Dec. 2022, doi: https://doi.org/10.3390/en15239231

[13] L. Noriega. "Multilayer perceptron tutorial." School of Computing. Staffordshire University 4 (2005): 5

[14] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” arXiv:1511.08458 [cs], Dec. 2015, Available: https://arxiv.org/abs/1511.08458

[15] Kanagachidambaresan, G.R., Ruwali, A., Banerjee, D., Prakash, K.B. (2021). Recurrent Neural Network. In: Prakash, K.B., Kanagachidambaresan, G.R. (eds) Programming with TensorFlow. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-57077-4_7

[16] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” arXiv:1801.01078 [cs], Feb. 2018, Available: https://arxiv.org/abs/1801.01078