Senior Project II Final Report– Spring 2025

Group Number	31
Project	Synthetic Wi-Fi fingerprint generation and indoor localization under Wi-Fi scan throttling constraint
Team Members	Imangali Zhumangali, Alan Igilikov, Abylay Kubeyev, Abylay Sydykov
Project Advisor/Co-Advisors	Shinnazar Seytnazarov

Executive Summary (10%)

This project addresses the growing demand for accurate indoor localization in environments where GPS is ineffective and Wi-Fi scan availability is constrained by modern smartphone operating systems. Specifically, it tackles the challenge introduced by Android's Wi-Fi scan throttling policies, which limit scan frequency and thereby degrade the performance of traditional Wi-Fi fingerprint-based positioning systems.

The key objective of the project was to design and evaluate a hybrid indoor positioning system capable of operating under Wi-Fi scan throttling. The proposed solution combines **synthetic Wi-Fi fingerprint data generation** using a **Conditional Denoising Diffusion Probabilistic Model (cDDPM)** with **Pedestrian Dead Reckoning (PDR)** based on **CNN+LSTM deep learning** models for IMU sensor data. A **fusion strategy** then integrates these two modalities to deliver a robust indoor localization system.

The methodology included: (1) generating synthetic RSSI data to augment real-world datasets, (2) constructing a localization pipeline using k-Nearest Neighbors (kNN) for Wi-Fi positioning, (3) building a displacement prediction model using CNN+LSTM for IMU-based tracking, and (4) implementing a throttling-aware fusion algorithm to simulate real-world constraints.

The evaluation results showed that diffusion-generated synthetic data can significantly reduce localization errors—by up to 22% in low-data scenarios. The hybrid model maintained continuous trajectory estimates and partially mitigated PDR drift despite infrequent Wi-Fi corrections, validating the effectiveness of the fusion approach under Android scan throttling. This project exemplifies the design, implementation, and evaluation of a practical, computing-based solution to a real-world systems limitation.

Introduction (10%)

The demand for precise indoor location services has grown substantially across numerous sectors including commercial, retail, healthcare, and security applications (Zafari et al., 2019). As traditional outdoor positioning technologies like GPS prove inadequate within buildings, Indoor Positioning Systems (IPS) have developed to address this technological gap. Among the various approaches to indoor localization, Wi-Fi fingerprinting has gained prominence due to its ability to leverage existing wireless network infrastructure commonly found in modern buildings (He & Chan, 2016).

Despite its advantages, practical implementation of Wi-Fi fingerprinting encounters significant challenges, primarily related to the resource-intensive nature of data collection and the inherent variability of Wi-Fi signals in dynamic indoor environments (Khalajmehrabadi et al., 2017). To overcome these limitations, this research focuses on the generation of synthetic Wi-Fi fingerprint data as a promising alternative that can create artificial yet realistic training data for indoor localization systems.

Indoor localization has attracted extensive research attention in recent years due to the increasing demand for accurate location-based services in indoor environments where Global Navigation Satellite Systems (GNSS) fail to perform effectively. Among various approaches, the fusion of smartphone inertial sensors with Wi-Fi Received Signal Strength (RSS) has emerged as one of the most widely investigated and practically feasible solutions, owing to the ubiquity of smartphones and existing Wi-Fi infrastructure.

Despite the progress in hybrid indoor localization systems, one of the major challenges when implementing smartphone-based solutions in real-world scenarios is the restriction policies enforced by mobile operating systems, particularly Android. Starting from Android 8.0 (API level 26), Google introduced Wi-Fi scan throttling mechanisms aimed at preserving battery life and protecting user privacy. These policies limit foreground apps to approximately 4 Wi-Fi scans every 2 minutes, and background apps are limited to one scan every 30 minutes. Such throttling significantly affects the performance of Wi-Fi-based localization systems, especially when they rely on frequent RSS updates to correct PDR drift. The reduction in Wi-Fi scan rates implies that a user may travel considerable distances before obtaining updated Wi-Fi measurements, increasing the reliance on less accurate PDR data and consequently degrading the overall positioning accuracy.

Indoor Positioning Systems (IPS)

An Indoor Positioning System comprises a network of devices and methodologies designed to locate objects or individuals within enclosed spaces where satellite-based technologies are ineffective. These systems serve a wide range of applications across multiple industries (Liu et al., 2007). In healthcare environments, IPS facilitates equipment tracking and patient monitoring, enhancing operational efficiency and safety protocols. Retail establishments implement IPS for improving customer experience through indoor navigation and context-aware promotions. In logistics and manufacturing, these systems optimize asset tracking and workflow management, while security operations benefit from enhanced personnel monitoring capabilities (Zafari et al., 2019).

The technological foundation of IPS encompasses diverse approaches including Wi-Fi, Bluetooth, Ultra-Wideband (UWB), Radio Frequency Identification (RFID), magnetic positioning, acoustic signals, and vision-based systems. The selection of a specific technology typically depends on environmental requirements, accuracy needs, and budgetary constraints. This technological diversity highlights the absence of a universal standard for IPS implementation, with each solution customized to the specific characteristics of the deployment environment (He & Chan, 2016).

Wi-Fi Fingerprinting for Indoor Localization

Wi-Fi fingerprinting has become a widely adopted technique for indoor positioning due to the ubiquitous presence of Wi-Fi networks and the broad compatibility with consumer mobile devices (Zou et al., 2016). The methodology operates through two distinct phases: offline training and online positioning.

During the offline phase, a comprehensive site survey is conducted to collect Received Signal Strength Indicator (RSSI) values from available Wi-Fi Access Points (APs) at predetermined reference points throughout the target environment. These RSSI measurements, combined with their corresponding spatial coordinates, constitute a radio map or fingerprint database. The accuracy and density of this reference database significantly impact the system's localization performance, though creating this foundation requires substantial time and effort (Khalajmehrabadi et al., 2017).
In the online phase, a mobile device seeking to determine its position measures real-time RSSI values from

surrounding APs. These measurements are compared against the stored fingerprint database using various algorithms such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), or neural networks to estimate the device's current location. The effectiveness of this process heavily depends on both the comprehensiveness of the reference database and the robustness of the matching algorithm in handling the natural variability of Wi-Fi signals (He & Chan, 2016).

To address these limitations, generating **synthetic Wi-Fi fingerprint data** has emerged as a promising solution. Synthetic data generation can reduce manual effort, improve model generalization, and simulate rare or variable conditions that may be underrepresented in real-world datasets. It also enables continuous dataset updates without repeated site visits and eliminates privacy risks, as the data does not originate from real users.

In this project, we propose using a **diffusion probability model** to generate realistic synthetic RSSI fingerprints for indoor localization. Diffusion models, originally developed for image synthesis, are now being adapted for structured tabular data. By modeling the distribution of real RSSI fingerprints, our approach aims to create synthetic data that supports robust and accurate location estimation.

IMU data-based Indoor Localization (PDR) and Fusion

While Wi-Fi fingerprinting offers a robust method for indoor localization by leveraging existing infrastructure, its performance can be significantly hampered by practical constraints such as the Wi-Fi scan throttling implemented in modern mobile operating systems like Android. To address the resulting infrequency of absolute position updates, Inertial Measurement Unit (IMU) data-based localization, commonly known as Pedestrian Dead Reckoning (PDR), presents a complementary approach.

PDR leverages the suite of inertial sensors typically found in smartphones—namely the accelerometer, gyroscope, and magnetometer—to estimate a user's trajectory relative to a known starting point. The core principles involve:

1. Step Detection: Analyzing the accelerometer data patterns (e.g., peaks in vertical acceleration) to identify individual steps taken by the user.
2. Step Length Estimation: Estimating the distance covered by each detected step. This can be done using empirical models based on step frequency, acceleration magnitude, or user-specific parameters (like height), although simpler fixed-length estimates are sometimes used.
3. Heading Estimation: Determining the user's direction of movement. This typically involves integrating angular velocity data from the gyroscope to track changes in orientation. However, gyroscopes suffer from cumulative drift over time. To correct this and provide an absolute orientation reference (relative to North), magnetometer readings are often fused with gyroscope data, frequently using sensor fusion algorithms like Complementary or Kalman filters. The accelerometer can also contribute by sensing the direction of gravity to help determine device tilt.

By combining the detected steps, estimated step length, and calculated heading, PDR continuously updates the user's relative position (x, y coordinates) without relying on external signals like Wi-Fi.

This continuous, high-frequency relative positioning capability makes PDR particularly valuable for mitigating the Android Wi-Fi scan throttling issue. While Wi-Fi scans might only be available infrequently (e.g., once every 30 seconds for background apps, or 4 times per 2 minutes for foreground apps under throttling), PDR can fill the

gaps, providing smooth and continuous trajectory estimation between these sparse Wi-Fi updates. This prevents the localization from becoming stagnant or jumping abruptly when a new Wi-Fi fix is obtained.

However, PDR's primary limitation is its **cumulative error or drift**. Small inaccuracies in step detection, step length estimation, and particularly heading calculation accumulate over time, causing the estimated position to gradually diverge from the true position. This is where fusion with Wi-Fi fingerprinting becomes highly effective.

The synergistic approach combines the strengths of both methods: PDR provides continuous relative tracking, while Wi-Fi fingerprinting offers periodic absolute position fixes (albeit with their own inherent noise and uncertainty). When a Wi-Fi fingerprinting location estimate becomes available, it can be used to correct the accumulated drift in the PDR estimate, effectively re-calibrating the user's position. This fusion is commonly implemented using advanced filtering techniques like Extended Kalman Filters (EKF) or Particle Filters, which can optimally combine the predictions from the PDR model with the measurements from the Wi-Fi localization system.

In the context of our project, we propose such **a hybrid fusion model**. The system primarily relies on PDR for continuous localization, tracking the user's movement via IMU sensor data. Then, periodically, leveraging the available Wi-Fi scan data (even under throttling constraints, e.g., receiving updates approximately every 30 seconds), the system obtains an absolute position estimate from the Wi-Fi fingerprinting module. This Wi-Fi-derived position is then used to correct and reset the drift accumulated by the PDR component, ensuring both continuous tracking and long-term accuracy. This approach aims to provide a robust and practical indoor localization solution that effectively navigates the challenges posed by OS-level restrictions like Wi-Fi scan throttling.

- Section 1 provides an **Executive Summary**, offering a high-level overview of the project's objectives and outcomes.
- Section 2 presents the **Introduction**, outlining the motivation, problem statement, and goals of the project. ● Section 3 covers **Background and Related Work**, reviewing key concepts in indoor positioning systems and Wi-Fi fingerprinting, as well as relevant literature on synthetic data generation.
- Section 4 details the **Project Approach**, including the design of the proposed table diffusion model and the methodology used for generating synthetic RSS data.
- Section 5 discusses **Project Execution**, describing the data preprocessing, model implementation, and training process.
- Section 6 presents the **Evaluation**, including experimental setup, evaluation metrics, and analysis of results.
- Section 7 concludes the report, summarizing findings and proposing directions for **Future Work**. ● Finally, Section 8 lists all **References** cited throughout the report.

Background and Related Work (15%)

Generative Adversarial Networks (GANs)

Previous research has extensively explored Generative Adversarial Networks for synthesizing Wi-Fi fingerprint data, establishing them as an influential baseline in the field. The fundamental GAN architecture employs a dual-network structure comprising a generator and discriminator engaged in adversarial training (Nabati et al.,

2020). This approach enables the system to learn complex signal distributions without explicit modeling of the indoor propagation environment.

Several researchers have adapted the standard GAN framework to address specific challenges in indoor positioning. Nabati et al. (2020) demonstrated that Semi-Supervised GANs (SSGANs) could effectively generate location-tagged fingerprint data, addressing the critical need for labeled training samples in supervised localization algorithms. Their experimental evaluation showed a 15% improvement in positioning accuracy when augmenting limited real datasets with GAN-generated synthetic samples. Similarly, Conditional GANs (CGANs) have been implemented to create fingerprints based on specific spatial coordinates, providing more precise control over the generation process.

Despite their demonstrated capabilities, GAN-based approaches exhibit significant limitations for Wi-Fi fingerprint synthesis. The adversarial training process frequently suffers from instability issues, including mode collapse where the generator produces limited varieties of outputs (Nabati et al., 2020). Additionally, GANs struggle to capture the full conditional distribution of RSSI values across spatial coordinates, often missing subtle signal variations that occur in real environments. These shortcomings have motivated exploration of alternative generative frameworks that might better preserve the statistical properties essential for accurate positioning.

Variational Autoencoders (VAEs)

Variational Autoencoders represent an alternative probabilistic approach to generating synthetic Wi-Fi fingerprints, offering distinct advantages over GAN-based methods. The VAE architecture encompasses an encoder that maps fingerprint data to a lower-dimensional latent space and a decoder that reconstructs fingerprints from latent representations (Hoang et al., 2019). Unlike the adversarial training in GANs, VAEs optimize a variational lower bound on the data likelihood, resulting in more stable training dynamics.

Research by Hoang et al. (2019) extended standard VAE architectures with recurrent neural networks specifically designed for sequential RSSI data, demonstrating how deep generative models can effectively capture temporal patterns in Wi-Fi signals. Their implementation achieved a 22% reduction in average localization error compared to traditional fingerprinting methods when deployed in dynamic environments where signal strengths fluctuate over time.

The primary advantage of VAEs for fingerprint generation lies in their explicit probabilistic formulation, which provides better uncertainty quantification in the generated samples. However, comparative analyses reveal that VAE-generated fingerprints often exhibit smoothing effects that obscure the fine-grained signal variations critical for distinguishing nearby locations. This limitation becomes particularly pronounced in environments with dense reference point distributions, where subtle RSSI differences significantly impact positioning accuracy (Hoang et al., 2019). Furthermore, VAEs typically struggle with handling the conditional generation aspect that is essential for creating location-specific fingerprints.

Diffusion Models and Table Diffusion

Recent advancements in generative modeling have introduced diffusion models as a promising alternative for synthetic data generation, addressing many limitations of previous approaches. Diffusion models operate through a gradual noise-addition process followed by learned denoising, producing high-quality samples that maintain complex statistical relationships present in the training data (Ho et al., 2020). Unlike GANs and VAEs, diffusion models have demonstrated superior capability in capturing multimodal distributions and fine-grained details,

making them particularly suitable for Wi-Fi fingerprint synthesis.

Table Diffusion, our selected methodology, represents a specialized adaptation of diffusion models designed specifically for tabular data like Wi-Fi fingerprints. This approach treats RSSI values from multiple access points as structured tabular data conditioned on spatial coordinates, enabling precise modeling of the relationship between location and expected signal strengths. Research by Yang et al. (2022) has shown that diffusion-based methods can achieve significant reductions in localization error by better preserving the spatial correlation patterns in RSSI distributions.

Our implementation of Table Diffusion addresses several critical challenges in synthetic fingerprint generation:

1. **Conditional Generation**: Unlike standard diffusion models, Table Diffusion effectively incorporates spatial coordinates (X,Y) as conditional inputs, ensuring generated RSSI values correspond to specific locations.
2. **Feature Correlation Preservation**: The gradual denoising process maintains interdependencies between signals from different access points, capturing the complex relationship between spatial proximity and signal similarity.
3. **Handling Sparsity**: Table Diffusion robustly handles missing or sparse RSSI measurements, a common challenge in real-world data collection scenarios.

Wi-Fi Positioning Component For IMU + WiFi Fusion Models

Within fusion systems, Wi-Fi is primarily used to provide periodic absolute position corrections. While some approaches use triangulation (Kim et al., 2022), fingerprinting is more common. Various algorithms are employed for matching real-time RSSI scans to the fingerprint database:

K-Nearest Neighbors (kNN): A widely used baseline algorithm that estimates position based on the coordinates of the 'k' most similar fingerprints in the database, often weighted by similarity (implicitly related to methods like Shi et al., 2018, though specific algorithms vary). This aligns with the approach chosen for our project.

Support Vector Machines (SVM): Zhang et al. (2016) utilized SVM as the location algorithm for their Wi-Fi component before EKF fusion.

Advanced Preprocessing and Matching: To combat RSSI noise and improve matching, researchers have used Gaussian Process Regression (GPR) for denoising and clustering techniques (like improved K-means) to structure the database and reduce matching complexity (Wang et al., 2022). Sun et al. (2023) proposed a novel ranked RSS vocabulary combined with the FAB-MAP algorithm for robust place recognition, mitigating device bias.

PDR Component For IMU + WiFi Fusion Models

The PDR component provides the continuous trajectory estimate between Wi-Fi fixes. Its accuracy hinges on robust step detection, step length estimation (SLE), and heading determination:

Step Detection and SLE: Yao et al. (2020) focused specifically on robust step detection (using DTW-based peak prediction) and walking pattern-aware SLE (using random forests) to handle variations like Normal Walk, Marching in Place, and Quick Walk. Zhang et al. (2016) employed adaptive step detection based on time windows and dynamic thresholds. Shi et al. (2018) used filtering and comparison of adjacent RSS values to potentially identify noise or shaking distinct from steps, and dynamically adjusted step size based on fusion results and map matching.

Heading Estimation: This remains the most challenging aspect due to gyroscope drift and magnetic interference. Basic approaches rely on integrating gyroscope data, often corrected using magnetometers and accelerometers via complementary or Kalman filters (Shi et al., 2018; Zhang et al., 2016).

Complex Motion Recognition (Deep Learning): Luo et al. (2021) specifically addressed the limitation of PDR in non-forward movements. They employed a deep learning model combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to recognize complex patterns (forward, backward, left/right sidestepping) from IMU data. Based on the recognized pattern, a correction angle was applied to the standard INS-derived heading to obtain the true motion direction. This use of CNN+LSTM for interpreting IMU data for PDR aligns closely with the PDR methodology selected for our project.

Fusion Strategies

Filtering (EKF/UKF): Extended Kalman Filters (EKF) are a common choice for fusing Wi-Fi position measurements with PDR state predictions (Zhang et al., 2016; Wang et al., 2022). Unscented Kalman Filters (UKF) have also been explored (Sun et al., 2023 implicitly refers to filtering in related work). These methods provide a statistically optimal way to merge noisy data streams but require careful modeling of system dynamics and noise characteristics.

Adaptive Weighting: Shi et al. (2018) proposed an adaptive fusion approach where the weighting between PDR and Wi-Fi results changes dynamically, potentially based on signal quality or estimated accuracy, often coupled with map matching for further refinement.

End-to-End Learning: Nurpeiissov et al. (2022a) bypassed explicit PDR calculation and filtering, using an end-to-end RNN (LSTM-based) model trained on their sequential IMUWiFine dataset (Nurpeiissov et al., 2022b). The network learns to implicitly fuse the time-series data from both sensors to directly predict high-frequency position outputs. While powerful, this requires large, accurately time-synchronized, sequential datasets.

Complementary Integration: Kim et al. (2022) focused on a tight integration where Wi-Fi triangulation helps correct PDR drift and step length estimation, while PDR smooths Wi-Fi fluctuations, highlighting the reciprocal benefits.

Fusion of CNN+LSTM for PDR and kNN for Wi-Fi fingerprinting:

Our project's approach, combining deep learning (CNN+LSTM) for PDR processing with kNN for Wi-Fi fingerprinting and a suitable fusion strategy, is well-supported by the successes reported in the related literature. The challenge of Wi-Fi scan throttling underscores the need for robust fusion, as standalone Wi-Fi performance degrades significantly with infrequent updates.

Deep Learning for PDR Enhancement: Luo et al. (2021) specifically demonstrated the effectiveness of using CNN+LSTM models to recognize complex motion patterns from IMU data, enabling the correction of PDR heading estimates in non-forward movements and leading to significantly more accurate PDR trajectories compared to traditional methods that assume forward motion. This validates the potential of using deep learning to create a more sophisticated and accurate PDR component within a fusion system.

Fusion System Performance: Various studies implementing different fusion strategies have achieved meter-level or sub-meter accuracy:

- Zhang et al. (2016), using EKF fusion with SVM-based Wi-Fi and improved PDR, reported an average localization error of 0.57 meters, a significant improvement over standalone Wi-Fi (1.21m) or PDR (0.71m).
- Shi et al. (2018), employing an adaptive fusion of PDR and RSS fingerprinting aided by map matching, achieved average errors of 0.41m on a line route and 1.18m on an L-shaped route, drastically reducing the errors of PDR alone (4.33m and 5.56m, respectively). Their overall average fusion error was 1.5 meters.
- Wang et al. (2022) combined GPR-denoised Wi-Fi, PDR, and EKF fusion to achieve an average positioning error of 1.76 meters.
- Sun et al. (2023), using pose graph optimization to fuse PDR with ranked Wi-Fi RSS, achieved average errors of 1.88m and 1.84m on different datasets.
- Nurpeiissov et al. (2022a) showcased an end-to-end deep learning approach (LSTM-based) for direct fusion of sequential Wi-Fi and IMU data, achieving a mean error distance of 1.1 meters on their IMUWiFine test set (Nurpeiissov et al., 2022b), highlighting the capability of deep learning in the fusion task itself.

These demonstrated results, ranging from 0.57m to ~1.8m average/mean error, confirm that fusing even sparsely available Wi-Fi updates (like those obtained via kNN fingerprinting) with continuous PDR (enhanced by methods like CNN+LSTM) can yield accurate and robust indoor localization. The success of deep learning approaches specifically for PDR motion analysis (Luo et al., 2021) and for end-to-end fusion (Nurpeiissov et al., 2022a) further supports our chosen methodology as a promising direction for developing effective indoor localization systems capable of operating under real-world constraints like Wi-Fi scan throttling.

Project Approach (20%)

• Provide a detailed description of the solution, including software/hardware architecture, algorithms, workflows, roles, features, tools, and use case diagrams.

• Explain any third-party components that the group did not implement themselves and their

integration. • Show how the project team functioned effectively to develop a computing-based

solution.

Our generative model architecture is based on cDDPM, that conditionally generates RSSI data. It is trained to generate RSSI signal values and conditioned with X and Y coordinates. The architecture is shown in the Fig1 below:

Dataset:

For this project, the public WIFI fingerprinting dataset was used (Minh et al., 2019). It has 12 columns containing RSSI signal values from different APs and 2 columns containing X and Y location coordinates. The dataset was collected with the autonomous robot in the 21 m x 16 m sized corridor. According to the authors of the dataset, the localization accuracy of the robot is 0.07 m ± 0.02 m. In Fig. 2, the view of the location and reference points can be seen. There are overall 345 reference points and for each point RSSI signal values were recorded 50 times. Thus, the shape of the dataset is (17270, 13).

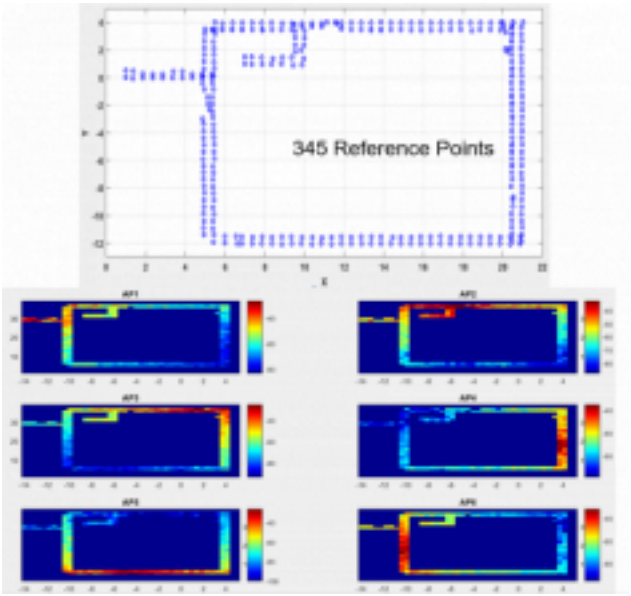

Figure 1. Open source Wi-Fi RSSI Indoor Localization

Noise prediction model:

```python
## Neural Network consists of 6 fully connected layers with following dimensions and each neuron is connected with ReLU function:
nn.Linear(13, 512),
nn.ReLU(),
nn.Linear(512, 512),
nn.ReLU(),
nn.Linear(512, 256),
nn.ReLU(),
nn.Linear(256, 170),
nn.ReLU(),
nn.Linear(170, 128),
nn.ReLU(),
nn.Linear(128, 11)
## Model was hyperparameter tuned using library optima.
## Best hyperparameters:
diffusion_steps=5,
batch_size=64,
n_epochs=70,
lr=0.0004360345526399257
```

Core Algorithms & Workflows

1. Data Preprocessing

Data preprocessing involves several key steps to ensure that the model can effectively learn from the dataset. In the context of the Diffusion model, these steps are crucial for preparing the RSSI data and the condition features for training.

Scaling

The RSSI values, which typically range in various magnitudes, need to be standardized for the model to perform effectively. This is achieved using **StandardScaler** from scikit-learn, which scales the RSSI columns to have a mean of 0 and a standard deviation of 1. This ensures that all the features are on the same scale, preventing features with larger values from dominating the learning process.

Dataset split

For training the generative model, the dataset is reduced by certain percentages (from 10% to 90%) in order to see the performance of the generative model under the condition of limited data. Further data is divided into training and validation sets. 80% of the data is used for training, and the remaining 20% is held out for validation. This split is crucial to assess how well the model generalizes to unseen data

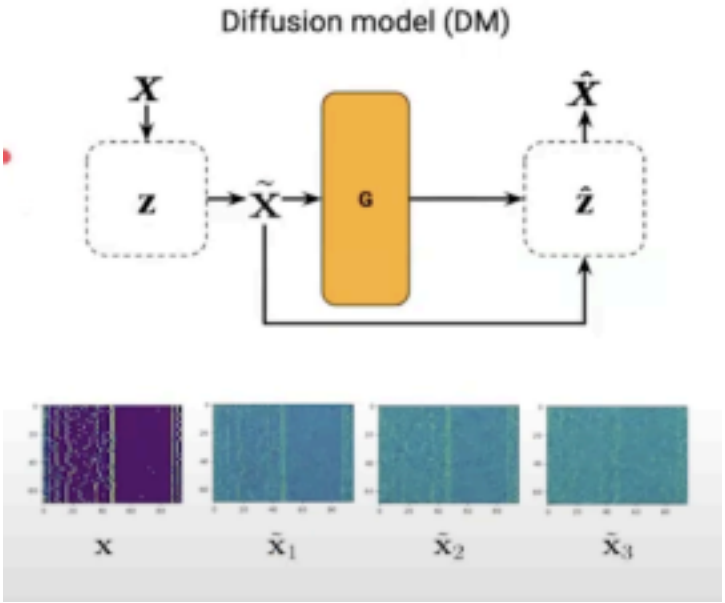

Figure 2. Diffusion model visualisation

2. Noise scheduler using cosine β

The noise scheduler controls how the noise is added to the data at each step of the diffusion process, as well as how the model reverses this process during training and sampling.

The cosine noise schedule was chosen as the best for this case. It significantly increased the performance of

generative models compared to other noise schedulers such as linear scheduler. The value of β increases as t progresses, starting from a small value and gradually growing to add more noise at later timesteps. The scheduler follows the formula:

$$\beta_t = \frac{1 - \cos\left(\frac{\pi \cdot t}{T}\right)}{2}$$

t - the current time step,
T - total number of steps in the diffusion process.

3. Loss Aggregation Over T Steps

During the training phase, the model learns to predict the added noise for each timestep in the diffusion process. The model predicts the noise at each timestep, which is then compared with the actual noise that was added to the data at that step. The Mean Squared Error loss is used to compute the difference between the predicted noise and the actual noise. The loss is accumulated over all diffusion steps, and the average loss is computed by dividing the total loss by the number of steps.This aggregated loss is then back propagated to update the model's weights, which enables the model to gradually improve its ability to reverse the noise process.

4. Sampling (Reverse Diffusion)

Once the model has been trained, the reverse diffusion process is used to **generate synthetic RSSI data**. The reverse diffusion process starts with **Gaussian noise**, which serves as the starting point. This is a crucial part of the model since it helps generate new samples from random noise.

Algorithm of sampling:

- The Gaussian noise is sampled randomly, creating a noisy version of the data.
- The model iteratively removes the noise across multiple timesteps, starting from the noise-filled data and moving backward to a cleaner sample.

- At each step, the model predicts the noise (ε), and this prediction is used to denoise the data. The model subtracts the predicted noise from the noisy data to reduce the noise at each step.
- Over T timesteps, the data transitions from noisy to clean, essentially reversing the diffusion process that added the noise in the first place.

5. Localization with KNN

Once the synthetic RSSI data is generated using the reverse diffusion process, it is used in a **KNN** algorithm to localize the position based on RSSI values. The dataset is divided into **real RSSI data** from the actual experiments and **synthetic RSSI data generated** via the diffusion model. The two datasets are compared in terms of localization accuracy to evaluate whether synthetic data can effectively replace real-world measurements in a localization system.

KNN training and prediction

- **Model Training:** The KNN model is trained on both the **real** and **synthetic** RSSI data. The goal is to predict the **X and Y location coordinates** based on the RSSI values.

- **Prediction:** Given a new RSSI sample (real or synthetic), the trained KNN model predicts the corresponding X, Y location.

Error Computation (Euclidean Distance)

- **Localization Error:** The error is computed using the **Euclidean distance** between the true location and the predicted location. The formula is:

$$\text{Error} = \sqrt{(X_{\text{true}} - X_{\text{pred}})^2 + (Y_{\text{true}} - Y_{\text{pred}})^2}$$

IMU + WiFi Fusion Model

This project develops and evaluates a hybrid indoor localization system designed to operate effectively under simulated Android Wi-Fi scan throttling constraints. The core approach fuses continuous relative positioning estimates derived from smartphone Inertial Measurement Unit (IMU) data using a deep learning model (CNN+LSTM) with periodic absolute position corrections obtained from Wi-Fi fingerprinting using a K-Nearest Neighbors (kNN) algorithm. The system processes sensor data sequentially, simulating an online scenario where Wi-Fi updates are intentionally delayed to mimic OS-level throttling, and evaluates the trajectory estimation accuracy against ground truth data.

2. Dataset: ISSAI IMUWiFine

This project utilizes the publicly available IMUWiFine dataset (Nurpeiissov et al., 2022b), accessible via GitHub at https://github.com/IS2AI/IMUWiFine and described in the paper "End-to-End Sequential Indoor Localization Using Smartphone Inertial Sensors and WiFi" (Nurpeiissov et al., 2022a).

The IMUWiFine dataset comprises IMU (accelerometer, gyroscope, magnetometer) and Wi-Fi RSSI data readings recorded in sequential order with a fine spatiotemporal resolution. This sequential recording is crucial for developing and testing time-series based localization algorithms like ours. The data was collected on the fourth, fifth, and sixth floors of the C4 building at the Nazarbayev University campus, covering a total area of over 9,000 m² across these three floors.

Trajectories: A total of 120 distinct trajectories (60 Train, 30 Validation, 30 Test).

Samples: Over 5.4 million time-synchronized sensor readings (3.1M Train, 1.2M Validation, 1.1M Test).

Coverage: The trajectories cover an aggregate distance of 14.2 kilometers (8.0 km Train, 3.1 km Validation, 3.0 km Test).

Duration: The total recording duration is 10.4 hours (5.5 hrs Train, 2.5 hrs Validation, 2.4 hrs Test).

The dataset's structure, including sequential IMU, Wi-Fi, and high-resolution ground truth position data, makes it highly suitable for developing and evaluating hybrid indoor localization systems designed to handle sequential estimation and real-world complexities.

3. Data Preprocessing Workflow

Raw data from the IMUWiFine dataset required significant preprocessing to prepare it for model training and evaluation. The preprocessing pipeline involved the following steps applied to the combined raw data from the designated training and validation directories:

1. Loading and Merging: All relevant DATA_*.csv files were loaded. Files with insufficient columns were skipped. Data from all valid files was concatenated into a single DataFrame.
2. Timestamp Sorting: The combined DataFrame was sorted chronologically based on the timestamp column to ensure sequential integrity crucial for time-series processing and splitting. Rows with invalid

timestamps were dropped.

3. Feature Deduplication (RSSI): (Note: This step was identified in the processing logic but may have been commented out in the final execution) Duplicate rows based only on the RSSI feature columns were potentially removed, keeping the first occurrence, to reduce redundancy.

4. Coordinate Deduplication: To handle static moments or very slow movement, rows with identical (x, y, z) coordinates, rounded to 2 decimal places, were deduplicated, keeping the first occurrence. 5. Sequential Train/Validation Split: The sorted, deduplicated data was split into final training and validation sets using a sequential 80/20 ratio (4:1). This ensures that the validation set contains data chronologically later than the training set, simulating a more realistic deployment scenario.

6. Distance-Based Filtering: Both the training and validation splits were filtered separately. A point was kept only if its 3D Euclidean distance from the previously kept point in that split was greater than or equal to 0.5 meters. This further reduces data density, focusing on points where significant movement occurred.

7. NaN Filling (Features Only): Missing RSSI values (NaNs) within the feature columns of both the filtered training and validation sets were filled with a constant value of 0.0.

8. Feature Scaling (Wi-Fi RSSI): A MinMaxScaler was used to scale the filled RSSI features to a range of [0, 1]. The scaler was fit only on the filtered, filled training data and then used to transform both the training and validation data.

9. Final Reconstruction and Saving: The final processed training and validation DataFrames were constructed, containing the scaled feature columns and the essential meta/target columns ('timestamp', 'x', 'y', 'z'), and saved as separate CSV files.

Figure 3. Data preprocessing of ipin logs to csv format

4. Wi-Fi Localization Component (kNN Radio Map)

Radio Map Construction:

1. A RadioMapWifi class was utilized to encapsulate the kNN logic.
2. To collect the reference point database for fingerprinting, all of the available tracks were utilized to extract unique dataset entries by rssi values and ground truth position rounded up to centimeters. Next to lower the dataFrame size and boost kNN model performance the points that have euclidean distance less than 0.5 meters were removed. Lastly, dataFrame was divided for train and test in 1 to 1 split, resulting in minimal distance between reference points in the database as 1 meter.
3. The RadioMapWifi instance was configured as class with two kNN models to find user location: a. First regression model with k=36 neighbors, weight strategy as distance and using manhattan metric to find X and Y coordinates.
   b. Second classifier model with k=14 neighbors, weight strategy as uniform and using manhattan metric, to find Z coordinate by getting the mode value of k neighbors.
4. The build method trained the kNN model using the sampled, preprocessed fingerprints (X) and their corresponding (x, y) coordinates (y), ensuring consistent AP column order.
5. The fitted RadioMapWifi object was saved to a persistent file using joblib.

Online Localization:

1. During the online simulation, the saved RadioMapWifi object is loaded.

2. When a Wi-Fi update is triggered, the current Wi-Fi RSSI scan is passed to the radio_map.find_position() method.
3. This method uses the kNN model to estimate (x, y, z) coordinates and a confidence score based on the input RSSI vector.

5. IMU Localization Component (CNN+LSTM Displacement Model)

Model Architecture: A deep learning model combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) units was employed. This architecture, instantiated as ModelIMUDisplacement, processes sequences of IMU features (accelerometer, gyroscope, magnetometer - 9 features) to predict the 3D displacement (Δx, Δy, Δz) over a defined time window.

Training:

- The model was trained using all available training data files.
- An internal 80/20 train/validation split was performed on these files using a fixed random seed (IMU_SPLIT_SEED = 42) solely for tuning and selecting the best IMU model during its training phase. ● A custom data loader (DataLoaderHybrid) generated batches of sequential IMU data windows (5 seconds at 20 Hz = 100 samples) and their corresponding ground truth displacement targets.
- Training occurred over 20 epochs using Mean Squared Error (MSE) loss, the AdamW optimizer, and a ReduceLROnPlateau learning rate scheduler.
- The best model state dictionary based on the internal validation loss was saved.

Online Prediction:

- The trained IMU model is loaded for the simulation.
- Sequential windows of IMU data are fed into the model to predict the 3D displacement (delta_pos_imu) for each window, maintaining the LSTM hidden state between windows.
- The current estimated position is updated by adding this predicted displacement.

6. Data Splitting Strategy Summary

Two distinct data splitting procedures were used:

Main Sequential Split: Applied after merging and initial preprocessing of all raw data. This created the final 80% training and 20% validation sets used for radio map building and overall system evaluation, ensuring chronological separation.

Internal IMU Training Split: Applied only to the files designated for training (i.e., the 80% from the main split). This random 80/20 split within the training data was used exclusively during the IMU model's training loop for model selection and hyperparameter tuning.

7. Fusion Strategy and Throttling Handling

1. Initialization: The simulation starts using the ground truth position from the beginning of the test trajectory.
2. IMU Updates: The IMU model continuously updates the estimated position based on predicted displacement for each time window.
3. Throttling Simulation: A timer enforces a minimum interval (WIFI_THROTTLE_MILLISECONDS = 30000.0) between Wi-Fi localization attempts.
4. Wi-Fi Correction Attempt: When the throttle interval is met, the system retrieves the Wi-Fi RSSI scan for the current timestamp and uses the kNN radio map to generate a position estimate and confidence score. 5. Fusion Execution: If the Wi-Fi estimate's confidence meets a predefined threshold (WIFI_CONFIDENCE_THRESHOLD = 0.0005), the current position estimate is directly replaced by the

Wi-Fi estimate (x, y coordinates are replaced; z coordinate from IMU is retained). The Wi-Fi timer is reset. If the confidence threshold is not met, or if Wi-Fi data lookup fails, the system continues relying on the IMU estimate.

6. Logging: Ground truth position, estimated position, timestamp, and the outcome of the fusion step ('init', 'imu_only', 'wifi_direct_replace', etc.) are recorded for subsequent evaluation.

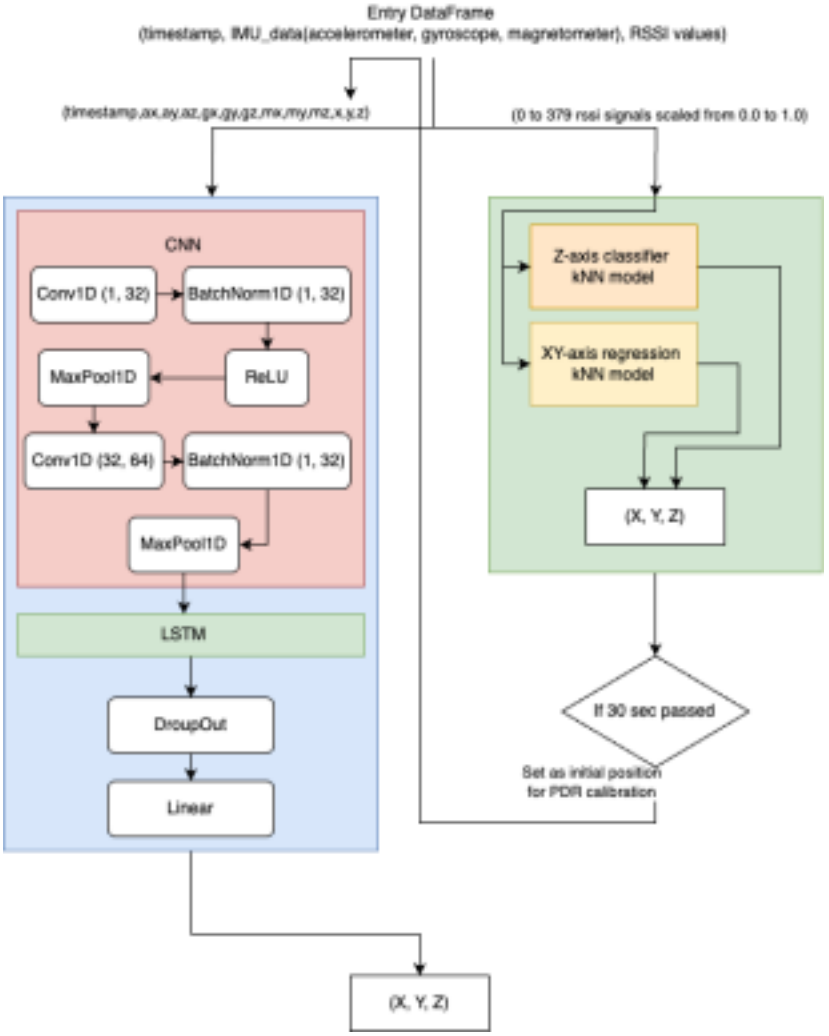

Figure 4. PDR + Wi-Fi localization model scheme

8. Evaluation

Error Calculation: The 2D Euclidean distance error between the final estimated (x, y, z) and ground truth (x, y, z) was calculated for each point in the logged simulation results.

Metrics: Standard localization metrics were computed from the distribution of these 3D errors, including MAE, RMSE, Median, Standard Deviation, Max Error, and percentiles (75th, 90th, 95th).

Project Execution (15%)

• Describe what happened over the course of the last two semesters, including design decisions made, changes in the project, what went wrong/right, how you dealt with the problems you encountered, etc.

• Highlight teamwork aspects, such as how responsibilities were divided, collaborative problem-solving strategies, and leadership roles taken during the project.s

Over the course of the two semesters, the trajectory of our senior project evolved significantly, shaped by deep theoretical exploration, self-driven learning, and ongoing technical experimentation. None of the team members had prior hands-on experience with wireless sensing, indoor localization, or generative models at the outset. As such, we invested substantial time and effort in understanding the underlying concepts by reading academic papers, analyzing code repositories, and replicating existing methods. Across both semesters, our team collectively reviewed and discussed nearly **100 research papers**, primarily sourced from IEEE Xplore and other peer-reviewed databases, forming the backbone of our theoretical foundation.

Initially, the project was titled **"Wireless Sensing Applications Using Wireless Signals and ML Algorithms"**, with the intention to work on **human activity recognition using CSI or RSSI data**. This idea emerged from early literature we encountered and seemed promising. However, as we delved deeper into the domain, we realized that acquiring CSI data was non-trivial—it often required specialized hardware and modified drivers. Moreover, the classification pipelines for activity recognition involved significant signal processing knowledge that was beyond our current scope. These realizations did not come abruptly; rather, they were the result of thorough exploration of prior work and technical documentation.

Building on this understanding, we shifted our focus to **indoor positioning systems (IPS)** using **Wi-Fi Round Trip Time (RTT)**. This method appeared theoretically sound and well-supported in recent literature. However, as we analyzed deployment requirements more closely, we discovered practical constraints: RTT localization depends on devices supporting the IEEE 802.11mc protocol, available only in newer routers and Android 9.0+ smartphones. Our own hardware did not meet these specifications, and obtaining compliant devices was not feasible within our resources. This again was not a spontaneous decision but one made after multiple advisor discussions and weeks of study into RTT localization mechanisms.

With the start of the second semester, we began exploring **Wi-Fi fingerprinting**, particularly focusing on synthetic data generation methods to address limitations in real-world data collection. Android devices impose a restriction of **4 Wi-Fi scans per 2 minutes** starting from Android 8.0, which significantly hinders dense data acquisition. To deal with the multiple facets of this challenge, we split the team into two focused subgroups:

- **Data Generation Subgroup** (Abylay Sydykov & Imangali Zhumangali)
- **PDR and Android Data Collection Subgroup** (*Igilikov Alan & Kubeyev Abylay*)

The PDR and Android Data Collection Subgroup's core technical contribution involved developing the IMU-based localization component using deep learning and integrating it into the hybrid system simulation.

IMU Displacement Model (CNN+LSTM):

Instead of traditional PDR algorithms relying on explicit step detection, length estimation, and heading

integration, a deep learning approach was adopted to directly predict user displacement from IMU sensor data.

Input: The model processes sequential windows of 9-axis IMU data (3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer). Each window represented a fixed time duration (5 seconds, sampled at 20 Hz, resulting in 100 data points per window). Timestamps and ground truth positions associated with these windows were used during training.

Architecture: A combined CNN+LSTM architecture was employed:

- CNN Layers: These layers acted as feature extractors, processing the input window of multi-axis sensor data. Their role was to automatically learn relevant spatial patterns and correlations within the sensor readings indicative of motion dynamics (capturing aspects related to steps, turns, etc., within that 5-second interval).
- LSTM Layers: The sequence of feature vectors extracted by the CNN layers across consecutive time windows was fed into LSTM layers. The LSTMs captured the temporal dependencies and sequential nature of human movement, integrating information over time.

Output: The final layers of the network predicted the net 3D displacement vector (Δx, Δy, Δz) representing the change in the user's position over the duration of the input window.

Training: The model was trained end-to-end using the IMUWiFine dataset. The input was the windowed IMU data, and the target was the ground truth displacement calculated between the start and end ground truth positions corresponding to that window. Mean Squared Error (MSE) was used as the loss function to minimize the difference between predicted and actual displacement.

This deep learning model effectively learns a complex function mapping raw sensor sequences to relative motion, replacing the need for hand-crafted PDR heuristics.

Hybrid Fusion and Throttling Simulation:

The online simulation integrated the trained IMU displacement model with the kNN Wi-Fi radio map, simulating the 30-second throttling constraint:

1. Continuous IMU Tracking: The simulation processed the test trajectory data window by window. In each step, the IMU model predicted the displacement, which was added to the current estimated position, providing a continuous path estimate.
2. Throttled Wi-Fi Corrections: A timer tracked the elapsed time since the last Wi-Fi correction. This timer was checked periodically at the boundaries of the 5-second IMU processing windows.
3. Wi-Fi Localization Trigger: If the check revealed that 30 seconds or more had passed since the last correction, the system attempted Wi-Fi localization using the kNN radio map with the RSSI data corresponding to the current timestamp.
4. Position Reset: If the kNN localization was successful and met a predefined confidence threshold (0.0005), the current estimated position (derived from accumulated IMU displacements) was entirely replaced by the absolute (x, y) coordinates from the Wi-Fi estimate (the z-coordinate was retained from the IMU estimate). The Wi-Fi timer was then reset. If the Wi-Fi attempt failed or did not meet the confidence threshold, the system continued relying solely on the IMU estimate for that step.

Challenges:

IMU Model Accuracy and Tuning: A significant challenge was achieving high accuracy with the PDR component alone. Initial results showed a high Root Mean Square Error (RMSE) for the IMU-based trajectory estimates. This necessitated considerable effort in hyperparameter tuning for the CNN+LSTM model using Optune study method. Iterative experimentation was required with learning rates, optimizer parameters (AdamW decay), network architecture (number/size of CNN and LSTM layers, activation functions), and potentially the input window size and stride to minimize the displacement prediction error and thus reduce the overall PDR drift.

Infrequent Wi-Fi Fusion Triggers: The simulation was designed to mimic a 30-second Wi-Fi throttle. Given the average length of trajectories in the validation set, it was anticipated that roughly 10 Wi-Fi localization attempts would occur per trajectory. However, the actual execution resulted in only 3-5 Wi-Fi correction events per trajectory. The root cause was identified in the implementation: the check for the 30-second interval was performed only at the start or end of processing each 5-second IMU window. If the 30-second threshold was crossed mid-window, the Wi-Fi localization trigger was missed until the check occurred at the boundary of the next window. This effectively quantized the trigger points to 5-second intervals, preventing Wi-Fi updates from occurring as frequently as the 30-second timer would theoretically allow, thus limiting the opportunities to correct PDR drift.

Potential Solutions and Future Directions:

IMU Model Enhancement: Continued fine-tuning of the CNN+LSTM model's hyperparameters is essential. Exploring alternative architectures (e.g., different CNN structures, attention mechanisms, bidirectional LSTMs) or incorporating more sophisticated preprocessing or feature engineering for the IMU data could further reduce the inherent error in the displacement predictions.

Improved Fusion Algorithm: The current "direct replace" fusion strategy is simple but potentially suboptimal, especially if Wi-Fi estimates are occasionally noisy despite passing the confidence threshold. Furthermore, the timing issue needs addressing:

- Decoupled Timer Logic: The Wi-Fi trigger check should be decoupled from the fixed IMU window processing loop. A more precise timer mechanism could trigger the Wi-Fi localization attempt immediately when the 30-second interval is reached, regardless of the IMU window state. This would likely increase the frequency of Wi-Fi corrections closer to the theoretical expectation.
- Advanced Fusion Methods: Replacing the direct position reset with a more robust fusion algorithm like an Extended Kalman Filter (EKF) could yield better results. The IMU model would provide the state prediction (position and potentially velocity, along with an estimated covariance), while the kNN Wi-Fi result would serve as the measurement update. This allows for a weighted combination of the two estimates based on their respective uncertainties, rather than discarding the IMU path information entirely during a Wi-Fi update.
- Adaptive Weighting: A simpler alternative to EKF could involve dynamically weighting the IMU and Wi-Fi estimates based on the Wi-Fi confidence score or other quality indicators (e.g., number of visible APs, consistency checks).

The **data generation team** initially explored **Generative Adversarial Networks (GANs)** to synthesize Wi-Fi RSSI fingerprints. GANs are known for their ability to learn complex data distributions, but their training dynamics are notoriously unstable. Our experimentation with GANs revealed several technical challenges: models were slow to converge, hyperparameters were highly sensitive, and the output fingerprints lacked physical plausibility. Training was also computationally demanding—our laptops often overheated or lagged due to the intensive model runs, limiting our ability to iterate quickly.

In response to these obstacles, we shifted to **table diffusion models**, inspired by recent advancements in generative modeling for tabular data. Diffusion models exhibited greater stability and consistently produced higher-quality synthetic data. However, adopting this approach required a deep understanding of **noise schedules, learning rates, reverse sampling, and denoising processes**. Without formal instruction in these models, we relied on extensive literature reviews and community-supported resources (e.g., GitHub projects, blog posts, tutorials) to master these techniques and fine-tune our implementations.

To validate our synthetic data generation, we used a **public Wi-Fi fingerprint dataset** from IEEE DataPort, collected by an autonomous robot equipped with LIDAR, IMU, RGB-D cameras, and sonar. The dataset features RSSI values from 6 access points (APs) over multiple days within a 21×16 meter indoor space, and provides high-accuracy ground truth positions (±0.07 m). This dataset served as our experimental baseline for training and evaluating generative models.

To overcome the instability of GANs, we transitioned to **diffusion models**, specifically building our own **conditional table diffusion model in PyTorch**. This model was designed to learn a generative process over tabular RSSI data conditioned on (X, Y) coordinates. The architecture included a **sinusoidal time embedding module** and a deep feedforward network that maps the noised RSS input, spatial condition, and time embedding to a denoised output. The final model was implemented from scratch using core PyTorch modules and techniques, including:

- **torch.nn** for defining the model architecture and activations
- Custom sinusoidal time-step embedding logic using math and torch
- **scikit-learn** modules for preprocessing and evaluation (StandardScaler, train_test_split, mean_squared_error)
- **matplotlib.pyplot** for visualizing training loss and data distributions

The following table summarizes the primary tools and frameworks used during project execution:

Category	**Tools / Frameworks**
Programming Language	Python
Data Analysis	NumPy, Pandas, SciPy
ML & Deep Learning	PyTorch, Keras, TensorFlow (used in PDR CNN/LSTM pipeline)
Generative Modeling	Custom PyTorch-based Table Diffusion Model with sinusoidal time embeddings
Visualization	Matplotlib, Seaborn, t-SNE
Version Control	Git, GitHub

Communication Telegram, Google Meet (weekly advisor meetings)

Project Management Notion

Collaboration & File Sharing Google Drive, GitHub

To manage work and ensure progress, we followed weekly development cycles. Meetings with our advisor were held via **Google Meet** and **in person**, during which we reported on progress, asked questions, and discussed strategic pivots. **Telegram** served as our primary daily communication platform, while project documentation, timelines, and task tracking were organized in **Notion**, which we adopted as our project management tool.

Throughout the project, decision-making was **collaborative and research-driven**, grounded in academic inquiry rather than assumption. The team developed critical thinking skills through repeated cycles of hypothesis formation, model implementation, evaluation, and feedback.

Evaluation (20%)

The KNN model and the Mean Euclidean Distance from its output was used as an evaluation metric of the generative models output.

For example below is an example of training of the generative model using only 10% the overall reference points and 90% of the dataset was cropped for testing of the localization accuracy later. In Fig. 3, all 34 reference points (10% of the original dataset) can be seen. It can be clearly seen visually that these points sufficiently differ from the original dataset and slightly give its pattern.

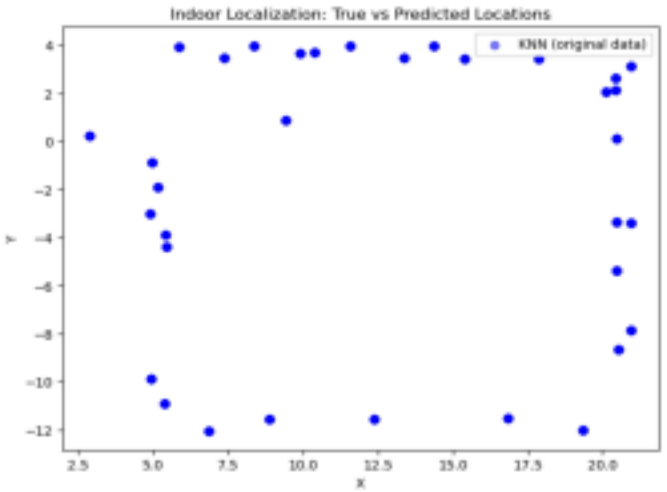

Figure 4. 10% of original dataset used for training

Below in Figure 7., the result of the final fingerprinting map that was generated with only usage of synthetic data that was generated via Diffusional model that was trained only on the 10% of the data as visualized above. It can be visually assessed that the generative model correctly finds patterns between RSSI signals data and can provide accurate results.

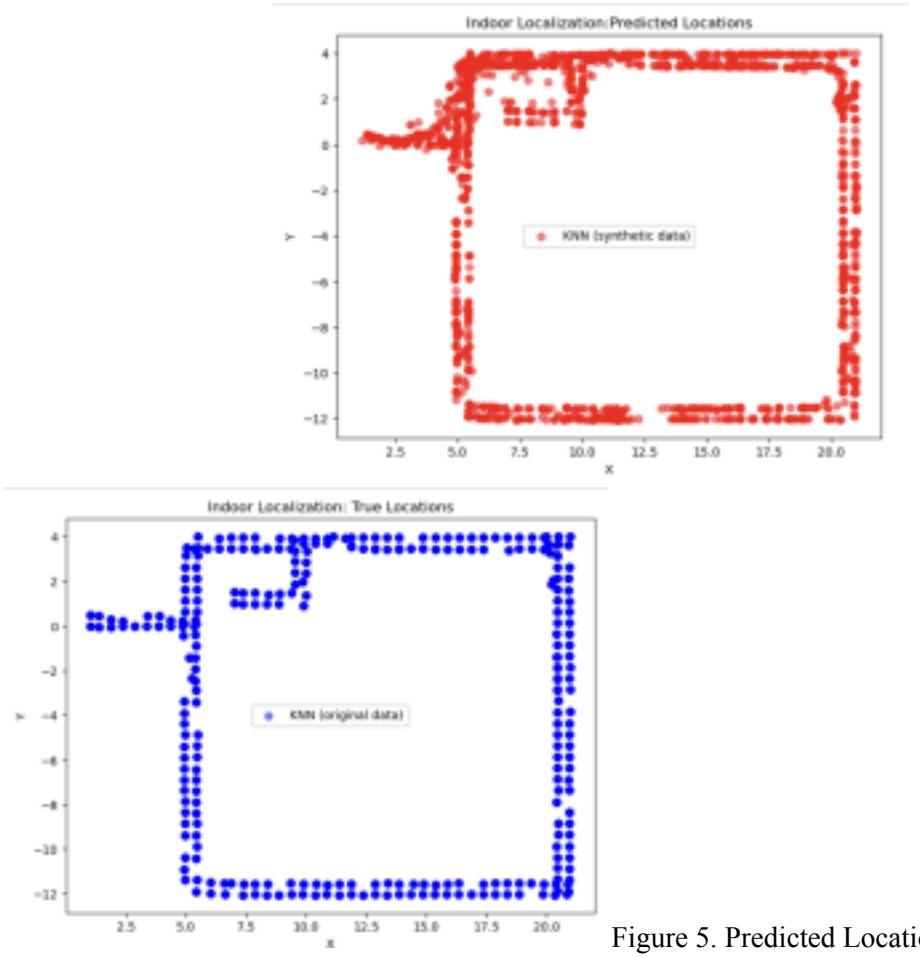

Figure 5. Predicted Locations and True Location

Table 1. contains experiment results based on training of diffusion based models on different proportions of original data and final results. It can be seen that on average, Mean Euclidean Error from combined dataset is 15-20 % lower than using only small original dataset.

% of real data DDPM trained on.	KNN localization Mean Euclidean Error for original + synthetic dataset (m)	KNN localization Mean Euclidean Error for original training dataset for DDPM model (m)	KNN localization Mean Euclidean Error for full original dataset (m)	Improvement in localization error (%)
10%	1.1901	1.5069	0.2082	21.02
20%	1.0549	1.2689	0.2082	16.87
30%	0.9594	1.2345	0.2082	22.28
40%	0.9813	1.1063	0.2082	11.29
50%	0.7505	0.8682	0.2082	13.56
60%	0.7251	0.7317	0.2082	0.90

70%	0.6882	0.6964	0.2082	1.17
80%	0.3447	0.3512	0.2082	1.85
90%	0.2219	0.2224	0.2082	0.22

Table 1. Comparison of model performance based on the percentage of real data diffusion model trained on with results measured as Mean Euclidean Distance (MED) in meters

Trajectory ISSAI Fusion Model (RMSE = 1.12m)	**ISSAI data with Limitation on Our Fusion Model (RMSE = 4.5m)**	**Mean Error Distance (ISSAI Fusion Model)**	**Mean Error Distance (ISSAI data with Limitation on Our Fusion Model)**
T1	1.0574 m	4.2321 m	1.0124 m 4.4601 m
T2	1.1550 m	5.0320 m	1.2294 m 4.6095 m
T3	1.1034 m	4.8233 m	1.0415 m 4.3186 m
T4	1.2069 m	4.3924 m	1.1435 m 4.5986 m
T5	1.0520 m	4.6586 m	1.0825 m 4.7477 m
T6	1.0893 m	4.905 m	1.0386 m 4.5277 m
T7	1.1332 m	4.7293 m	1.1308 m 4.4589 m
T8	1.0790 m	4.5293 m	1.1276 m 4.6802 m
T9	1.1285 m	4.4484 m	1.04695 m 4.3461 m
T10	1.0959 m	4.8352 m	1.1236 m 4.5299 m
T11	1.1623 m	4.626 m	1.2037 m 4.8234 m
T12	1.1092 m	4.3647 m	1.1363 m 4.5256 m

Table 2. RMSE and Mean Error Distance metrics values for ISSAI Fusion Model and for our Fusion Model for each trajectory in the validation

Table 2 clearly shows a significant increase in localization error (overall RMSE increasing from 1.12m to 4.5m, and MED increasing substantially for each trajectory) when our fusion model operates under the simulated 30-second Wi-Fi throttling constraint compared to the baseline ISSAI model performance. This quantitatively demonstrates the substantial negative impact of infrequent Wi-Fi updates, which was the core problem identified. While the accuracy is reduced, the system does produce continuous localization estimates throughout the trajectories, validating that the fusion mechanism is operational. The error metrics reflect the accumulated PDR drift between the sparse Wi-Fi corrections.

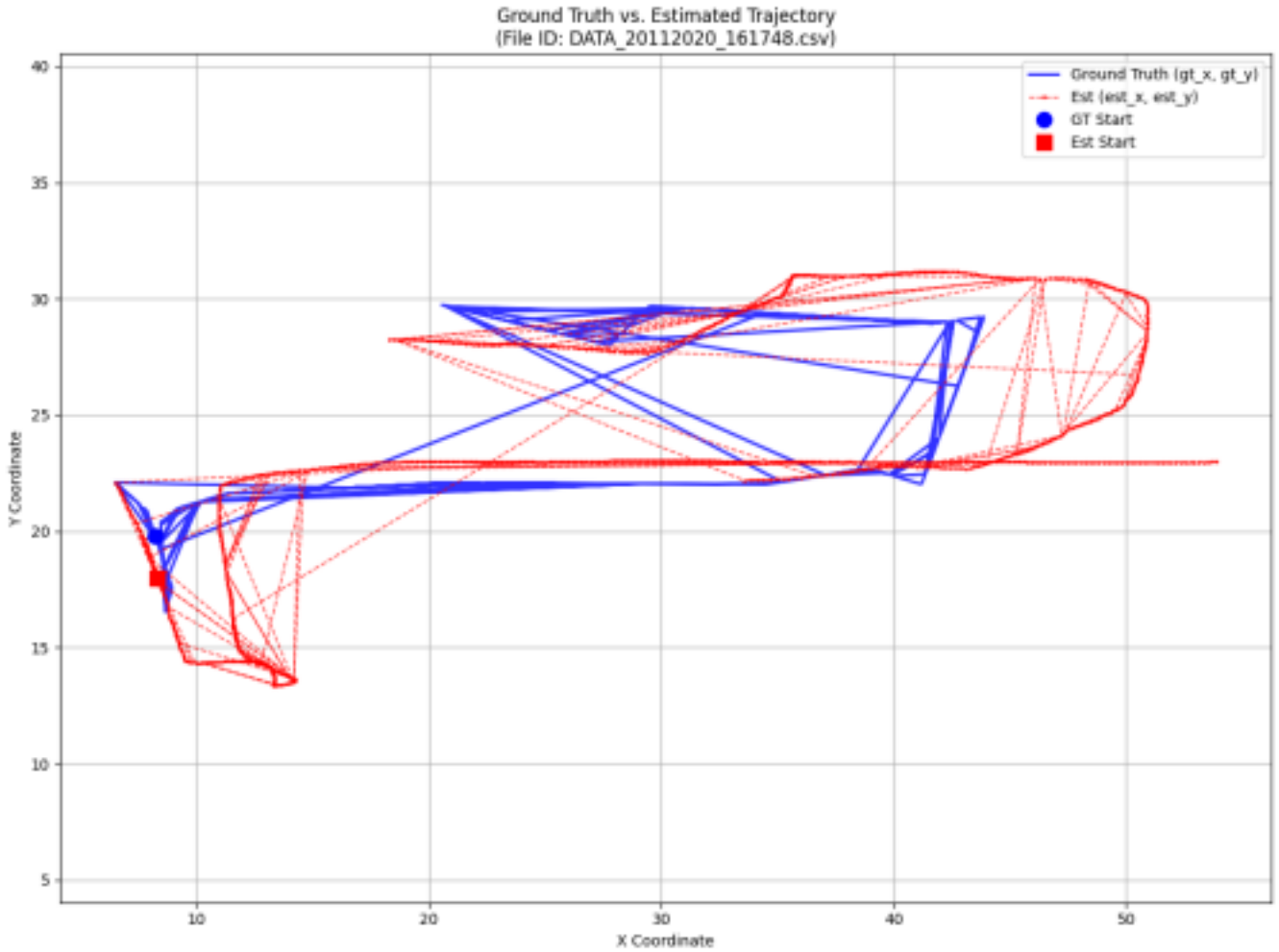

Figure 6. GT Trajectory Plot vs Estimated Trajectory Plot

The plot of Ground Truth vs. Estimated Trajectory for file DATA_20112020_161748.csv visually corroborates the quantitative findings. The estimated trajectory (red, dashed line) attempts to follow the ground truth (blue, solid line). However, significant deviations are visible where the red line drifts away from the blue line. These periods of drift represent the system relying solely on the CNN+LSTM PDR component (imu_only steps in the logged CSV data). The sharp, sometimes erratic corrections or jumps in the red path indicate moments where a Wi-Fi update (wifi_direct_replace step) likely occurred, attempting to reset the position estimate closer to the ground truth. The plot clearly illustrates the challenge of PDR drift accumulation and the necessity, yet infrequency, of Wi-Fi corrections under the simulated throttling, validating the behavior of our computing-based fusion approach in the face of this constraint.

The evaluation process validates the computing-based solution in several ways:

Problem Simulation: The simulation framework directly models the core challenge – Wi-Fi scan throttling – by enforcing a minimum time interval between Wi-Fi localization attempts.

Component Integration: It tests the integration and interplay of the distinct computing components: the deep learning model (CNN+LSTM) for PDR displacement estimation, the machine learning algorithm (kNN) for Wi-Fi fingerprinting, and the rule-based fusion logic incorporating the throttle timer and confidence threshold.

Performance Quantification: The use of standard metrics (RMSE, MED) against ground truth provides objective measures of the system's accuracy under the specific constraints it was designed for.

Behavioral Visualization: The trajectory plots provide qualitative evidence of the system's dynamic behavior, showing both the continuous estimation capability (via PDR) and the drift-correction mechanism (via Wi-Fi fusion).

The evaluation demonstrates that the developed hybrid fusion model partially addresses the problem of indoor localization under Wi-Fi scan throttling mentioned in the introduction. The system successfully integrates PDR and Wi-Fi fingerprinting to provide continuous tracking, preventing complete localization failure between sparse Wi-Fi updates. However, the quantitative results (Table 2) and qualitative plots (Figure 6) clearly indicate that the 30-second throttling interval significantly degrades localization accuracy compared to scenarios with more frequent updates. The PDR component, while providing continuity, accumulates considerable drift over the 30-second intervals, which the infrequent Wi-Fi corrections cannot fully compensate for to achieve high accuracy (meter-level as seen in the baseline). Therefore, while the computing-based fusion approach is validated as functional under throttling, the evaluation highlights that achieving high accuracy remains a significant challenge solely with this strategy under such severe constraints. Further improvements, potentially in PDR accuracy or more sophisticated fusion techniques (as discussed in Potential Solutions and Future Directions), would be necessary to bridge the performance gap.

Conclusion and possible future work (5%)

Conclusion

This senior project makes several key contributions to the field of indoor positioning systems:

- **Innovative use of diffusion models** for generating synthetic Wi-Fi fingerprints, which proved to improve localization accuracy, especially when training data was scarce.
- **Development of a CNN+LSTM-based PDR model** that can predict displacement from raw IMU sensor data, avoiding the need for traditional hand-crafted PDR heuristics.
- **Design of a throttling-aware fusion algorithm**, simulating Android Wi-Fi scan limitations, which allowed continuous localization by combining absolute (Wi-Fi) and relative (IMU) positioning.

Quantitative results confirm that the fusion model maintains localization capability with an RMSE of approximately 4.5 meters under scan throttling, compared to 1.12 meters under ideal conditions. While not optimal, the system demonstrates operational feasibility under real-world constraints and lays a foundation for future enhancements.

Possible Future Work

Several directions can be pursued to improve and extend the current system:

1. **Enhanced Fusion Algorithms**: Implementing Kalman Filters or Particle Filters can enable uncertainty-aware integration of IMU and Wi-Fi data, replacing the current “direct replace” strategy.
2. **Dynamic Wi-Fi Scan Management**: A more precise scan-timing controller, decoupled from IMU windows, could increase Wi-Fi update frequency and reduce PDR drift.

3. **Advanced IMU Models**: Exploring Transformer-based models or attention mechanisms may improve the accuracy of displacement predictions.

4. **Real-World Deployment**: Collecting live data in various indoor settings using commodity smartphones would validate system performance beyond public datasets.

5. **Augmented Datasets**: Expanding synthetic fingerprint datasets using environmental simulation (e.g., ray tracing) could enhance robustness in more complex or multi-floor environments.

6. **Battery and Latency Optimization**: Assessing and optimizing the energy and computational overhead of the hybrid system will be essential for deployment on resource-constrained mobile devices.

This work demonstrates a strong integration of generative modeling, deep learning, and sensor fusion to address a timely and practical systems challenge in indoor localization. It opens pathways toward scalable, real-time, and infrastructure-independent indoor positioning systems.

References (5%)

1. He, S., & Chan, S. H. G. (2016). Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons. IEEE Communications Surveys & Tutorials, 18(1), 466-490.
2. Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33, 6840-6851.
3. Hoang, M. T., Yuen, B., Dong, X., Lu, T., Westendorp, R., & Reddy, K. (2019). Recurrent neural networks for accurate RSSI indoor localization. IEEE Internet of Things Journal, 6(6), 10639-10651. 4. Khalajmehrabadi, A., Gatsis, N., & Akopian, D. (2017). Modern WLAN fingerprinting indoor positioning methods and deployment challenges. IEEE Communications Surveys & Tutorials, 19(3), 1974-2002. 1. Liu, H., Darabi, H., Banerjee, P., & Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(6), 1067-1080.
2. Nabati, M., Ghorashi, S. A., & Shahbazian, R. (2020). WiFi fingerprinting using generative adversarial networks. IET Communications, 14(20), 3750-3755.
3. Yang, L., Chen, Y., Li, X., & Xiao, C. (2022). Diffusion models for wireless communication: Applications, challenges, and opportunities. IEEE Communications Magazine, 60(12), 94-100. 4. Zafari, F., Gkelias, A., & Leung, K. K. (2019). A survey of indoor localization systems and technologies. IEEE Communications Surveys & Tutorials, 21(3), 2568-2599.
5. Zou, H., Lu, X., Jiang, H., & Xie, L. (2016). A fast and precise indoor localization algorithm based on an online sequential extreme learning machine. Sensors, 16(9), 1438.
6. Kim, Y., Kyung, Y., & Ko, H. (2022). Indoor localization system with PDR and WiFi complementary integration. Proceedings of the 13th International Conference on Information and Communication Technology Convergence (ICTC) (pp. 1674-1676). https://doi.org/10.1109/ICTC55196.2022.9952948
7. Luo, Y., Guo, C., Su, J., Guo, W., & Zhang, Q. (2021). Learning-based complex motion patterns recognition for pedestrian dead reckoning. IEEE Sensors Journal, 21(4), 4280-4289. https://doi.org/10.1109/JSEN.2020.3029719
8. Nurpeiissov, M., Kuzdeuov, A., Assylkhanov, A., Khassanov, Y., & Varol, H. A. (2022a). End-to-end sequential indoor localization using smartphone inertial sensors and WiFi. Proceedings of the IEEE/SICE International Symposium on System Integration (SII) (pp. 566-571). https://doi.org/10.1109/SII52469.2022.9708854

9. Nurpeiissov, M., Kuzdeuov, A., Assylkhanov, A., Khassanov, Y., & Varol, H. A. (2022b). IMUWiFine dataset and localization system. Institute of Smart Systems and Artificial Intelligence (ISSAI), Nazarbayev University. GitHub repository. https://github.com/IS2AI/IMUWiFine
10. Shi, L.-F., Wang, Y., Liu, G.-X., Chen, S., Zhao, Y.-L., & Shi, Y.-F. (2018). A fusion algorithm of indoor positioning based on PDR and RSS fingerprint. IEEE Sensors Journal, 18(23), 9691-9698. https://doi.org/10.1109/JSEN.2018.2873052
11. Sun, J., Sun, W., Chen, Z., Zhang, X., & Tang, C. (2023). A large indoor localization system based on WiFi and smartphone inertial sensors. Proceedings of the 42nd Chinese Control Conference (CCC) (pp. 3583-3588).
12. Wang, Q., Li, J., Luo, X., & Chen, C. (2022). Fusion algorithm of WiFi and IMU for indoor positioning. Proceedings of the 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS) (pp. 349-353). https://doi.org/10.1109/ISPDS56360.2022.9874146
13. Yao, Y., Pan, L., Feng, W., Xu, X., Liang, X., & Xu, X. (2020). A robust step detection and stride length estimation for pedestrian dead reckoning using a smartphone. IEEE Sensors Journal, 20(17), 9685-9697. https://doi.org/10.1109/JSEN.2020.2989865
14. Zhang, M., Shen, W., Yao, Z., & Zhu, J. (2016). Multiple information fusion indoor location algorithm based on WIFI and improved PDR. Proceedings of the 35th Chinese Control Conference (CCC) (pp. 5086-5092).
15. Minh Tu Hoang, Xiaodai Dong, Tao Lu, Brosnan Yuen, Robert Westendorp, November 30, 2019, "WiFi RSSI Indoor Localization", IEEE Dataport, doi: https://dx.doi.org/10.21227/1yd5-rn96.
16. Hybrid-pdr-wifi-localization. GitHub repository. 09.05.2025, from https://github.com/smolanpolan/hybrid-pdr-wifi-localization
17. Wifi-fingerprint-diffusion. GitHub repository. 09.05.2025, from https://github.com/ImokAAA/wifi-fingerprint-diffusion