# Dynamic Positioning in Outdoor Cellular Networks: A Hybrid Approach Integrating RSS Fingerprinting and LSTM-Based Trajectory Prediction

Manish Kumar Krishne Gowda, 0033682812

(Spring 2024)

## 1 Abstract

Accurate positioning of mobile users in wireless networks is essential for various location-based applications and services. In this project, I propose a novel approach for mobile positioning using a combination of feedforward neural networks (FFNN) and gated recurrent units (GRU). The FFNN is utilized for static positioning estimation, while the GRU handles the dynamic aspect by incorporating historical trajectory information. I train and evaluate the proposed model using GPS trajectory data collected from the Geolife dataset, overlaying it onto a simulated distributed multiple-input multiple-output system. Through extensive experimentation, I demonstrate the effectiveness of our approach in improving positioning accuracy, especially in scenarios with high shadowing noise. The findings suggest that the integration of FFNN and GRU models offers a promising solution for enhancing mobile positioning accuracy in real-world environments.

## 2 Introduction

Geospatial tracking and positioning systems have become integral components of modern technology, revolutionizing the way we interact with our surroundings and shaping the landscape of location-based services on mobile devices. These systems are crucial for delivering precise and contextually relevant information to users, empowering them to navigate their environments, monitor their movements, access real-time geospatial services, and make informed decisions. By leveraging these systems, users can access a wealth of information tailored to their specific geographic location, enhancing their overall experience and utility of mobile devices. Whether it's finding the nearest restaurant, navigating through unfamiliar streets, or tracking fitness activities, the applications of geospatial tracking and positioning are diverse and far-reaching.

The evolution of wireless communication systems has given rise to many positioning techniques, each addressing the distinct challenges encountered across diverse environments and scenarios. Notably, among these techniques, radio frequency (RF) technologies, radio positioning System, cellular networks, ultrasonic, geomagnetic, and optical systems have emerged as indispensable pillars within the realm of positioning technology. Among these technologies, the Global Positioning System (GPS), a prime example of a radio positioning System, is widely known and found in most modern mobile phones [1]. Many apps rely on this satellite-based positioning technology. However, indoors, GPS signals often weaken, and direct transmission becomes difficult, making traditional GPS less reliable in such settings. As a result, RF positioning systems such as Wi-Fi [2] and Bluetooth [3] have become popular alternatives for indoor positioning. These systems utilize radio frequency signals to determine the location of devices within indoor environments where GPS signals may be unreliable.

Even in outdoor settings, the reliability of GPS faces significant hurdles, particularly in densely urbanized areas. These challenges arise due to the presence of tall buildings, dense vegetation, and the unpredictable cover of moving clouds, resulting in less accurate and reliable GPS data. Interestingly, in locations where cellular infrastructure is abundant, cellular positioning emerges as

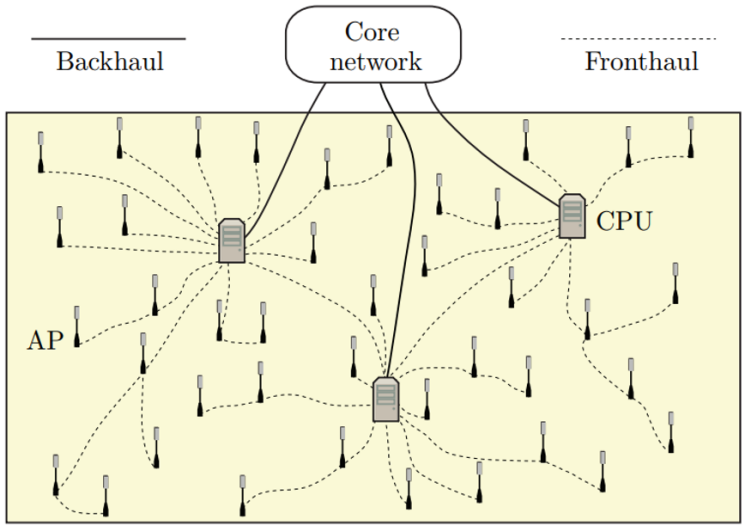


Figure 1: Cell-Free Massive Mimo Network

a strong competitor that could surpass GPS performance. This is particularly evident in areas with densely distributed cellular towers, promising improved accuracy and effectiveness. Moreover, the demands of emergency response systems underscore the importance of reliable positioning even when GPS data is unavailable. In such critical situations, cellular positioning techniques become especially crucial [4].

Discussing densely distributed cellular towers, future wireless networks such as cell-free and distributed multiple-input multiple-output (MIMO) systems [5, 6] feature distributed massive MIMO arrays, offer potential for enhanced cellular positioning. The concept envisions each user being encompassed by small access points (APs) collaborating to ensure consistently high service quality. These systems have demonstrated the ability to leverage the spatial domain of mobile fading channels effectively, resulting in notable performance enhancements for wireless communication systems. For positioning, the presence of multiple antennas strategically dispersed across the landscape provides a crucial advantage. This strategic spatial separation ensures that radio signal measurements collected by individual APs vary significantly for a given location. These differences in measurements can be utilized to conduct positioning in these systems. A depiction of a cell-free network is presented in the Figure 1.

In such distributed MIMO systems, received signal strength (RSS)-based fingerprinting emerges as a strong candidate for positioning [7, 8, 9, 10]. Each dedicated AP calculates the RSS information of the user equipment (UE) at specific geographical points, called reference points (RPs) within the network area. In fingerprint-based positioning systems, RPs serve as known locations within the environment with associated fingerprint data, comprising signal strength measurements from various wireless APs or positioning signals. These RPs act as anchors for the system, providing a basis for comparison and aiding in the estimation of the location of other points within the environment. On the other hand, test points (TPs) are locations where the system aims to estimate or predict its position based on the collected fingerprint data. Unlike RPs, TPs may not have previously recorded fingerprint information (although for simulations they are collected to infer accuracy). Instead, the system collects real-time signal measurements at TPs and compares them with the fingerprint database to infer their locations. Test points are instrumental in evaluating the accuracy and performance of the positioning system, helping to validate the effectiveness of the algorithm and refine system parameters to enhance positioning accuracy. The RSS information

from the RPs is stored within a central processor, coordinating with all the APs. This repository of RSS data serves as a robust fingerprint for precise location estimation. The RSS readings from all APs at an unknown location are acquired and compared against the stored fingerprints, enabling accurate positioning and localization in these outdoor cellular networks.

Most positioning techniques primarily target static UE within the network area, those aimed at moving UEs often rely heavily on GPS data. Considerable research has been dedicated to enhancing GPS accuracy in dynamic scenarios, particularly in driving contexts, employing methods such as assisted-GPS, differential GPS and sensor fusion. For instance, [11] introduces a novel data fusion approach that combines Kalman filter, gradient boosting decision tree, and particle swarm optimization to refine the accuracy of vehicle navigation systems. Additionally, [12] explores the integration of GPS and Inertial navigation system data using artificial neural networks (ANNs) to sustain precise positioning even in GPS signal blackouts. Furthermore, [13] focuses on urban navigation by accounting for sensor quality and driving context, enabling navigation systems to adapt to varying conditions and notably enhancing the accuracy and reliability of urban navigation.

Research is also underway to predict the positions of mobile users over time, with many approaches relying on GPS data. For a comprehensive overview of various techniques, applications, and challenges in next location prediction, [14] offers valuable insights. Numerous prediction methods are based on recurrent neural network (RNN) architectures, particularly variants of long short-term memory (LSTM) networks [15, 16, 17, 18]. For instance, [15] introduces a spatio-temporal position prediction model for mobile users using LSTM networks, emphasizing the capture of both spatial and temporal dependencies in user trajectories. Additionally, [17] presents a bidirectional LSTM model with an attention mechanism for predicting vehicle destinations. Moreover, the gated recurrent unit (GRU) model, another variant of RNN, is employed in [19] and [20] to effectively utilize RSS data from network sources and enhance the accuracy of mobile positioning systems. Furthermore, [21] utilizes transformer-based models to learn travel mode patterns and behaviors, thereby improving the accuracy of predicting users' next locations in geographic information systems. In contrast, [22] employs bidirectional convolutional RNNs, while [23] utilizes an XGBoost classifier for the same predictive task. Trajectory prediction, as addressed in these papers, refers to the task of forecasting the future movement paths of mobile users or vehicles based on historical data, such as GPS trajectories. This predictive capability is valuable in various applications, including transportation planning, urban mobility management, and location-based services.

Returning to RSS fingerprint-based positioning systems, it's notable that existing outdoor RSS fingerprinting solutions predominantly focus on static positioning. However, they often fail to capitalize on the valuable historical trajectory data of the UE. In this paper, I advocate for a novel approach that harnesses the past positional information acquired through the RSS fingerprinting method, utilizing it to estimate the current location of the UE. The proposed framework involves two interconnected models: the first is a traditional model that employs stored RSS fingerprints to offer a static location estimate, while the second model integrates past estimated locations from the initial model to provide an enhanced and more accurate estimation of the current UE location. The overarching goal is to significantly improve positioning accuracy, particularly in mobile UE scenarios. This innovative methodology incorporates tracking techniques that capitalize on short-term historical data, thereby addressing the dynamic nature of user mobility within outdoor cellular networks.

The proposal introduces an innovative research concept centered around the real-time position tracking of a UE within an outdoor distributed and cell-free MIMO environment. The methodology leverages two distinct deep learning models: the first, a straightforward feed-forward neural network (FFNN), is dedicated to training the fingerprint using RSS data, ultimately generating the static 2D positioning coordinates of the UE. The second model employs GRU for processing user

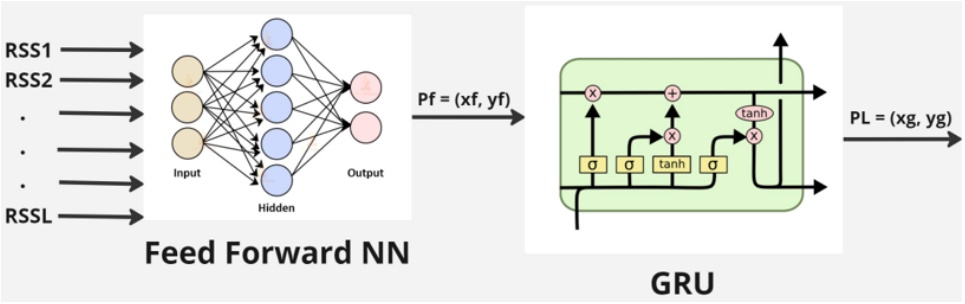


Figure 2: Proposed Model

location information alongside timestamps corresponding to different like walking and driving. This recurrent neural network adjusts the location estimates of the FFNN by incorporating historical movement data, thereby addressing the dynamic positioning aspect

The system operates within defined time intervals, during which the UE requests positioning information from a central entity. Within these intervals, RSS data from APs is utilized for positioning estimation. The initial stage involves a FFNN, which leverages the RSS data to estimate the position coordinates of the unknown location. The FFNN does not inherently consider the dynamic movement scenario of the UE. To address this limitation, the predicted location from the FFNN serves as input to the GRU network. The GRU is trained using both the FFNN output and GPS movement data. By incorporating this additional information, the GRU can capture the dynamics of the UE's movement, including parameters such as speed, distance traveled, acceleration, and movement conditions.

The role of the GRU is to refine the initial estimates provided by the FFNN. By analyzing how the FFNN output evolves over time, the GRU continuously updates its estimation to provide a more accurate and reliable output of the 2D cordinate estimates of the FFNN. This mutually assistive process allows the system to adapt to changing movement patterns and environmental conditions, ultimately improving the overall positioning accuracy for user mobility scenarios.The proposed model is shown in Figure 2.

## 3 System Model

The system model involves a distributed MIMO system configuration, comprising $L$ remote radio units (RRUs), each equipped with a single antenna. These RRUs function as APs for positioning purposes. In this collaborative setup, all RRUs synchronize their operations to serve K UEs within the network's coverage area. It is assumed that all the $L$ RRUs co-serve the $K$ UEs by joint coherent transmission and reception. he connectivity between RRUs is established through fronthaul links, also connecting them to edge-cloud processors referred to as central processing unit(s) (CPU). The CPUs coordinate the operation of RRUs within the distributed MIMO system.

The uplink channel from UE $k$ to RRU $\ell$, denoted by $\mathbf{h}_{k\ell} \in \mathbb{C}$ is modeled using a coherence block fading model, wherein the channel coefficients are assumed remain constant for a duration corresponding to $\tau_c$ complex samples. The received combined uplink signal $\mathbf{Y}_\ell \in \mathbb{C}^{1\times\tau_p}$ at RRU $\ell$ from all $K$ UEs is given by

$$\mathbf{Y}_\ell = \sum_{k=1}^{K} \sqrt{\rho_k} h_{k\ell} \boldsymbol{\psi}_k^T + \mathbf{N}_\ell \tag{1}$$

where $\rho_k$ is the transmit power of UE $k$, $\mathbf{N}_\ell \in \mathbb{C}^{1\times\tau_p}$ is the combined noise at RRU $\ell$ with i.i.d elements distributed as $\mathcal{N}_\mathcal{C}(0,\sigma_n^2)$ and $\boldsymbol{\psi}_k \in \mathbb{C}^{\tau_p}$, $k = \{1, 2, \ \dots \ , K\}$ are the mutually orthogonal pilot vectors assigned to UEs when they gain access to the network i.e.

$$\boldsymbol{\psi}_i^H \boldsymbol{\psi}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{2}$$

and $h_{k\ell}$ is characterized using an uncorrelated Rayleigh fading model in each coherence block i.e.

$$h_{k\ell} \sim \mathcal{N}_\mathbb{C}(0, \beta_{k\ell}) \tag{3}$$

For $\beta_{k\ell}$, the large-scale fading coefficient, a log-distance path loss model is adopted, specifically.

$$\beta_{k\ell}[dB] = -28.8 - 35.3 \log_{10}\left(\frac{d_{k\ell}}{1m}\right) + \nu \tag{4}$$

This is derived from the 3GPP 38.901 urban micro street canyon path loss model, defined in [24]. This equation is valid for an operating frequency of 2GHz, with the RRU positioned at a height of 10m and the UE at a height of 1.5m. For the purpose of positioning it is assumed that ($K \leq \tau_p$) so that there is no interference between UEs due to pilot contamination. The signal $y_{k\ell} \in \mathbb{C}$ corresponding to UE $k$ at RRU $\ell$ obtained by correlating the received signal $\mathbf{Y}_\ell$ with the conjugate of the pilot $\boldsymbol{\psi}_k$ is given by [5]

$$y_{k\ell} = \mathbf{Y}_\ell \boldsymbol{\psi}_k^H = \sqrt{\rho_k} h_{k\ell} + n_{k\ell} \tag{5}$$

where $n_{k\ell} \sim \mathcal{N}_\mathcal{C}(0, \sigma_n^2)$

To mitigate the variations in RSS due to small scale fading in the UL channel $h_{k\ell}$, channel hardening is implicitly assumed in the system [5, 7, 10]. RSS calculation and position estimation by both the FFNN and GRU is performed within the duration of this hardened channel by averaging the small scale variations across multiple coherence blocks.

The shadowing terms from an RRU to distinct location points in the network area is correlated as [5]

$$\mathbb{E}\{\nu_{m\ell}\nu_{ij}\} = \begin{cases} \sigma_{SF}^2 * 2^{-\frac{d_{mi}}{d_{corr}}} & \text{if } \ell = j \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $\nu_{m\ell}$ is the shadowing from RRU $\ell$ to location point $m$, $d_{mi}$ is the distance between locations $i$ and $m$ and $d_{corr}$ is the decorrelation distance that is characteristic of the environment. The location points correspond to the RPs and the TPs during the positioning exercise. It's worth noting that in simulations, shadowing effects are specifically accounted for in the context of TPs, whereas for RPs, such consideration is not necessary [8]. This distinction arises from the fact that shadowing can be effectively mitigated for RPs through spatial averaging, leveraging the known locations of both the UE and the RRU. However, in the case of TPs, this spatial information is unavailable beforehand, as the system lacks prior knowledge of their locations

The estimated RSS $\Theta_{k\ell} \in \mathbb{R}$ for UE $k$ at RRU $\ell$, considering channel hardening is given by [10]

$$\begin{aligned} \Theta_{k\ell} &= \mathbb{E}\{|y_{k\ell}|^2\} \\ &= \mathbb{E}\{|\sqrt{\rho_k} h_{k\ell} + n_{k\ell}|^2\} \\ &= \rho_k \mathbb{E}\{|h_{k\ell}|^2\} + \mathbb{E}\{|n_{k\ell}|^2\} \\ &= \rho_k \beta_{k\ell} + \sigma_n^2 \end{aligned} \tag{7}$$

where the expectation is with respect to the channel realizations.

To construct a fingerprint, a known UE $k$ is strategically positioned at $J$ predefined locations across the coverage area. At each of these J positions, the RSS from the known UE is measured at every RRU, resulting in an $J \times L$ matrix of RSS values. The estimated RSS at RRU $\ell$ from a UE $k$ placed at an RP $j$ with a known coordinate $\mathbf{p}_j = (x_j, y_j) \in \mathbb{R}^{1\times 2}$ is denoted as $\Theta_{j\ell} \in \mathbb{R}$, for $j = \{1, 2, \ \dots \ , J\}$ where $J$ is the total number of RPs in the coverage area. Specifically, when the UE $k$ is placed RP $j$,

$$\Theta_{j\ell} = \Theta_{k\ell} = \rho_k \beta_{k\ell} + \sigma_n^2 \tag{8}$$

$\mathbf{\Theta}_j = [\Theta_{j1}, \Theta_{j2}, \ \dots \ , \Theta_{jL}] \in \mathbb{R}^{1\times L}$ is the vector of RSS from RP $j$ at all the $L$ APs. The RSS fingerprint vector $\mathbf{\Theta} \in \mathbb{R}^{J\times L}$ is thus computed as

$$\mathbf{\Theta} = [\mathbf{\Theta}_1^T, \mathbf{\Theta}_2^T, \ \dots \ , \mathbf{\Theta}_J^T]^T \tag{9}$$

This matrix forms the RSS fingerprint corresponding to the specified positions. Subsequently, this comprehensive RSS fingerprint dataset is systematically constructed and stored in the CPU.

It's worth noting that this model is adapted from my research study tentatively titled "*Fingerprint-based Positioning in Cell-Free Massive MIMO Systems with Gaussian Process Regression*", which is currently under review (and hence not referenced). While the fundamental system model remains consistent, the simulation setting has been tailored to a distributed MIMO scenario, deviating from the cell-free setting outlined in the aforementioned paper.

# 4 Proposed Fingerprinting Solution

As highlighted in the introduction, the fingerprinting solution comprises two neural networks concatenated together, each serving distinct roles that, when combined, provide a refined positioning estimate. The first neural network, the FFNN, specializes in static positioning, leveraging its architecture to accurately determine instantaneous positions within the environment. On the other hand, the GRU network focuses on dynamic positioning for handling the continuous changes in user location over time. This section provides a detailed exposition of both the FFNN architecture and the GRU architecture, elucidating their respective components and functionalities. Additionally, it delves into the Geolife GPS trajectory dataset utilized for training the GRU model, emphasizing its significance in capturing real-world mobility patterns and behaviors. The section articulates the workflow that interconnects these components, elucidating how the FFNN and GRU coupled with the Geolife GPS trajectory dataset, collaboratively contribute to the overall positioning process, forming a robust framework for precise and adaptive positioning.

## 4.1 Feed-forward Neural Network Architecture

A Feedforward Neural Network (FFNN) is a fundamental type of artificial neural network where information flows in one direction, forward from the input layer through one or more hidden layers to the output layer without any feedback loops [26]. Here's a detailed overview of the components and functioning of an FFNN:

1. **Input Layer**:
    - The input layer is the initial layer of the neural network where external data is introduced.
    - Each node in the input layer represents a feature or attribute of the input data.

- The number of nodes in the input layer is determined by the dimensionality of the input data.

2. **Hidden Layers**:
   - Hidden layers are intermediary layers between the input and output layers.
   - Each hidden layer consists of multiple nodes, also known as neurons or units.
   - Each node in a hidden layer receives inputs from all nodes in the previous layer and computes a weighted sum of these inputs.
   - The weighted sum is then passed through an activation function to introduce non-linearity into the network and produce the output of the neuron.
   - Multiple hidden layers allow the network to learn complex hierarchical representations of the input data.

3. **Activation Functions:**
   - Activation functions introduce non-linearity into the network, enabling it to learn and approximate complex functions
   - Common activation functions include:
     - Sigmoid : Maps the weighted sum of inputs to a value between 0 and 1, suitable for binary classification tasks.
     - Hyperbolic Tangent (tanh): Similar to the sigmoid function but maps inputs to a value between -1 and 1, allowing for stronger gradients during training.
     - Rectified Linear Unit (ReLU): Returns the input if it is positive and zero otherwise, facilitating faster convergence during training.
   - The choice of activation function depends on the nature of the problem and the characteristics of the data.

4. Weights and Biases:
   - Each connection between nodes in adjacent layers is associated with a weight parameter.
   - Weights determine the strength of connections between neurons and are adjusted during the training process to minimize the error between predicted and actual outputs.
   - Additionally, each node in the hidden layers and the output layer is associated with a bias parameter, which allows the network to fit the data better.

5. Output Layer:
   - The output layer is the final layer of the neural network responsible for producing the network's predictions or outputs.
   - The number of nodes in the output layer depends on the nature of the problem :
     - For binary classification tasks, a single node with a sigmoid activation function is commonly used to produce a probability score between 0 and 1.
     - For multi-class classification tasks, the output layer may consist of multiple nodes, each representing a class, with a softmax activation function to produce probability distributions over the classes.
     - For regression tasks, the output layer typically consists of a single node with a linear activation function to produce continuous output values.

Overall, an FFNN learns to map input data to output predictions through a series of interconnected layers, with each layer performing computations on the data before passing it on to the next layer. By adjusting the weights and biases during training, the network optimizes its parameters to accurately model complex relationships in the data and make accurate predictions.

A neural network is trained to understand our problem by iteratively adjusting its internal parameters based on the data it's exposed to. Training a neural network encompasses the iterative process of adjusting its parameters, such as weights and biases, to minimize the disparity between predicted outputs and true target values. This iterative adjustment is achieved through optimization algorithms like gradient descent, which compute the gradients of a loss function with respect to the network's parameters and update them accordingly to minimize the loss. The training process continues for multiple iterations or epochs until the network converges to a satisfactory level of performance, where further training does not significantly improve its performance on a validation dataset. Once trained, the network's performance is evaluated using a separate dataset called the test dataset, which was not used during training. This evaluation involves computing the network's predictions for the input samples in the test dataset and comparing them with the true target values. Various evaluation metrics, such as accuracy, precision, recall, or mean squared error, are computed to assess the network's performance and its ability to generalize to new, unseen data.

In the current study, the FFNN is employed for static estimation and is trained with the stored RSS fingerprint $\mathbf{\Theta}$ to generate static positioning coordinates $\widehat{\mathbf{p}} = \mathbf{p_f} = (x_f, y_f) \in \mathbb{R}^{1\times 2}$ for the UE. The network architecture comprises an input layer with $L$ nodes, corresponding to the RSS values from the $L$ RRUs, followed by two hidden layers, each consisting of 16 nodes. The choice of a Leaky ReLU activation function after each linear transformation aids in capturing nonlinear relationships within the data. The output layer comprises two nodes, corresponding to the predicted x and y coordinates of the target location. This FFNN used is shown in Figure 3. The supervised training model is shown in Figure 4

The neural network supervised training involves three steps :

1. Aggregate all training pairs, denoted as $(\mathbf{p_j}, \mathbf{\Theta_j})$, where for j ={1,2,..,J}
2. Define a loss function $L(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the parameters (i.e., tunable weights and biases) of the FFNN.
3. Select $\boldsymbol{\theta}$ to minimize the loss function $L(\boldsymbol{\theta})$.

A loss function, also known as a cost function or objective function, is a mathematical function that quantifies the discrepancy between the predicted outputs of a model ($\widehat{\mathbf{p}}$) and the true target values in a training dataset ($\mathbf{p}$).The loss function is a measure of training error. For the positioning problem at hand we use root mean squared error(MSE) or the eucledian distance between the the true and predicted coordinates i.e.

$$L(\boldsymbol{\theta}) = \sqrt{L_{MSE}(\boldsymbol{\theta})} = \frac{1}{J}\sum_{i=1}^{J} \|\boldsymbol{p}_j - f_{\boldsymbol{\theta}}(\mathbf{\Theta}_j)\| \tag{10}$$

and the optimal parameter ($\boldsymbol{\theta}^*$) estimation is performed using loss minimization algorithms such as gradient descent, stochastic gradient descent or adaptive moment estimation (Adam) optimizers i.e.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}(L(\boldsymbol{\theta})) = \arg\min_{\boldsymbol{\theta}} \frac{1}{J}\sum_{i=1}^{J} \|\boldsymbol{p}_j - f_{\boldsymbol{\theta}}(\mathbf{\Theta}_j)\| \tag{11}$$

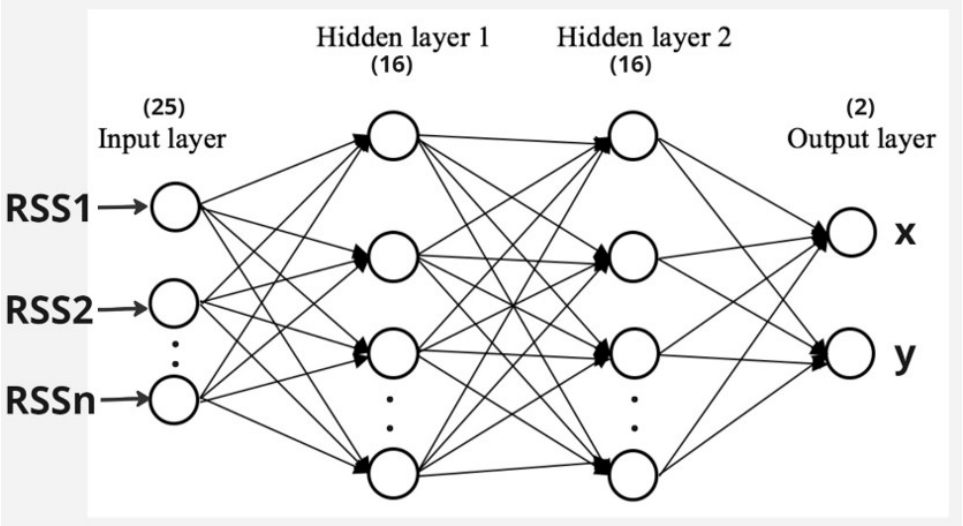


Figure 3: Feed-forward Neural Network Architecture

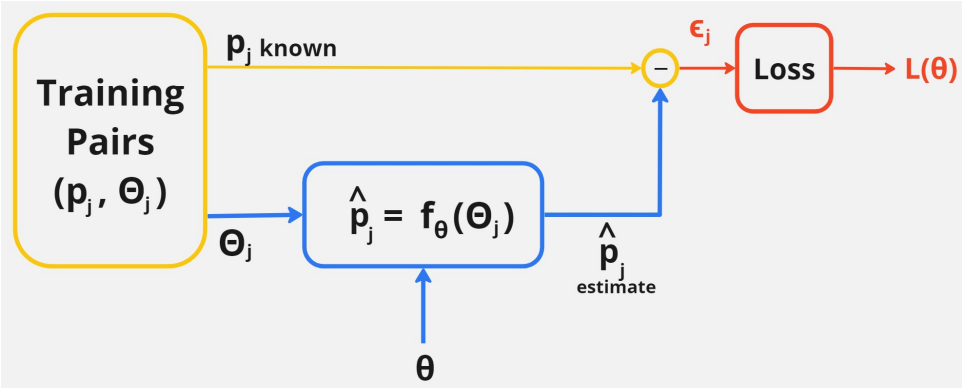


Figure 4: Feed-forward Neural Network Supervised Training

The selection of specific architectural elements, such as the number of hidden layers and the activation function, was informed by simulation experiments conducted during the research process. Initially, a FFNN with 10 layers, each comprising 32 nodes and approximately 10,000 parameters, was considered for the mid-term review. However, this architecture exhibited significant estimation errors, prompting further experimentation. Through iterative refinement, the architecture was ultimately revised to feature two hidden layers, each with 16 nodes, resulting in improved performance.

While the input layer's size is constrained by the number of RRUs, ensuring that each node is input the RSS $\Theta_{kl}$ calculated at every RRU during both the training and testing phases, the output layer's dimensions correspond to the number of coordinates required for positioning (two for 2D positioning and three for 3D positioning). Moreover, the flexibility to adjust the number of hidden layers allows for scalability, facilitating adaptation to larger simulation areas with increased training samples. Through iterative experimentation and refinement, the chosen FFNN architecture demonstrates its efficacy in accurately estimating static positioning coordinates for the UE within the designated environment.

To calculate the number of parameters $n_{\boldsymbol{\theta}}$ for the FFNN, we need to consider the weights and biases associated with each layer:

1. **Input Layer to First Hidden Layer:**
   - Number of weights = Number of input nodes * Number of nodes in the first hidden layer = L * 16
   - Number of biases = Number of nodes in the first hidden layer = 16
2. **First Hidden Layer to Second Hidden Layer:**
   - Number of weights = Number of nodes in the first hidden layer * Number of nodes in the second hidden layer = 16 * 16
   - Number of biases = Number of nodes in the second hidden layer = 16
3. **Second Hidden Layer to Output Layer:**
   - Number of weights = Number of nodes in the second hidden layer * Number of output nodes = 16 * 2
   - Number of biases = Number of output nodes = 2

Total number of parameters = $n_{\boldsymbol{\theta}}$ = Number of weights + Number of biases

$$\begin{aligned} n_{\boldsymbol{\theta}} = & \underbrace{(\text{L} * 16) + 16}_{input\ to\ first\ layer} + \underbrace{(16 * 16) + 16}_{first\ to\ second\ layer} + \underbrace{(16 * 2) + 2}_{second\ to\ output\ layer} \\ & = 16L + 16 + 256 + 16 + 32 + 2 \\ & = 16L + 322 \end{aligned} \tag{12}$$

## 4.2 Geolife GPS trajectory dataset

Now that the architecture for static position estimation is established, the focus shifts to incorporating dynamic motion of a UE within the distributed MIMO network area. To achieve this, actual GPS trajectory data collected from real-world scenarios is utilized to understand how users move outdoors during various activities such as walking, driving, cycling, swimming, etc.

The rationale behind integrating real GPS trajectory data lies in its ability to encapsulate valuable insights into user movement patterns, including velocity, acceleration, jerks, changes in speed during curves and breaks, uphill and downhill movements, and more. By this data, we gain a comprehensive understanding of how users navigate outdoor environments, thereby enabling us to simulate realistic user movements within the distributed MIMO network area.

The fundamental idea underlying this approach is that the GPS data captured during real user movement scenarios provides crucial information about the user's behavior and trajectory characteristics. When the user traverses a path within a distributed MIMO network area in a manner similar to the patterns observed in the GPS trajectory data, the FFFNN can generate static position estimates for each location along that path. Subsequently, the GRU can leverage the knowledge of these movement patterns to refine and correct the estimates generated by the FFNN. By incorporating information about the user's movement history and previous estimates, the GRU enhances the accuracy of the position estimates, effectively accounting for variations in movement dynamics and ensuring more reliable localization within the distributed MIMO network area. This dynamic integration of FFNN and GRU allows for robust and accurate positioning of the UE, even in the presence of varying movement patterns and environmental conditions.

GPS data is readily available online and can be easily accessed from various sources, making it a convenient and cost-effective option for training models. Unlike fingerprint collection, which requires extensive fieldwork and specialized equipment, acquiring GPS data involves minimal effort and expense. Additionally, GPS data provides a rich source of information about real-world user movements, including diverse activities and environments. Alternatively, path simulators offer a viable approach for generating paths similar to real-world user movements. These simulators use algorithms to simulate user trajectories based on predefined parameters and movement patterns. While they may not capture the intricacies of real-world movements as accurately as GPS data, path simulators provide a controlled environment for testing and evaluating positioning algorithms.

Learning movement patterns from readily available GPS data or path simulators offers a distinct advantage over RSS data collection, which necessitates extensive fieldwork and specialized equipment. The process of manually collecting RSS data along possible paths in the actual network and then learning the patterns from this data is labor-intensive and resource-intensive. In contrast, leveraging GPS data or path simulators provides a more convenient and cost-effective alternative. These approaches allow researchers to access a wealth of movement data without the need for time-consuming fieldwork or expensive equipment.

The GPS trajectory dataset utilized in this study originates from the GeoLife project conducted by Microsoft Research Asia [25]. Over a span of three years, 182 users contributed to this dataset, covering the period from April 2007 to August 2012. Each trajectory in the dataset comprises a sequence of time-stamped points, with each point containing latitude, longitude, and altitude information. In total, the dataset encompasses trajectories spanning approximately 1.2 million kilometers and a cumulative duration exceeding 48,000 hours. The trajectories were captured using various GPS loggers and GPS-enabled mobile devices, resulting in a diverse range of sampling rates. Notably, 91 percent of the trajectories are densely represented, with sampling rates ranging from every 1 to 5 seconds or every 5 to 10 meters per point.

This dataset encapsulates a wide array of outdoor movements undertaken by users, including walking, driving, cycling, boating, and more. Its comprehensive coverage and rich diversity make it an invaluable resource for studying human mobility patterns, urban transportation dynamics, and location-based services.

Moreover, this dataset has found extensive application in the domain of deep learning, particularly in the context of trajectory prediction and behavior analysis. Deep learning models, such as RNNs, CNNs, and their variants, have been leveraged to extract meaningful insights from the tra-

jectories captured in this dataset. These insights facilitate the development of intelligent systems for diverse applications, including personalized route recommendation, traffic forecasting, urban planning, and location-based advertising. By harnessing the spatial and temporal dependencies embedded within the trajectories, deep learning algorithms can uncover intricate patterns and behaviors, thereby enhancing our understanding of human mobility and informing the design of more effective location-based services.

A path extracted from the GPS dataset is superimposed onto the distributed MIMO (dmimo) simulation area (further elaborated in the simulation section), and the UE is simulated to traverse along this path. At each point along the path, the RSS vector $\boldsymbol{\Theta}$ is computed and serves as input to the FFNN. Concurrently, the position estimates generated by the FFNN within a specified time window are fed into the GRU, which is trained to predict the exact coordinates of the most recent estimate along the path. This dynamic integration of the FFNN and GRU enables the refinement of position estimates based on the historical trajectory of the UE, thereby enhancing the accuracy of localization within the dmimo simulation area.

### 4.3 Gated Recurrent Unit Architecture

The GRU handles the dynamic aspect of positioning in the proposed positioning model.A GRU is a type of RNN architecture that is specifically designed to address the vanishing gradient problem commonly encountered in traditional RNNs [27]. The GRU architecture was introduced as an enhancement over the standard RNN model, offering improved training performance and capturing long-range dependencies in sequential data. The GRU achieves this by incorporating gating mechanisms that regulate the flow of information within the network. GRU gates allow selective update of information and is is trained using backpropagation through time.

The key components of a GRU include:

1. **Update Gate** ($\boldsymbol{z_t}$): The update gate determines how much of the previous memory state should be retained and how much of the new candidate memory state should be added. It is computed as a sigmoid function of the concatenation of the current input ($x_t$) and the previous hidden state ($h_{t-1}$), allowing the network to learn which information to update or discard.

2. **Reset Gate** ($\boldsymbol{r_t}$): The reset gate controls how much of the previous memory state should be forgotten when computing the new candidate memory state. Similar to the update gate, it is computed as a sigmoid function of the concatenation of the current input and the previous hidden state, allowing the network to selectively reset its memory.

3. **Candidate Memory State** ($\boldsymbol{\tilde{h}_t}$): The candidate memory state represents the new information that the network considers relevant for the current time step. It is computed as a combination of the current input and the previous hidden state, modulated by the reset gate. This allows the network to update its memory state while preserving relevant information from previous time steps.

4. **Hidden State** ($\boldsymbol{h_t}$): The hidden state represents the output of the GRU at each time step and serves as the memory state that is passed to the next time step. It is computed as a combination of the previous hidden state and the candidate memory state, weighted by the update gate. This adaptive combination allows the network to selectively update its memory while retaining relevant information over time.

The above componets are matematically described in (13).

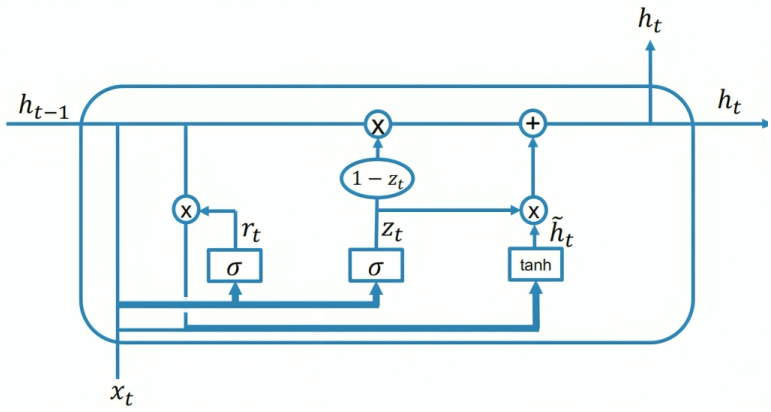


Figure 5: Typical architecture of a GRU cell

$$
\begin{aligned}
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
\tilde{h}_t &= \tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
\tag{13}
$$

In these equations,

- $\sigma$ represents the sigmoid activation function.
- tanh represents the hyperbolic tangent activation function.
- $W_r, W_z$ *and* $W$ are weight matrices
- $b_r, b_z$ *and* $b$ are bias vectors
- $[h_{t-1}, x_t]$denotes the concatenation of the previous hidden state $h_{t-1}$ and the current input $x_t$

The architecture of a typical GRU cell is shown in Figure 5. By dynamically adjusting the update and reset gates at each time step, the GRU can effectively capture and retain long-term dependencies in sequential data, making it well-suited for tasks such as sequence prediction, language modeling, and time series analysis. Moreover, the GRU architecture offers advantages in terms of computational efficiency and parameter optimization compared to other RNN variants like the LSTM network, making it a popular choice in various applications requiring sequential data processing.

For the dynamic positioning task, a GRU comprising 10 input nodes and 300 hidden states, resulting in approximately 100,000 parameters ($\boldsymbol{\theta_g}$). This architecture was chosen to capture temporal dependencies and enable the model to learn patterns in the sequence of input data. The training dataset was constructed with a window size of 5, meaning that each time step contributes two coordinates $\boldsymbol{p}_f^t = (x_f^t, y_f^t) \in \mathbb{R}^{1\times 2}$ to the GRU input, totaling 10 nodes. Here, $(x_f^t, y_f^t)$ denotes the value of $(x_f, y_f)$ at time instant $t$. Specifically, each entry in the dataset includes 5 pairs of estimated $(x_f, y_f)$ coordinates from 5 consecutive time instants obtained from the preceding FFNN. The GRU processes this input sequence to capture temporal dependencies and generate predictions for the corrected estimates of the x and y coordinates. The selection of a window size of 5 for the training pairs was determined through experimentation and can be regarded as a hyperparameter.

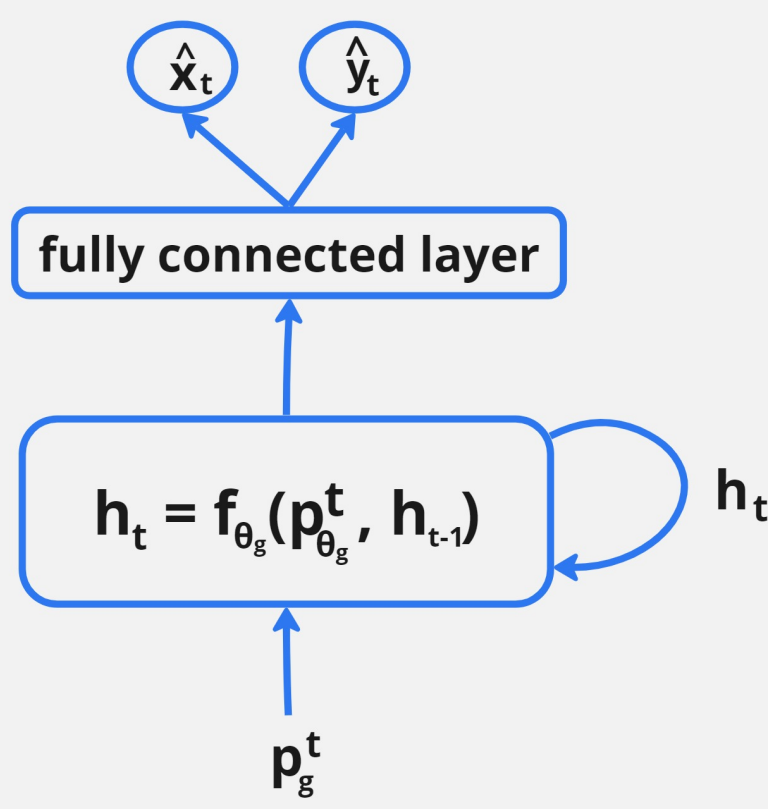


Figure 6: State Machine Viewpoint of GRU

The output of the GRU is then passed through a fully connected layer, which produces a 2-node output representing the corrected estimates of the x and y coordinates derived from the FFNN output. The target labels correspond to the actual $\boldsymbol{p} = (x, y) \in \mathbb{R}^{1\times 2}$ coordinates obtained from the overlaid path from the latest time instant from which $\boldsymbol{p}_f = (x_f, y_f)$ was generated by the FFNN, providing the ground truth for training and evaluation. This architecture enables the GRU to refine the static position estimates produced by the FFNN by incorporating temporal information and correcting for any inaccuracies or discrepancies.

The training pairs $(\boldsymbol{p}^t_g, \boldsymbol{p^t})$ are constructed for each time instant $t \in 1, 2, ..., T-4$, where $T$ represents the time of the last path in the training set and is used for training the parameters of the GRU. Each training pair $\boldsymbol{p}^t_g = (\boldsymbol{p}^{t-4}_f, \boldsymbol{p}^{t-3}_f, \boldsymbol{p}^{t-2}_f, \boldsymbol{p}^{t-1}_f, \boldsymbol{p}^t_f) \in \mathbb{R}^{1\times 10}$ consists of five consecutive estimated positions obtained from the FFNN at each time step as input and $\boldsymbol{p^t}$ as the label. $\boldsymbol{\hat{p}^t} = \boldsymbol{(\hat{x}^t, \hat{y}^t)} \in \mathbb{R}^{1\times 2}$ is the final estimate of the UE position by the model.

The state machine viewpoint of the GRU and the time unrolled viewpoint of the GRU is shown in Figures 6 and 7 respectively. These represent the recurrent computation across multiple time steps as a series of individual computational steps, essentially expanding or "unrolling" the recurrent connections over time. In the context of GRU, unrolling the loop allows for backpropagation through time (BPTT) to be applied more easily, facilitating gradient calculation and parameter updates.

When unrolling the loop of a GRU, the network architecture remains the same across all time steps, with the same set of parameters being used at each time step. This means that the weights and biases of the GRU are shared across different time steps during both forward and backward passes through the network. By sharing parameters across time steps, the GRU can effectively

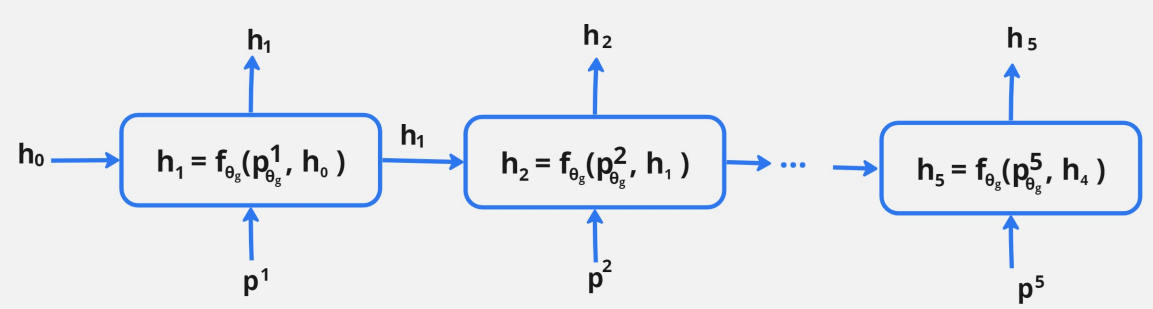


Figure 7: Time unrolled Viewpoint of GRU

learn and capture temporal dependencies in sequential data.

During training, the forward pass of the GRU is performed for each time step, processing input sequences sequentially. The hidden state at each time step is computed based on the input at that time step and the previous hidden state. The gradients are then computed using backpropagation through time, which essentially propagates errors backward through the unrolled network to update the model parameters.

The total number of parameters $n_{\boldsymbol{\theta_g}}$ in the proposed GRU is calculated as follows:

1. Parameters in the GRU layer:
   - Input to hidden weights: 10 (input size) * 300 (hidden size) = 3000 parameters
   - Hidden to hidden weights: 300 (hidden size) * 300 (hidden size) = 90000 parameters
   - Hidden bias: 300 parameters

   Total parameters in the GRU layer: 3000 + 90000 + 300 = 93300
2. Parameters in the fully connected (linear) layer:
   - Input to output weights: 300 (hidden size) * 2 (output size) = 600 parameters
   - Output bias: 2 parameters

   Total parameters in the fully connected layer: 600 + 2 = 602

Total number of parameters = $n_{\boldsymbol{\theta_g}}$ = Parameters in GRU layer + Parameters in fully connected layer

$$\begin{aligned} n_{\boldsymbol{\theta}_g} &= \underbrace{93300}_{GRU\ layer} + \underbrace{602}_{Linear\ layer} \\ &= 93902 \end{aligned} \tag{14}$$

## 5 Simulations and Discussion

The simulation setup of the distributed-MIMO system is illustrated in Figure 8. In the setup of the 200mx200m coverage area, 25 RRUs or APs are strategically distributed in a grid pattern across the entire coverage area. The distance between two successive RRUs is 45m. Additionally, 100 RPs are strategically placed uniformly within the coverage area. The RPs are systematically arranged in a square pattern, forming a grid that spans the entirety of the simulation area. Each of these

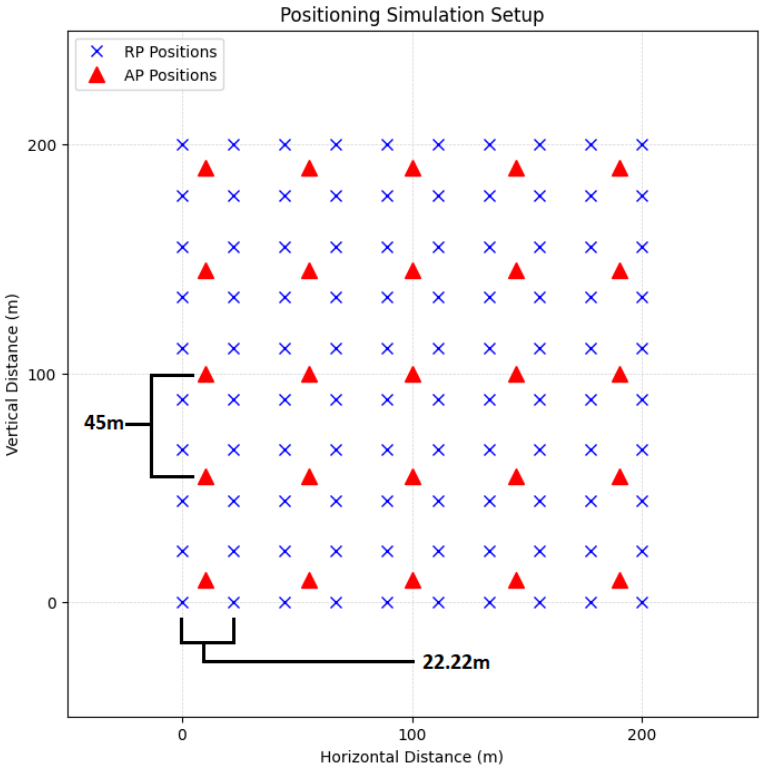


Figure 8: Simulation Setup

RPs demarcates a small square within the simulation landscape as shown in Figure 8. The distance between two successive RPs is 22.2m.

Approximately 1000 paths representing two activities, walking ( 350 paths) and biking ( 700 paths), were extracted from the Geolife GPS trajectory dataset and overlaid onto the simulation area. This was aimed at generation of realistic movement paths essential for simulating user mobility patterns. By extracting and overlaying paths from the dataset onto a predefined grid, this approach enables the creation of diverse movement patterns representative of outdoor activities. The algorithm accounts for spatial constraints, such as grid boundaries, ensuring that generated paths remain within a specified area for accurate simulation. Through this methodology described in Algorithm 1, the generation of realistic movement data for evaluating the performance of positioning algorithms in wireless networks is facilitated. A sample setup illustrating the path trails of the GPS trajectory data overlaid onto the simulation setup is depicted in Figure 9.

In the simulations, an instance of the neural network model. The experiments were conducted on an Nvidia T4 GPU. The root mean squared error (MSE) loss function is defined using torch.sqrt(nn.MSELoss()). This loss function is commonly used for regression tasks. The Adam optimizer is chosen to optimize the parameters of the neural network. Adam is a popular choice due to its adaptive learning rate capabilities, which can lead to faster convergence during training.

The equations 15 illustrate the fundamental principles of the Adam optimization algorithm

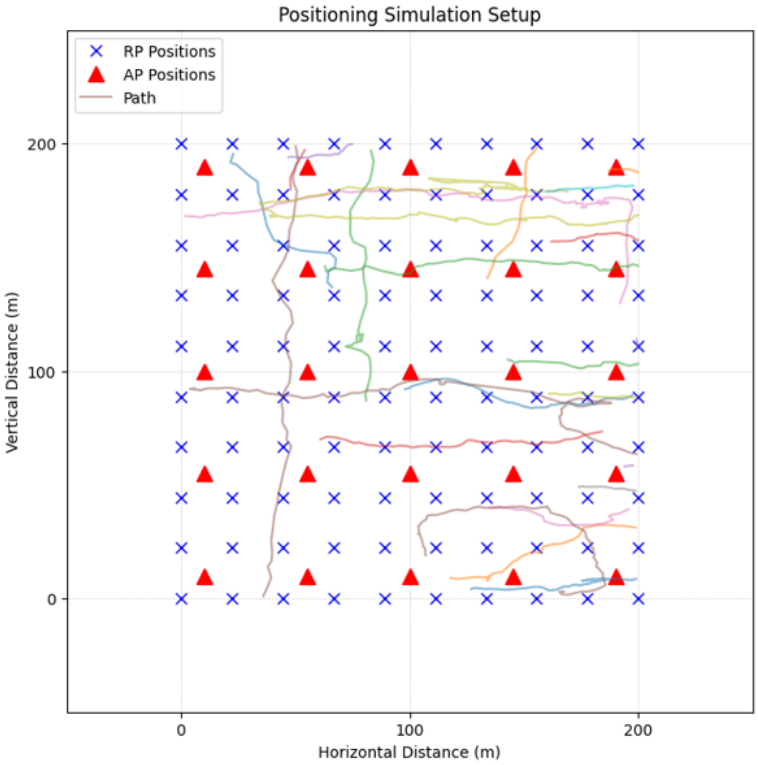


Figure 9: GPS trajectory paths overlaid onto simulation setup

$$
\begin{aligned}
m_{t+1} &= \beta_1 m_t + (1-\beta_1) g_{t+1} \\
v_{t+1} &= \beta_2 v_t + (1-\beta_2)(g_{t+1})^2 \\
\theta_{t+1} &= \theta_t - \alpha * \frac{\hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + \epsilon}
\end{aligned}
\tag{15}
$$

Here,

- $m_{t+1}$ and $v_{t+1}$ represent exponentially weighted averages of gradients and squared gradients respectively.
- $\beta_1$ and $\beta_2$ are the decay rates for the first and second moments.
- $g_{t+1}$ denotes the gradient of the objective function at time $t+1$.
- $\theta_t$ denotes the parameters at time $t$.
- $\alpha$ represents the learning rate.
- $\hat{m}_{t+1}$ and $\hat{v}_{t+1}$ are bias-corrected estimates of the first and second moments.
- $\epsilon$ is a small constant added to prevent division by zero.

The learning rate was set to $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The training loop iterates over a specified number of epochs (about 3500 epochs for the FFNN and 30 epochs

for GRU) to train the neural network. Within each epoch, the loop iterates over the training data batches, computes the loss, performs backpropagation, and updates the model parameters.

**Algorithm 1** Generating Paths for Walking and Biking Activities

```
Initialize path dictionary path_dict
Initialize path_idx ← 0
for each activity type (walk, bike) do
    Start at the first point of the activity
    Generate random starting point (x_1, y_1) within a 200mx200m grid
    Append (x_1, y_1) and corresponding time t_1 to path_dict[path_idx]
    for each subsequent point (long_2, lat_2) @ t_2 do
        Calculate (x_2, y_2) coordinates of the next point
        if (x_2, y_2) is within the 200mx200m grid then
            Append (x_2, y_2) and t_2 to path_dict[path_idx]
        else
            Increment path_idx
            Go to the next point if available
            if no more points available for current path then
                Move to the next path in same activity type
            end if
        end if
    end for
end for
```

For the FFNN, RSS tensors of size (25,1000) were generated, representing RSS values from 25 RRUs collected at 1000 RPs within a 100x100 grid. The training loss curve is illustrated in Figure 10, demonstrating stabilization around 2000 epochs. Remarkably, the training process completed in less than an hour due to the minimal parameters in the FFNN (722; from (12) with L=25). Subsequently, 1000 randomly generated TPs were utilized to evaluate the trained FFNN under varying shadowing noise conditions. The cumulative distribution function (CDF) of the testing loss, calculated as the Euclidean distance between the predicted and true coordinates, for different values of shadowing is displayed in Figure 11. As anticipated, the CDF curve shifts left with decreasing shadowing noise, indicating improved estimation accuracy, and hence, a reduction in average static positioning error. These results are consistent with other positioning methods in [8, 9].

For GRU training, root mean square error was once again utilized as the loss metric. Approximately 1000 trajectory paths were generated using Algorithm 1, resulting in around 20,000 (x, y) coordinates when overlaid on the simulation area. From these, the data was split into a training set and a testing set in an 8:2 ratio. Training and testing were conducted for two shadowing noise values, specifically 4dB and 8dB, with the corresponding loss curves depicted in Figure 12 and Figure 13, respectively. Around the 10th epoch, losses stabilized for 4dB shadowing, while for 8dB shadowing, it took until the 30th epoch for stabilization. As expected, the 8dB noise exhibited a higher loss compared to the 4dB noise. For 8dB shadowing, the GRU model yielded a lower estimation error, as evidenced by the CDF plot in Figure 14. The average loss with the FFNN was measured at 19.04m, whereas with the GRU, it reduced to 16.57m. However, the scenario was different for 4dB shadowing, where the FFNN alone demonstrated a lower estimation error compared to the combined FFNN-GRU model. Specifically, the average loss with the FFNN was 9m, whereas with the GRU, it increased to 12m. This observation suggests that for higher shadowing

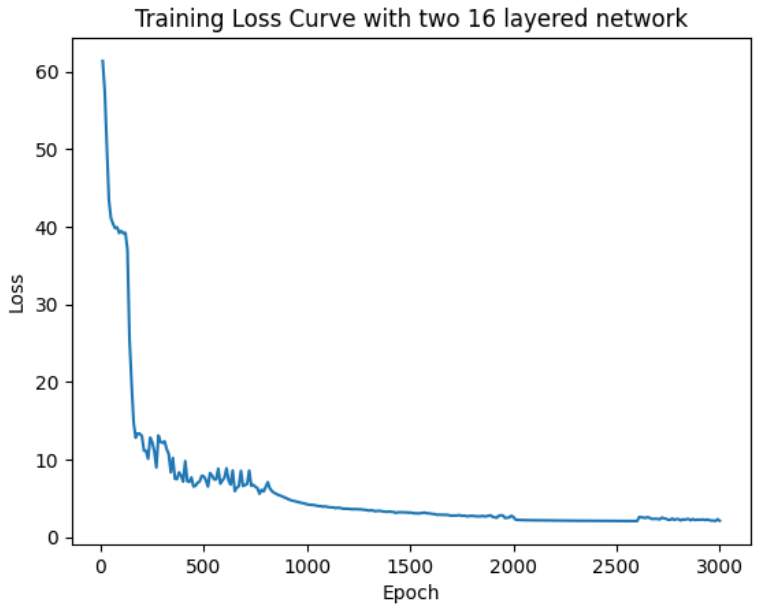


Figure 10: FFNN training loss

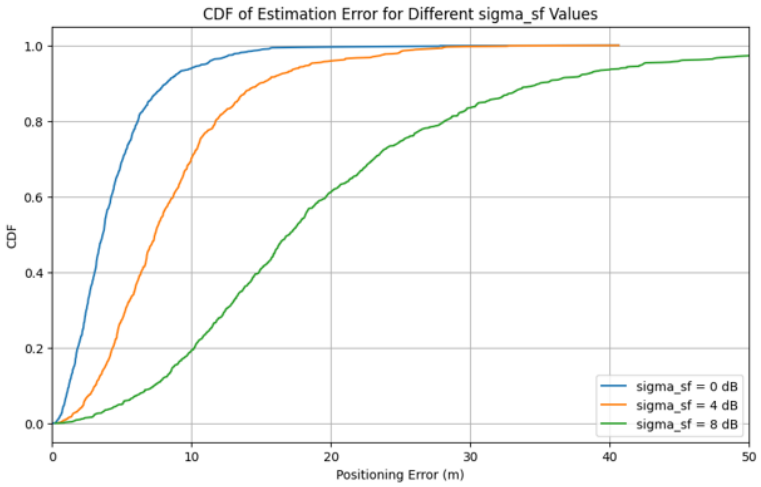


Figure 11: CDF of estimation error for FFNN ecross various shadowing noise values


Figure 12: Training Loss Over 10 Epochs for GRU, $\sigma_{SF}$ =4dB

noise scenarios, the proposed GRU-FFNN hybrid model outperforms the standalone FFNN model.

# 6 Conclusion

This study has presented a comprehensive investigation into the use of neural network architectures, specifically the FFNN and GRU, for improving the accuracy of positioning of mobile UEs in distributed MIMO systems. Through rigorous experimentation and analysis, we have demonstrated the efficacy of these models in handling dynamic positioning tasks. The FFNN, trained on RSS fingerprints, exhibited promising results for static positioning, while the GRU, leveraging GPS trajectory data, effectively addressed the dynamic aspect of the problem. Our findings highlight the importance of considering temporal information in positioning tasks. Furthermore, the proposed GRU-FFNN hybrid approach proved to be particularly effective in scenarios with higher shadowing noise, outperforming the standalone FFNN model.

In exploring future directions and addressing ongoing challenges, several avenues emerge to further advance the field of wireless positioning systems. Firstly, there is an opportunity to investigate alternative neural network architectures such as LSTMs, Transformers, or hybrid models, comparing their performance with the GRU model. This comparative analysis could provide valuable insights into the strengths and weaknesses of different architectures, guiding the selection of the most suitable model for specific positioning tasks.

Expanding the task to classification represents another promising direction, where the objective shifts to predicting the type of motion of the UE based on its trajectory. This extension would require adapting existing models or developing new ones capable of effectively classifying different types of user motion, such as walking, driving, or cycling. Such classification capabilities would enhance the versatility and applicability of positioning systems in various contexts.

Furthermore, the implementation of predictive modeling to forecast the UE's location at future time instances presents an exciting opportunity for enhancing location-based services and facilitating handoff preparations in cellular networks. Predictive modeling techniques could en-

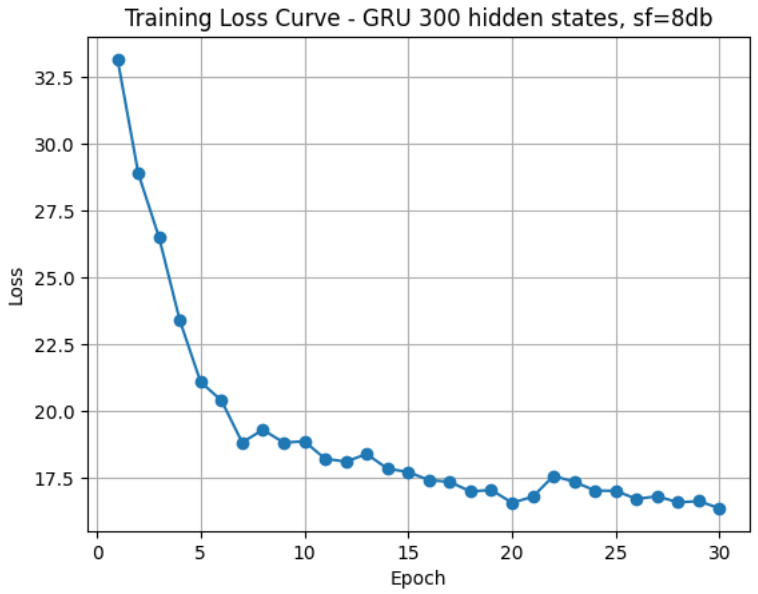


Figure 13: Training Loss Over 30 Epochs for GRU, $\sigma_{SF}$=8dB

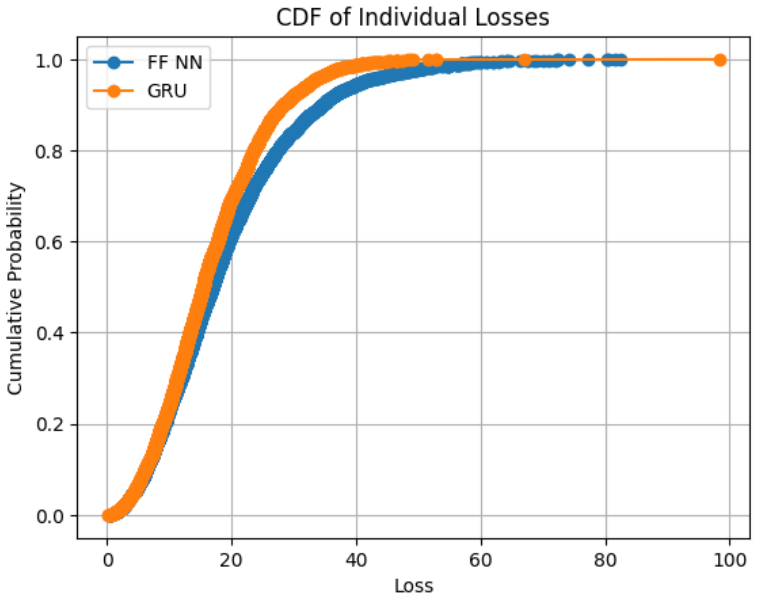


Figure 14: CDF of Testing Loss for GRU and FFNN for $\sigma_{SF}$=8dB

able proactive decision-making and resource allocation, optimizing network performance and user experience as discussed at the start.

However, as we look towards these future directions, it is essential to acknowledge and address the challenges associated with scaling up simulation areas and overlaying data collected over millions of kilometers onto smaller simulation areas. This endeavor may require the development of innovative techniques for data preprocessing, dimensionality reduction, or data augmentation to effectively handle large datasets and ensure the accuracy and reliability of simulation results. By tackling these challenges and embracing future directions, we can continue to advance the field of wireless positioning systems, unlocking new possibilities for enhanced location-aware applications and service. Overall, this research has the potential to contribute to advancing the field of wireless positioning systems, offering valuable insights and methodologies for enhancing accuracy and robustness in real-world applications.

# 7 Appendix

The code implementation for the system model, the FFNN architecture, GRU architecture, and the conducted experiments can be found in the following link: `https://tinyurl.com/mtkw7yx2`

# References


[1] A. El-Rabbany, Introduction to GPS: the global positioning system. Artech house, 2002.

[2] F. Liu et al., "Survey on WiFi-based indoor positioning techniques," IET Communications, vol. 14, no. 9, pp. 1372–1383, Jun. 2020, doi: https://doi.org/10.1049/iet-com.2019.1059.

[3] Rodriguez, Miguel & Pece, Juan & Escudero, Carlos. (2005). In-building location using bluetooth.

[4] A. Ramtohul and K. K. Khedo, 'Mobile positioning techniques and systems: A comprehensive review', Mobile Information Systems, vol. 2020, pp. 1–18, 2020.

[5] Ö. T. Demir, E. Björnson, L. Sanguinetti, and Others, 'Foundations of user-centric cell-free massive MIMO', Foundations and Trends® in Signal Processing, vol. 14, no. 3–4, pp. 162–472, 2021

[6] X. You, D. Wang, and J. Wang, Distributed MIMO and cell-free mobile communication. Springer, 2021

[7] V. Savic and E. G. Larsson, "Fingerprinting-Based Positioning in Distributed Massive MIMO Systems," 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), Boston, MA, USA, 2015, pp. 1-5, doi: 10.1109/VTCFall.2015.7390953.

[8] K. N. R. S. V. Prasad, E. Hossain and V. K. Bhargava, "Machine Learning Methods for RSS-Based User Positioning in Distributed Massive MIMO," in IEEE Transactions on Wireless Communications, vol. 17, no. 12, pp. 8402-8417, Dec. 2018, doi: 10.1109/TWC.2018.2876832.

[9] K. N. R. S. V. Prasad, E. Hossain, V. K. Bhargava and S. Mallick, "Analytical Approximation-Based Machine Learning Methods for User Positioning in Distributed Massive MIMO," in IEEE Access, vol. 6, pp. 18431-18452, 2018, doi: 10.1109/ACCESS.2018.2805841.

[10] C. Wei et al., "Joint AOA-RSS Fingerprint Based Localization for Cell-Free Massive MIMO Systems," 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 2020, pp. 590-595, doi: 10.1109/ICCC51575.2020.9344979.

[11] H. Zhang et al., 'A novel KGP algorithm for improving INS/GPS integrated navigation positioning accuracy', Sensors, vol. 19, no. 7, p. 1623, 2019

[12] M. Aslinezhad, A. Malekijavan, and P. Abbasi, 'ANN-assisted robust GPS/INS information fusion to bridge GPS outage', EURASIP Journal on Wireless Communications and Networking, vol. 2020, no. 1, p. 129, 2020.

[13] E. D. Martí, D. Martín, J. García, A. De la Escalera, J. M. Molina, and J. M. Armingol, 'Context-aided sensor fusion for enhanced urban navigation', Sensors, vol. 12, no. 12, pp. 16802–16837, 2012.

[14] A. G. Chekol and M. S. Fufa, 'A survey on next location prediction techniques, applications, and challenges', EURASIP Journal on Wireless Communications and Networking, vol. 2022, no. 1, p. 29, 2022

[15] S. Tian, X. Zhang, Y. Zhang, Z. Cao and W. Cao, "Spatio-Temporal Position Prediction Model for Mobile Users Based on LSTM," 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 2019, pp. 967-970, doi: 10.1109/ICPADS47876.2019.00146

[16] H. Y. Song, 'A future location prediction method based on lightweight LSTM with hyperparamater optimization', Scientific Reports, vol. 13, no. 1, p. 17928, 2023

[17] P. Casabianca, Y. Zhang, M. Martínez-García, and J. Wan, 'Vehicle destination prediction using bidirectional LSTM with attention mechanism', Sensors, vol. 21, no. 24, p. 8443, 2021

[18] X. Fan, L. Guo, N. Han, Y. Wang, J. Shi and Y. Yuan, "A Deep Learning Approach for Next Location Prediction," 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)), Nanjing, China, 2018, pp. 69-74, doi: 10.1109/CSCWD.2018.8465289

[19] J. Wang, C. Guo, and L. Wu, 'Gated recurrent unit with RSSIs from heterogeneous network for mobile positioning', Mobile Information Systems, vol. 2021, pp. 1–7, 2021

[20] C. Jakteerangkool and V. Muangsin, 'Short-term travel time prediction from GPS trace data using recurrent neural networks', in 2020 Asia Conference on Computers and Communications (ACCC), Singapore, Singapore, 2020.

[21] Y. Hong, H. Martin, and M. Raubal, 'How do you go where? improving next location prediction by learning travel mode information using transformers', in Proceedings of the 30th International Conference on Advances in Geographic Information Systems, 2022, pp. 1–10.

[22] A. Nawaz, Z. Huang, S. Wang, A. Akbar, H. AlSalman, and A. Gumaei, 'GPS trajectory completion using end-to-end bidirectional convolutional recurrent encoder-decoder architecture with attention mechanism', Sensors, vol. 20, no. 18, p. 5143, 2020

[23] H. Jin, H. Wu, Z. Xu, W. Huang, and C. Liu, 'Travel-mode classification based on GPS-trajectory data and geographic information using an XGBoost classifier', in IOP Conference Series: Earth and Environmental Science, 2022, vol. 1004, p. 012012.

[24] 3GPP. 2020. Study on channel model for frequencies from 0.5 to 100 GHz (Release 16). 3GPP TS 36.901

[25] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, 'Mining interesting locations and travel sequences from GPS trajectories', in Proceedings of the 18th international conference on World wide web, 2009, pp. 791–800.

[26] F. Rosenblatt, 'The perceptron: a probabilistic model for information storage and organization in the brain', Psychological review, vol. 65, no. 6, p. 386, 1958

[27] K. Cho et al., 'Learning phrase representations using RNN encoder-decoder for statistical machine translation', arXiv preprint arXiv:1406. 1078, 2014

[28] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', arXiv preprint arXiv:1412. 6980, 2014