Received: 3 October 2022 | Revised: 19 July 2023 | Accepted: 15 September 2023

DOI: 10.33012/navi.622

ORIGINAL ARTICLE

Learning GNSS Positioning Corrections for Smartphones Using Graph Convolution Neural Networks

Adyasha Mohanty | Grace Gao

Department of Aeronautics and Astronautics, Stanford University

Correspondence
Grace Gao
Department of Aeronautics and Astronautics, Stanford University
Stanford, CA, USA, 94305
Email: gracegao@stanford.edu

Abstract
Smartphone receivers comprise approximately 1.5 billion global navigation satellite system receivers currently manufactured worldwide. Smartphone receivers provide measurements with lower signal levels and higher noise than commercial receivers. Because of constraints on size, weight, power consumption, and cost, it is challenging to achieve accurate positioning with these receivers, particularly in urban environments. Traditionally, global positioning system measurements are processed via model-based approaches, such as weighted least-squares and Kalman filtering approaches. While model-based approaches can provide meter-level positioning accuracy in a postprocessing manner, these approaches require strong assumptions on the corresponding noise models and require manual tuning of parameters such as covariances. In contrast, learning-based approaches have been proposed that make fewer assumptions about the data structure and can accurately model environment-specific errors. However, these approaches provide lower accuracy than model-based methods and are sensitive to initialization. In this paper, we propose a hybrid framework for learning position correction, which corresponds to the offset between the true receiver position and the estimated position. For a learning-based approach, we propose a graph convolution neural network (GCNN) that can learn different graph structures with multi-constellation and multi-frequency signals. For better initialization of the GCNN, we use a Kalman filter to estimate a coarse receiver position. We then use this coarse receiver position to condition the input features to the graph. We test our proposed approach on real-world data sets from the Google Smartphone Decimeter Challenge and show improved positioning performance over model-based methods such as the weighted least-squares and Kalman filter methods.

Keywords
AI, convolutions, GNSS, graph learning, machine learning, urban environment

1 | INTRODUCTION

High-precision positioning with smartphones could bring in-demand technologies to users around the world, enabling applications such as lane-level accuracy for road users and autonomous cars, precise mapping, indoor positioning, and improved localization in augmented reality-based gaming environments. Over the last few years, raw global navigation satellite system (GNSS) measurements from smartphone receivers have become more publicly accessible, as demonstrated by the release of the Android GNSS application program in 2016 (Humphreys et al., 2016) and the Google open data sets in 2020 (Fu et al., 2020). More recently, Google launched the Google Smartphone Decimeter Challenge (GSDC) (Fu et al., 2020) to invest in the development of novel technologies that can achieve high-precision positioning from smartphone measurements.

The current challenge with smartphone receivers is that they can only offer 3–5 m of positioning accuracy under good multipath conditions and over 10-m accuracy under harsh multipath environments. Because of limitations in GNSS chipset, size, and hardware cost, GNSS measurements from smartphones have lower signal levels and higher noise than commercial receivers (Guangcai & Jianghui, 2019; Zhang et al., 2018).

However, new opportunities have emerged that can be leveraged to design novel positioning algorithms. For example, with the advent of the new Android application programming interface (API) (Humphreys et al., 2016), raw GNSS measurements have become more publicly accessible, which has encouraged the development of new tools and software for processing these data sets. We also have access to multi-frequency and multi-constellation measurements, which provide redundancy while navigating in dense urban canyons, whereas GNSS measurements from a single frequency/constellation can be sparse. Lastly, we also have the capability to utilize more precise measurements, such as carrier-phase measurements, for providing decimeter-level accuracy.

Many works in the literature use a model-based approach to provide a positioning solution from raw GNSS measurements. In the work by Realini et al. (2017), the authors used the open-source goGPS software to achieve decimeter-level accuracy in stationary scenarios. In the study by van Diggelen and Wang (2018), Google designed APIs to obtain high-precision GNSS positioning with the use of the accumulated delta range (ADR) from carrier-phase measurements. Other researchers have designed algorithms combining real-time kinematics and inertial measurement units (IMUs) to achieve meter-level accuracy (Bochkati et al., 2020). The author of the winning paper of the GSDC challenge for 2021 and 2022 designed a factor graph global optimization method using ADR observations and corrected pseudorange observations from GNSS reference stations as constraints, to achieve nearly meter-level accuracy (Suzuki, 2021). Although model-based approaches have shown promising results in the GSDC challenge, these approaches require manual tuning of parameters such as covariances and require strong assumptions for the noise models.

In contrast, some learning-based approaches have been proposed that make fewer assumptions about the underlying data structure and are known for their ability to model complex environmental errors using data. Instead of learning the position directly, one can instead learn the positioning correction, which refers to the offset of the baseline position from a standard algorithm such as the weighted least-squares (WLS) or Kalman filter algorithm from the ground truth. In the work by Siemuri et al. (2021), the authors trained machine learning algorithms such as linear regression, Bayesian ridge regression, and neural network algorithms as well as a weighted combination of all three approaches to predict the positioning correction. The results showed that the weighted combination approach outperformed all three algorithms in terms of positioning accuracy. Another line of work (Kanhere et al., 2022) proposed the use of deep neural networks (DNNs) to learn the correction, with pseudorange residuals and satellite line-of-sight (LOS) vectors as inputs. The DNN leveraged a set transformer (Lee et al., 2019) that accounts for the varying number of measurements at different time instances while being permutation-invariant to the order of input measurements. The approach showed an improvement in the positioning error over the WLS baseline on real-world data. However, the positioning accuracy was limited by approximation errors from linearization around the initial position estimate and the reliance on only pseudorange measurements.

In general, learning-based approaches have not been able to outperform model-based methods, specifically on the GSDC data sets. Moreover, standalone learning-based methods have been shown to be sensitive to initialization and feature design. A notable work from 2021 (Han et al., 2021) attempted to use reinforcement learning to tune measurement noise covariances in a Kalman filter that combines GNSS-IMU measurements. However, adopting a reinforcement learning approach requires manual tuning of the reward function to learn the right measurement noise covariances. With the exception of this work, few studies have considered a hybrid approach, i.e., one that combines the benefits of both learning-based approaches and model-based approaches.

In this work, we propose a hybrid framework to learn position corrections from smartphone GNSS measurements. For the learning-based approach, we use a graph convolutional neural network (GCNN) that predicts a position correction given an initial coarse receiver position. To overcome the sensitivity of the GCNN to the inputs, we use a Kalman filter as a model-based approach to predict an initial position and to condition the input measurements to the graph. The GCNN then predicts a finer position correction by applying convolution operations to the input graph. Given the success of factor graphs in winning both the 2020 and 2021 GSDC challenges (Suzuki, 2021), the GCNN forms our design choice for the learning-based approach because the GCNN can learn such a graph with satellite positions as nodes and preconditioned inputs from a Kalman filter. The GCNN is also capable of handling varying satellite visibility in urban environments via an unordered structure of nodes and of modeling measurements from multiple constellations and multiple signal frequencies. Given a graph structure that is derived from known properties of GNSS measurements and the Kalman filter solution, the GCNN performs inference in an end-to-end manner, where the graph structure helps propagate information among neighboring nodes (Kipf & Welling, 2017).

In our work, we provide the following contributions. This paper is based on our recent Institute of Navigation GNSS+ 2022 conference paper (Mohanty & Gao, 2022).

- We propose a hybrid framework using model-based and learning-based methods to learn position corrections from smartphone GNSS measurements.
- For the learning-based module, we design a GCNN that can represent different graph structures.
- We use a Kalman filter for better initialization of the GCNN, as well as for conditioning the input features to the graph.
- We evaluate our proposed approach on real-world data sets collected in urban environments.

2 | PROPOSED ALGORITHM

2.1 | Background of GCNNs

Traditionally, neural networks have shown high predictive power in learning tasks that require fixed-size, regularly structured inputs. However, graph neural networks can operate on vector data structures such as a graph and make more informed predictions by utilizing features from the graph nodes. As a subset of graph neural networks, GCNNs (Kipf & Welling, 2017) offer the same advantages with the extended capability of performing convolutions on arbitrary graphs. Although ordinary convolutions are not node-invariant, GCNNs still retain the permutation-invariance properties of a graph neural network. This property indicates that the function learned by the graph is independent of the rows and columns in the adjacency matrix. Moreover, this property also implies that changing the order of the inputs (or the nodes in the graph) does not affect the graph-level prediction. This feature makes GCNNs suitable for modeling the varying number of GNSS satellites at each time step.

As shown in Figure 1, a GCNN takes as input a connected graph $G=(V,E)$, where V represents the nodes and E represents the edges (Kipf & Welling, 2017). We also define the cardinalities of the edges and nodes as $|V|=n$ and $|E|=m$, respectively. The graph also takes additional inputs. The first input is an adjacency matrix representation A that depicts the connection of each node to its neighboring nodes. In A, an element has a value of 1 if two nodes i and j are connected. From A, we define the degree matrix for the graph as D, where $D(i,i)=\sum_{j=1}^{n}A(i,j)$, and the Laplacian matrix as $L=D-A$. The normalized symmetric version of the Laplacian matrix is denoted as $\tilde{L}=I-D^{-1/2}AD^{-1/2}$.

The second input to the graph is a feature matrix X that describes the graph signal for each node. The matrix X has n rows, contains d signals of the graph, and represents all of the node features stacked together.

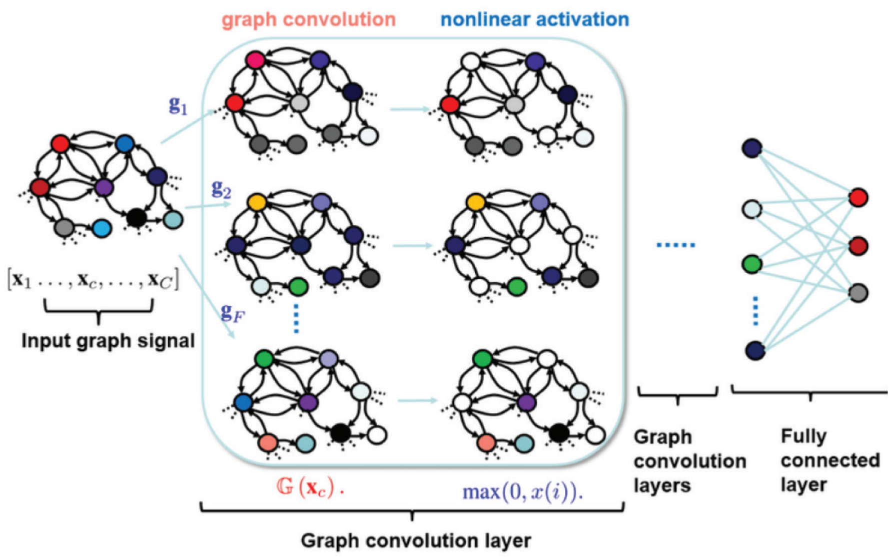


FIGURE 1 Multilayer GCNN with an example convolution layer
A GCNN takes two inputs: a feature description for every node, as indicated via different colors in the figure, and a description of the graph structure, as represented via edge connections among the nodes. The model first predicts a node-level output, which is then passed through the fully connected layer to produce a graph-level output. Figure adapted from Jian et al. (2018).

Each layer of the neural network is then described by a nonlinear function as follows:

$$H^{(l+1)} = g(H^l, A) \tag{1}$$

where H^l and $H(l+1)$ represent the output from the hidden layers of the network and the function $g(.)$ is learned by the GCNN. Kipf and Welling (2017) formulated the propagation rule for each convolution layer of the GCNN as follows:

$$f(H^l, A) = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^l W^l) \tag{2}$$

where $\tilde{A} = A + I$, $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$ and W^l refers to the weights that are learned by the GCNN.

The above equation alludes to the Weisfeiler–Lehman (WL) isomorphism test (Weisfehler & Leman, 1968), which is a heuristic for graph isomorphism testing and is used to analyze the discriminative power of graph neural networks or the ability to distinguish between different types of representations. Two graphs are considered isomorphic if there is a mapping between the nodes of the graphs that preserves node adjacencies, as illustrated in Figure 2. The WL test produces a canonical form for each graph. If the canonical forms of two graphs are not equivalent, then the graphs are not isomorphic. In a message-passing layer within a GCNN, the features of each node are updated by aggregating the features of the node's neighbors. The choice of the convolution and aggregation layers is important because only certain choices of GCNN satisfy the WL test and are thus able to learn different graph structures.

After performing several layer-wise propagations in the GCNN, we can generate a latent representation for each node and infer a node-level output Z from the graph, which can then be aggregated using pooling operations to produce a single graph-level output. In general, this operation can be summarized as follows:

$$h_G = mean / max / sum(h_1^K, h_2^K, h_3^K, \ldots) \tag{3}$$

where h_G is the pooling operation on the entire graph and h_i^K refers to the propagation output from each hidden layer of the network. Note that, similar to the weight-sharing mechanism in a standard convolutional neural network, the GCNN reuses the same filter weights across all of the different nodes because of the layer-wise propagation rule. Because of this property, the number of parameters in

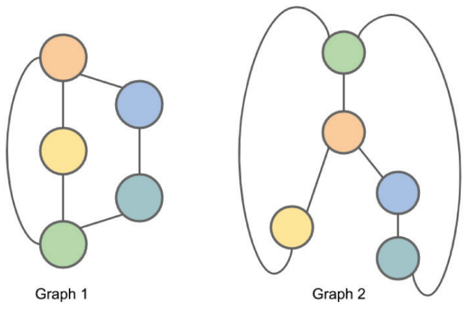


FIGURE 2 Graph 1 and Graph 2 are isomorphic. The correspondence between nodes is illustrated by the colored nodes. If we execute the WL test on these graphs, we arrive at the same canonical form for both graphs, indicating that these graphs might be isomorphic. Figure adapted from Beiber (2019).

the GCNN is not limited to the size of the graph, which makes the GCNN scalable for large predictive tasks and flexible to varying input sizes. The final prediction can be represented as follows:

$$\hat{y} = predict(h_G) \tag{4}$$

where $predict$ denotes any standard neural network, such as a perceptron or a fully connected network.

2.2 | Problem Setup

Let x_t represent the position of the receiver at time t, with an associated GNSS measurement set $M_t = (m_t^1, \ldots m_t^S)$, where S denotes the total number of satellites from multiple constellations. The constellations are denoted by C and span 1–3 where 1: GPS, 2: GALILEO, and 3: GLONASS. For this work, we use code-phase, ADR, and Doppler measurements from GPS, GLONASS, and GALILEO constellations. For GPS, we use the L1, L2, and L5 frequencies. For GLONASS, we use the L1 and L2 frequencies, and for GALILEO, we use the E1 and E5 signal frequencies. The satellites are denoted by sv and indexed with $j = 1, 2, \ldots S$, with measured pseudoranges given as ρ_t^j and positions given as p_t^j. In addition to M_t, we also have access to an initial position estimate $\tilde{x}_t$ from a model-based approach, i.e., the Kalman Filter approach. We use this initial position estimate to compute the expected range $\bar{r}_t^j$ and the true correction using the ground truth, $\Delta\mathbf{x}_t$. At each time step, the GCNN learns both a positioning correction δx_t using the measurement set M_t and the latent function $f(.)$, which captures the relationship of various node features to the predicted correction.

2.3 | Architecture

Figure 3 presents an overview of our proposed algorithm, and each module is described below.

- Kalman Filter Initialization: To mitigate the sensitivity of the network to initial inputs, we provide an initial position to the network, which is a coarse estimate of the receiver's true position. We choose a Kalman filter as our model-based approach because the Kalman filter uses the temporal history of GNSS measurements, converges quickly, and is computationally efficient. We first select GNSS measurements based on carrier-to-noise density power ratio (C/N0), satellite elevation, and carrier error values and then compute the receiver's position using the Kalman filter.
- Feature Preprocessing: We preprocess the GNSS measurements using the following steps:
  1. We group all measurements into different constellations and signal types.
  2. We eliminate inter-system and inter-frequency biases, clock biases, and tropospheric and ionospheric errors from the code-phase measurements.
  3. We apply carrier smoothing over two consecutive time epochs to the code-phase measurements by utilizing the ADR measurements. In the event of a cycle slip, we use Doppler values for smoothing the code-phase measurements. Note that the presence of a cycle slip is indicated by a bitwise operation that is recorded from the receiver.

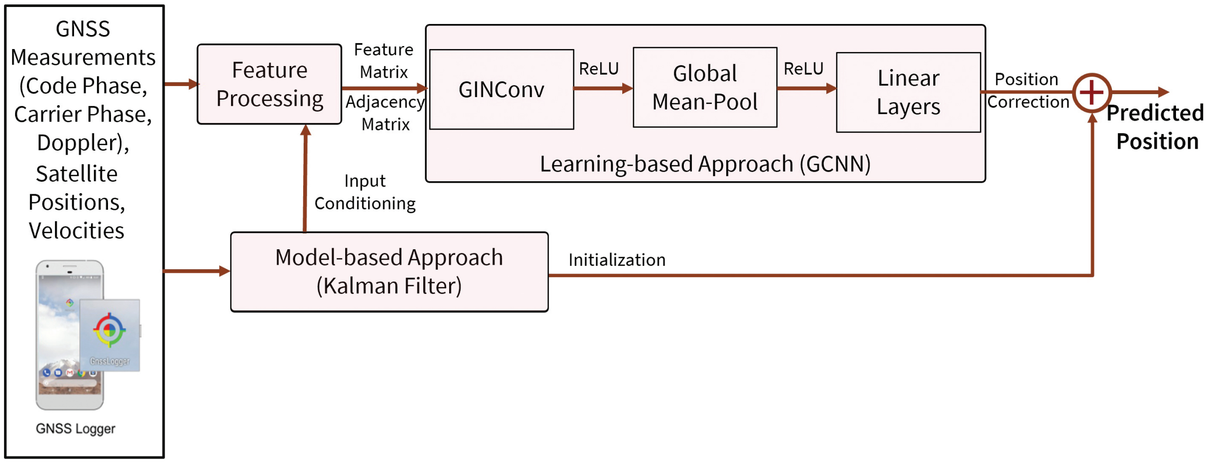


FIGURE 3 Our proposed hybrid framework has three core modules, which are highlighted. We use a Kalman filter to obtain a coarse position correction and to condition the GNSS measurements for better initialization of the learning module (feature preprocessing). The learning-based module uses a GCNN to aggregate measurements across satellites from multiple constellations and signal frequencies and fine-tunes the initial position correction.

Using the satellite grouping (Step 1) and smoothed code-phase measurements (Step 3), we construct feature vectors for each satellite. Because the choice of features is a design choice, as a proof of concept, we use simple features for the GCNN, such as LOS vectors and measurement residuals. Given the initial position estimate from the Kalman filter and satellite positions, we compute the LOS vector and measurement residuals for each satellite as follows:

$$x_{LOS} = \frac{\tilde{x}_t - p_t^j}{\| \tilde{x}_t - p_t^j \|}, x_{RES} = \rho_t^j - r_t^j \tag{5}$$

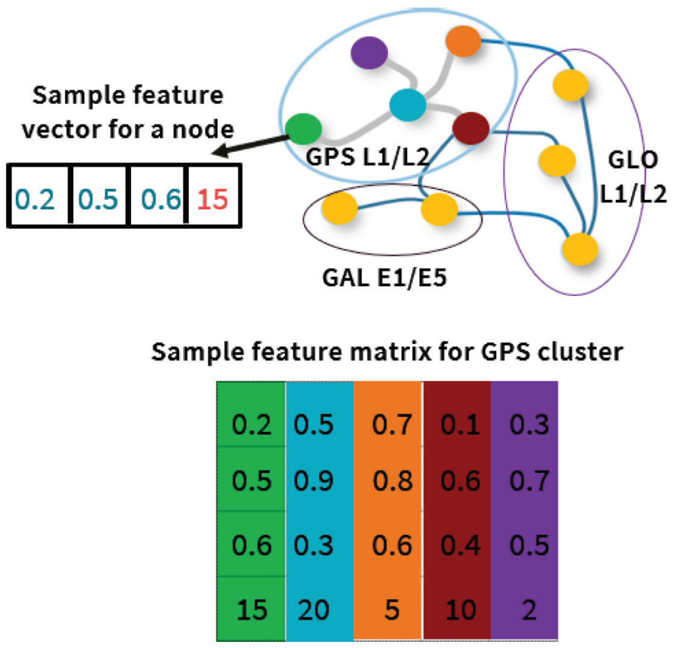
where x_{LOS} is the LOS vector and x_{RES} is the measurement residual.

- GCNN: In the graph, we represent each node using the satellite position p_t^j. Note that the graph is dynamic in nature because, for every *t*, we have a new set of measurements M_t and a new set of satellite positions. For every satellite, we concatenate the LOS vector and the measurement residual to form a 4×1 feature vector. Given the feature vectors for each node, we can form the feature matrix for the entire graph. A sample feature vector and feature matrix are illustrated in Figure 4.

To create the adjacency matrix, we establish edge connections between satellites that belong to the same constellation as well as connections between satellites from different constellations if their node features have similar measurement residuals. Mathematically, this step is described as follows:

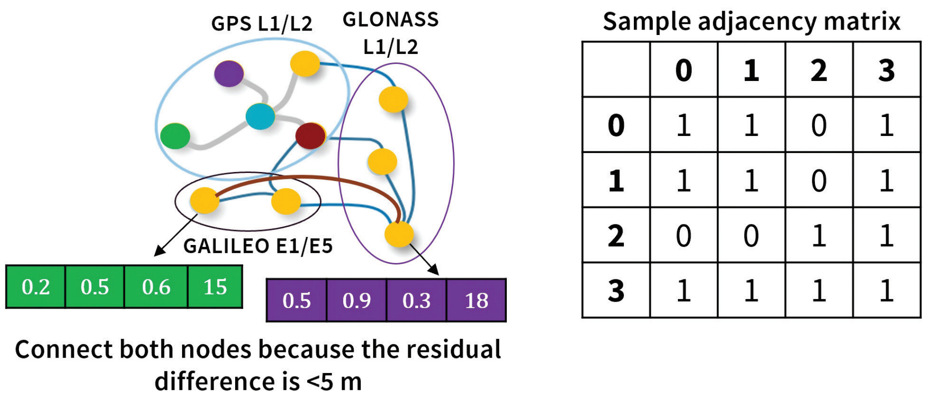
$$\begin{cases} A_{ij} = 1, & \text{if } sv_i \text{ and } sv_j \in C \text{ or } |(x_{iRES} - x_{jRES})| < \tau \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where C refers to the constellation type, τ is a hyperparameter that dictates the threshold for determining whether two nodes have similar measurement residuals, and i, j are two arbitrary nodes in the graph. This threshold is important for


Sample feature vector for a node
0.2 0.5 0.6 15
GPS L1/L2
GLO L1/L2
GAL E1/E5
Sample feature matrix for GPS cluster

0.2	0.5	0.7	0.1	0.3
0.5	0.9	0.8	0.6	0.7
0.6	0.3	0.6	0.4	0.5
15	20	5	10	2

FIGURE 4 A sample feature vector and feature matrix *X*
The feature vector contains the measurement residuals and the LOS vectors for every satellite computed with respect to the initial position from the Kalman filter. The feature matrix contains the feature vectors for all of the satellites, across all constellations and signal frequencies.



	0	1	2	3
0	1	1	0	1
1	1	1	0	1
2	0	0	1	1
3	1	1	1	1

FIGURE 5 A sample adjacency matrix
In our graph, we establish an edge between satellites if they either belong to the same constellation or have similar measurement residuals as determined by a threshold. Having dense connections in the graph allows the GCNN to perform aggregation and provide improved positioning corrections compared with a fully connected network.

connecting different clusters to leverage multi-constellation measurements and improve aggregation over the entire graph. Otherwise, the GCNN loses its discriminative power and is reduced to a simpler nonlinear network. A sample adjacency matrix is illustrated in Figure 5.

Convolution Layers and Prediction: There are many available choices of convolution layers. For our work, we choose graph isomorphism network (GIN) convolution (Xu et al., 2019) layers because these layers satisfy the WL graph isomorphism test and achieve maximum discriminative power among all other graph neural networks.

Instead of following the traditional node representation, as shown in Equation (2), the GIN operator updates the node representation as follows:

$$h_v^k = MLP^K(1+\epsilon^k)h_v^{k-1} + \sum_{u \in \mathcal{N}(v)} h_u^{k-1} \tag{7}$$

where MLP denotes a multilayer perceptron that can represent the composition of several functions, ϵ is a fixed scalar, h_v^k is the feature vector of node v at the kth layer, and $\mathcal{N}(v)$ is the set of nodes adjacent to node v.

We learn an aggregation function over the features associated with each node in the graph, akin to the concept of an attention module, as explored by Lee et al. (2019). There are multiple choices for the aggregator function, such as the mean aggregator, long short-term memory aggregator, and pooling aggregator. For this work, we perform a mean pooling across all of the graph nodes because our final output is a graph-level prediction instead of a node-level prediction. Pooling instantiates message passing among different nodes of the network, allowing the neighboring nodes to update their features and weights concurrently. Pooling also reduces the spatial resolution of the graph for subsequent layers and is mathematically described as follows:

$$h_v^k = ReLU(W \cdot mean(h_u^{k-1}, \forall u \in \mathcal{N}(v) \cup v)) \tag{8}$$

where $ReLU$ is the activation function and W is a learnable matrix containing weights for the previous layer. After aggregation, we pass the updated features at every node through a series of fully connected linear layers, which increases graph expressivity and improves our final prediction.

To infer the position correction, we train the entire graph using a standard Euclidean loss function or one of its derivatives. The loss function is constructed from the estimated positioning correction and the true correction obtained from the ground truth, as shown below:

$$Loss = \sum_{i=1}^{N} || \delta x_t - \Delta \mathbf{x}_t ||_2 \tag{9}$$

where the first term is the predicted correction from the GCNN and the second term is the true correction calculated from the Kalman filter position and the true receiver position. Note that we can construct this loss function only during training, as we do not have access to the true correction during inference.

3 | EXPERIMENTS AND SETUP

Training: We tested our proposed algorithm on the GSDC 2021 data sets (Fu et al., 2020) that contain GNSS code-phase, ADR, and Doppler measurements and ground truth from an integrated GNSS inertial navigation system. The setup for data collection and sample trajectories showing routes of the collected data are displayed in Figure 6.

Baselines and Metrics: We compared our proposed algorithm against two baselines: A WLS solution, which is a snapshot positioning method, and a Kalman filter solution, which is a temporal method. We tuned both baselines using Bayesian hyperparameter optimization to achieve maximum performance on the test data sets. For evaluation, we used quantitative metrics such as the mean, median,

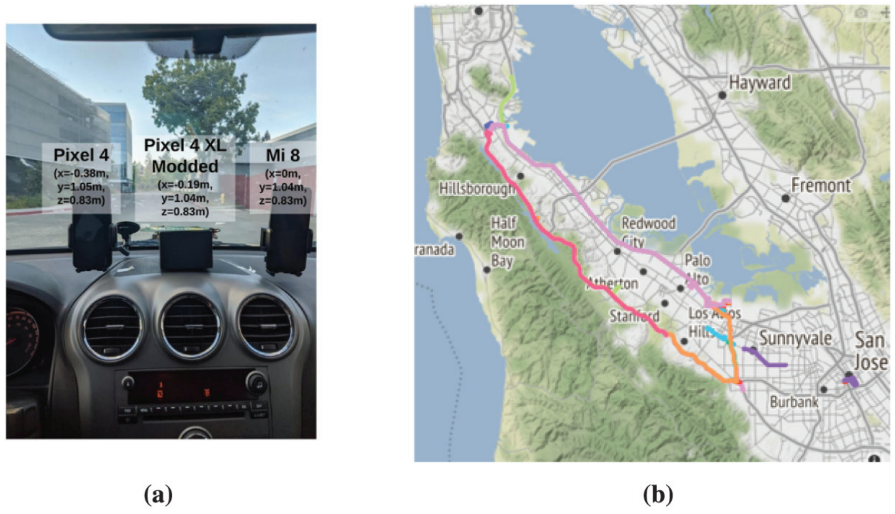


FIGURE 6 GSDC data sets: Setup and sample trajectories (Fu et al., 2020) (a) Setup of the smartphones in the cars during data collection (b) Ground-truth trajectories plotted from GSDC data sets for various cities

TABLE 1
Parameters of the GCNN Architecture

Module	Layer	Parameter
GINConv(1)	Linear	4×32
	ReLU*	-
	Linear	4×32
GINConv(2)	Linear	32×32
	ReLU	-
	Linear	32×32
	LayerNorm	32
Post-Message Passing	Linear	32×32
	Dropout	$p = 0.25$
	Linear	32×3

*rectified linear unit

maximum, and minimum horizontal positioning error. We also studied the distribution of positioning error for the baselines and our algorithm. For qualitative results, we compared the predicted trajectory from all of the algorithms with respect to the ground-truth trajectory.

Evaluation and Key Parameters: We split the GSDC data sets into 81 training data sets and 17 test data sets. Some of the test data sets contain data that were collected in cities previously absent from the training data sets. We made this design choice to enable us to stress-test our proposed approach. We leveraged the publicly available Pytorch Geometric tool (Fey & Lenssen, 2019) to design, train, and test the GCNN module of our framework. We trained the GCNN for 100 epochs using a mean squared error loss function and Adam optimizer (Kingma & Ba, 2015). We performed training across multiple hardware platforms such as Kaggle, Google Colab, and Amazon AWS using graphical processing unit and tensor processing unit accelerators for faster training. The network architecture and the number of parameters in each layer are shown in Table 1.

4 | RESULTS

We first analyzed the horizontal positioning error across all 17 unseen test data sets. As shown in Table 2, our algorithm outperforms both the snapshot method (WLS) and the temporal method (Kalman filter) for each metric. Our approach leads to the lowest mean, median, minimum, and maximum error on all 17 data sets.

We present qualitative results in Figure 7 via sample trajectory plots that were generated by plotting the position predictions from our algorithm and the Kalman filter baseline for the Mountain View data set. The figure indicates that our algorithm is able to closely track the ground-truth trajectory while showing improvement in regions that are depicted by the magnified plots. In these regions, we observe that the predictions from the Kalman filter baseline show some deviations. However, the GCNN module is able to fine-tune the correction and compensate for the deviations, leading to improved results from our algorithm.

For the next set of results, we only show results from the Kalman filter baseline for easier comparison, as the predicted positions from the WLS baseline always have the highest errors. We studied the distribution of the horizontal positioning error on selected test data sets. For this purpose, we consider both the Mountain View data set and an additional data set collected in Los Angeles.

TABLE 2
Summary of Positioning Error on Test Data Sets
Our algorithm outperforms the WLS and Kalman filter approaches across all 17 unseen test data sets.

Error Metric (m)	WLS	Kalman Filter	Our Approach
Mean	5.7	4.6	3.4
Median	4.4	3.9	3.3
Minimum	1.7	2.4	1.4
Maximum	25.5	7.8	5.6

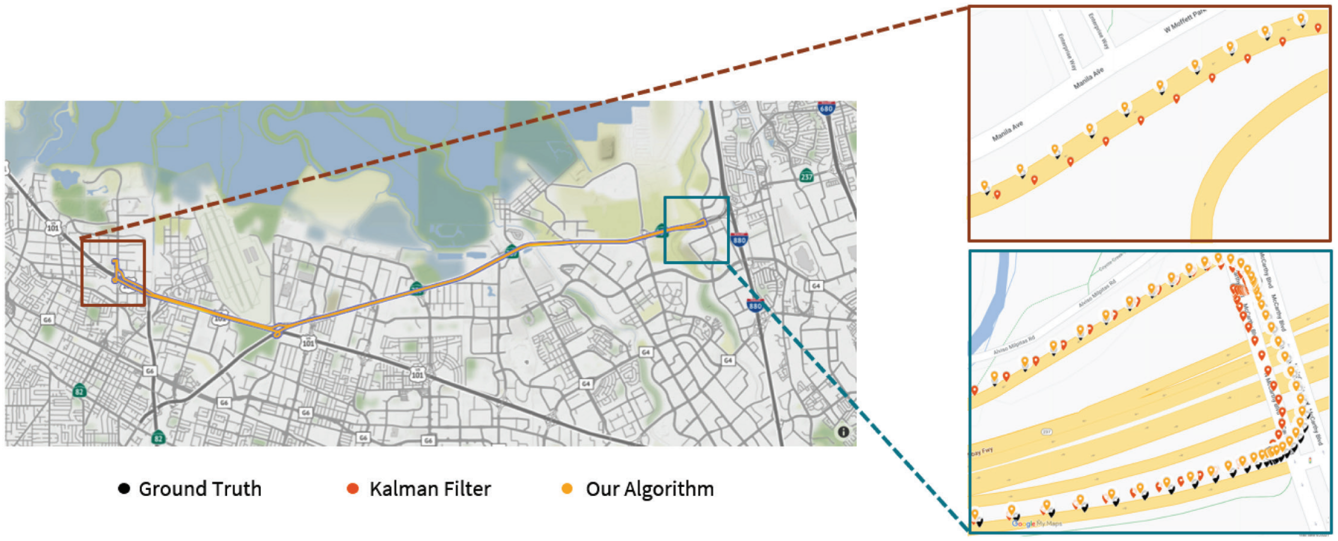


FIGURE 7 Trajectory tracking on one test data set from our algorithm and the Kalman filter baseline for the Mountain View data set
The left plot shows the entire trajectory as plotted in the city of Mountain View. The right plots show magnified maps of selected portions of the entire trajectory. Our approach follows the ground truth more closely than the estimated position obtained by the Kalman filter.

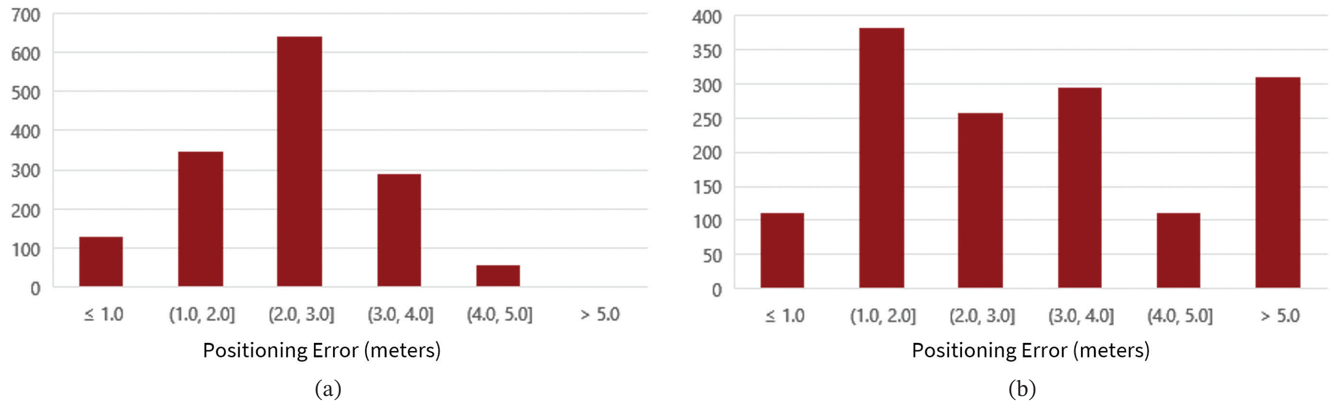


FIGURE 8 Horizontal positioning error on the Mountain View test data set (a) Our algorithm (b) Kalman filter baseline
Our approach has fewer outliers (>5 m error) and provides more accurate positioning compared with the Kalman filter baseline.

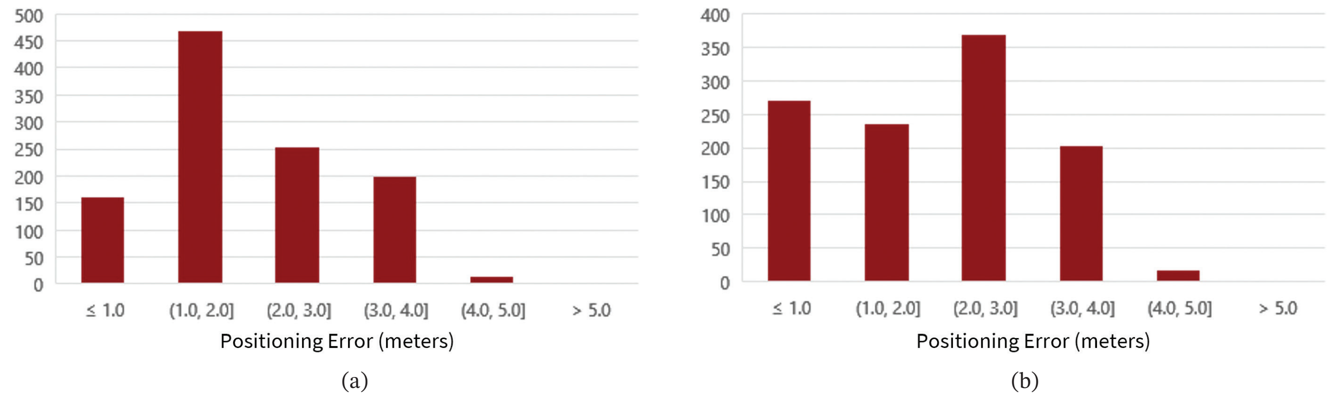


FIGURE 9 Horizontal positioning error on the Los Angeles data set (a) Our algorithm (b) Kalman filter baseline
Our approach has fewer outliers and provides more accurate positioning, even in cities previously unseen in the training data set.

Figure 8 shows the error distribution from our evaluation of the Mountain View data set. We observe that, compared with the Kalman filter baseline, our algorithm has few outliers, which are defined to be positioning errors > 5 m. Additionally, our algorithm provides more accurate positioning than the Kalman filter, as our error distribution has higher instances of error in the range of 0–3 m.

We also analyzed the error distribution on the Los Angeles data set, as shown in Figure 9. Similar to the Mountain View test data set, our approach provides positioning with fewer outliers than the Kalman filter baseline and shows improved positioning, with most errors in the range of 1–3 m.

5 | ADDITIONAL EVALUATION

We conducted a comprehensive evaluation of our proposed GCNN approach by benchmarking it against prior works that utilized neural-network-based corrections, specifically the study by DeepGNSS (Kanhere et al., 2022), on the GSDC data

TABLE 3
Comparison of Position Errors (in Meters) From Our GCNN With the Best Performing Neural Network From Kanhere et al. (2022)
Our algorithm outperforms this baseline in the north, east, and down directions in terms of the mean positioning error.

Approach	North	East	Down
Best Deep Neural Network from Kanhere et al. (2022)	6.4 ± 5.2	5.9 ± 5.0	6.2 ± 4.9
Our Proposed GCNN	2.0 ± 1.2	3.5 ± 0.9	4.9 ± 5.4

sets. We adopted a similar division strategy as performed in DeepGNSS, where the evaluation was conducted at the trace level.

Our evaluation process involved training our GCNN on 49 data sets comprising traces from various phones located in Mountain View. We then tested the performance of our approach in the cities of San Jose and Spring Valley Lake. Note that we utilized the updated version of the GSDC data sets from 2021 in the same cities.

To assess the accuracy of our proposed GCNN, we evaluated its performance using north–east–down corrections, following a methodology similar to that employed in DeepGNSS. The evaluation was conducted on the entire testing data set, allowing for a comprehensive analysis of our approach. For a fair comparison, we compared our results with the best-performing approach from DeepGNSS, which utilized data augmentation and an initialization range of 15 m.

The performance comparison of our GCNN approach against the best-performing approach from DeepGNSS is summarized in Table 3.

The results presented in Table 3 demonstrate the effectiveness of our approach. Our GCNN outperforms DeepGNSS in all three spatial directions: north, east, and down. Specifically, our GCNN achieves significantly lower position errors, with reductions of 4.4 m (north), 2.4 m (east), and 1.3 m (down) compared with the previous approach.

The improvement in the performance of our proposed GCNN approach compared with the previous set-based deep learning method can be attributed to several key factors. Firstly, our GCNN can capture and exploit spatial relationships in the input data. By leveraging the graph structure inherent in GNSS measurements, the GCNN is able to effectively model the dependencies and correlations between different satellites and their measurements. Secondly, our approach benefits from the integration of features and initialization from a Kalman filter. This combination of a model-based technique with the data-driven capabilities of the GCNN allows for a more comprehensive and robust approach to positioning. Lastly, the features used in our GCNN incorporate a wider range of information compared with the previous set-based deep learning method, such as ADR and Doppler smoothing of the pseudorange measurements and connections among satellites with similar measurement residuals. This enhanced feature set enables the GCNN to improve the positioning performance in both urban environments such as San Jose and suburban environments such as Spring Valley Lake.

6 | CONCLUSIONS

We have designed a hybrid approach for inferring position corrections from smartphone GNSS measurements using a GCNN and an initial position estimate from a Kalman filter. For the GCNN, we created features using measurement residuals and LOS vectors from multiple GNSS constellations and multiple signal frequencies after applying carrier smoothing. The GCNN performed aggregation across all available satellites and learned the position correction in an end-to-end manner. We evaluated our approach on real-world data sets collected in urban environments. Our approach demonstrated improved positioning accuracy and error distribution when compared against model-based approaches such as WLS and Kalman filter methods. Thus, our approach is a promising fusion of existing model-based and data-driven methods for achieving high smartphone positioning accuracy in urban environments.

ACKNOWLEDGEMENTS

We would like to thank Google for releasing the smartphone data sets for 2020 and 2021. We would also like to acknowledge Zach Witzel and Nikhil Raghuraman for their help with debugging and Adam Dai and Daniel Neamati for reviewing the paper. Lastly, we would like to thank members of the Stanford NAVLab for their insightful discussions and feedback.

REFERENCES

Beiber, D. (2019). Weisfeiler-Lehman isomorphism test. *Online blog post.* https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/

Bochkati, M., Sharma, H., Lichtenberger, C. A., & Pany, T. (2020). Demonstration of fused RTK (fixed) + inertial positioning using android smartphone sensors only. *Proc. of the IEEE/ION Position, Location and Navigation Symposium (PLANS)*, Portland, OR, 1140–1154. https://www.doi.org/10.1109/PLANS46316.2020.9109865

Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *Proc. of the International Conference on Learning Representations Workshop on Representation Learning on Graphs and Manifolds.* New Orleans, LA. https://doi.org/10.48550/arXiv.1903.02428

Fu, G. M., Khider, M., & van Diggelen, F. (2020). Android raw GNSS measurement datasets for precise positioning. *Proc. of the 33rd International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS + 2020),* 1925–1937. https://doi.org/10.33012/2020.17628

Guangcai, L., & Jianghui, G. (2019). Characteristics of raw multi-GNSS measurement error from Google Android smart devices. *GPS Solutions*, *23*, 1–16. https://doi.org/10.1007/s10291-019-0885-4

Han, K., Lee, S., Song, Y.-J., Lee, H.-B., Park, D.-H., & Won, J.-H. (2021). Precise positioning with machine learning based Kalman filter using GNSS/IMU measurements from android smartphone. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021),* St. Louis, MO, 3094–3102. https://doi.org/10.33012/2021.18005

Humphreys, T. E., Murrian, M., Diggelen, F. V., Podshivalov, S., & Pesyna, K. M. (2016). On the feasibility of cm-accurate positioning via a smartphone's antenna and GNSS chip. *Proc. of the 2016 IEEE/ION Position, Location, and Navigation Symposium (PLANS)*, Savannah, GA, 232–242. https://doi.org/10.1109/plans.2016.7479707

Jian, D., Shi, J., Kar, S., & Moura, J. (2018). On graph convolution for graph CNNS. *Proc. of the 2018 IEEE Data Science Workshop (DSW)*, Lausanne, Switzerland, 1–5. https://doi.org/10.1109/DSW.2018.8439904

Kanhere, A. V., Gupta, S., Shetty, A., & Gao, G. (2022). Improving GNSS positioning using neural network-based corrections. *NAVIGATION*, *69*(4). https://doi.org/10.33012/navi.548

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proc. of the 3rd International Conference on Learning Representations.* San Diego, CA. https://doi.org/10.48550/arXiv.1412.6980

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Poster presentation at the International Conference on Learning Representations (ICLR).* Toulon, France. https://doi.org/10.48550/arXiv.1609.02907

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., & Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. *Proc. of the International Conference on Machine Learning (ICML),* Long Beach, CA, 3744–3753. https://doi.org/10.48550/arXiv.1810.00825

Mohanty, A., & Gao, G. (2022). Learning GNSS positioning corrections for smartphones using graph convolution neural networks. *Proc. of the 36th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2022),* Denver, CO, 2215–2225. https://doi.org/10.33012/2022.18372

Realini, E., Caldera, S., Pertusini, L., & Sampietro, D. (2017). Precise GNSS positioning using smart devices. *Sensors*, *17*(10), 2434. https://doi.org/10.3390/s17102434

Siemuri, A., Selvan, K., Kuusniemi, H., Välisuo, P., & Elmusrati, M. S. (2021). Improving precision GNSS positioning and navigation accuracy on smartphones using machine learning. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021),* St. Louis, MO, 3081–3093. https://doi.org/10.33012/2021.18004

Suzuki, T. (2021). First place award winner of the smartphone decimeter challenge: Global optimization of position and velocity by factor graph optimization. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021),* St. Louis, MO, 2974–2985. https://doi.org/10.33012/2021.18109

van Diggelen, F., & Wang, W. (2018). How to achieve 1-meter accuracy in android. *GPS World Online Blog Post*. https://www.gpsworld.com/how-to-achieve-1-meter-accuracy-in-android/

Weisfehler, B., & Leman, A. (1968). The reduction of a graph to canonical form and the algebra which appears therein. *Proc. of the Nauchno-Technicheskaya Informatsia* (Vol. 9). https://api.semanticscholar.org/CorpusID:49579538

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? *Proc. of the International Conference on Learning Representations.* New Orleans, LA. https://doi.org/10.48550/arXiv.1810.00826

Zhang, K., Jiao, F., & Li, J. (2018). The assessment of GNSS measurements from android smartphones. *Proc. of the China Satellite Navigation Conference (CSNC),* Harbin, China, 147–157. http://dx.doi.org/10.1007/978-981-13-0029-5_14

How to cite this article: Mohanty, A., & Gao, G. (2023). Learning GNSS positioning corrections for smartphones using graph convolution neural networks. *NAVIGATION, 70*(4). https://doi.org/10.33012/navi.622