future internet

MDPI

Article

No Perfect Outdoors: Towards a Deep Profiling of GNSS-Based Location Contexts

Jin Wang and Jun Luo *

School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore; jwang033@ntu.edu.sg
* Correspondence: junluo@ntu.edu.sg

Abstract: While both outdoor and indoor localization methods are flourishing, how to properly marry them to offer pervasive localizability in urban areas remains open. Recently, proposals on indoor–outdoor detection have made the first step towards such an integration, yet complicated urban environments render such a binary classification incompetent. Fortunately, the latest developments in Android have granted us access to raw GNSS measurements, which contain far more information than commonly derived GPS location indicators. In this paper, we explore these newly available measurements in order to better characterize diversified urban environments. Essentially, we tackle the challenges introduced by the complex GNSS data and apply a deep learning model to identify representations for respective location contexts. We further develop two preliminary applications of our deep profiling: one, we offer a more fine-grained semantic classification than binary indoor–outdoor detection; and two, we derive a GPS error indicator that is more meaningful than that provided by Google Maps. These results are all corroborated by our extensive data collection and trace-driven evaluations.

Keywords: indoor-outdoor detection; GNSS measurements; deep learning

Citation: Wang, J.; Luo, J. No Perfect Outdoors: Towards a Deep Profiling of GNSS-Based Location Contexts. *Future Internet* **2022**, *14*, 7. https://doi.org/10.3390/fi14010007

Academic Editor: Michael Sheng

Received: 18 November 2021
Accepted: 21 December 2021
Published: 23 December 2021

1. Introduction

Existing localization/navigation systems are clearly separated into two categories: outdoor [1,2] versus indoor [3,4]. Normally, outdoor systems mostly rely on GPS, while indoor systems leverage other pervasively available signal sources, such as WiFi [3–5], visible light [6,7], and geomagnetism [8,9]. A strong implication behind such a clear-cut separation is the ability to differentiate indoor from outdoor, so as to trigger timely switches between these two systems; this has provoked several research proposals on detecting the transition from outdoor to indoor or vice versa (a.k.a. *IO-detection*) [10–12]. Nonetheless, though perfect indoor scenarios (wrapped by concrete walls and roof entirely) do exist, GPS signal can grow very weak while walking in downtown areas, while users inside a room with big windows may still obtain very good GPS signal. In short, our daily experience tells us that *there are no perfect outdoor scenarios for urban areas*.

Exactly due to these behaviors of received GPS signals, it was recently suggested to either leverage GPS readings for indoor localization [13,14] or complement GPS with WiFi readings for outdoor localization [15,16]. Nonetheless, the question about how much trust we may put on each signal source for localization (in particular, to use GPS or not) remains open. Apparently, the outcome of IO detection can be insufficient, given diversified urban environments, so we should better look into a holistic profiling (i.e., identifying semantic representations) of *location context*: *location-related contextual information that affects GPS performance*. In particular, location context concerns various scenarios lying between perfect outdoor areas (e.g., large parking areas outside of shopping malls) and perfect indoor spaces (e.g., windowless research laboratories surrounded by thick walls). Such a profiling may eventually allow us to fuse various signal sources for achieving ideal urban localization that integrates both the indoors and outdoors.

In the past, such an analysis was nearly impossible for users with commercial devices (e.g., smartphones), because users can only access the GPS location indicators as distilled through sophisticated filtering processes. Fortunately, new devices [17], supported since Android 7, open up the access to raw GNSS measurements. (global navigation satellite system (GNSS) is a generic term referring to satellite navigation systems such as GPS and Galileo; however, here, raw GNSS measurements denote the standard data format shared by all such systems, though we actually obtain (only) GPS–GNSS measurements). Although the GPS location indicators are derived from these GNSS measurements, when substantial information is lost, it may have been caused by the significant dimension reduction resulting from conventional signal processing pipelines. Therefore, it is reasonable to believe that effectively mining a large volume of these raw measurements could end up with a much more holistic profiling in urban environments than with binary IO-detection. With this initial attempt in profiling, we can not only characterize a location semantically (which an erroneous location indicator likely fails to achieve), but also gain a better understanding (or even correction) of the large number of location errors observed in urban areas.

To this end, we thoroughly conduct GNSS data collection in a major metropolitan city, targeting representative scenarios such as sheltered walkways, urban canyons, buildings with an open ground level, and so on. Our aim is to process these data using effective machine learning tools, so as to distill latent representations out of them. These representations, on one hand, are used to construct semantic profiles for characterizing various urban location contexts, and on the other, they can be used to provide a better accuracy indicator than, for example, the "blue disk" offered by Google Maps. One of the major challenges in mining GNSS data is its high dimensionality: each record of GPS–GNSS measurements includes up to 32 satellites, with several tens of fields for each of them. This is significantly different from the time-series or image inputs to machine learning algorithms. Another major challenge lies in the inherent misalignment in feature space over time: the same location can be served by very different satellites at various points in time. This is also the reason that we consider profiling locations *context*s (rather than merely locations), as other factors beyond pure locations should be taken into consideration. In order to make sense out of these data, we make the following main contributions:

- We, for the first time, propose to profile location context via mining raw GNSS measurements.
- We innovate in the feature engineering pipeline to handle both high-dimensionality and misalignment.
- We apply a deep autoencoder to distill compressed representations from the GNSS data sets.
- We develop two preliminary applications of our deep profiling; they deliver semantic representations and better location error estimations, respectively.
- We perform extensive data collection and experimentation to validate our proposals.

Note that we are not aiming to deliver yet another optimized GPS [2,12,18], although a future research direction could be combining conventional signal processing techniques with machine learning schemes. Instead, our main target is to demonstrate to the community that we have left out a substantial amount of information hidden in GNSS data by deriving only location indicators. Furthermore, we also deliver some preliminary outcomes of mining such latent information, hoping to evoke more mature applications.

2. Background and Challenges

In this section, we survey a few topics closely related to localization and (mobile) data mining. This motivates us to consider mining GNSS data, while suggesting a few major challenges in achieving it. Due to space limitations, we will not discuss general indoor localization proposals (e.g., [5–7]) that are only marginally relevant.

2.1. GPS-Based Localization

As the earliest and most representative GNSS, GPS has been in operation for more than 20 years, and it has been continuously optimized at both higher and lower levels since then. In order to improve energy efficiency, both EnTracked [18] and RAPS [19] add adaptivity by leveraging the estimation/prediction from historical GPS readings. Others (e.g., A-Loc [20]) also suggest fusing other sensing information to assist GPS so as to reduce its energy consumption. Thanks to these proposals and related engineering efforts such as A-GPS [21], the latest GPS modules in our smartphones have become much more energy efficient than those studied a decade ago [22], especially when only GNSS measurements are retrieved, as we will show in Section 5.5 using real-life data.

Optimization at the lower level focuses on signal processing and computation. In particular, signal sparsity has been exploited in both [2,23] to speed up computation and thus save energy. To improve localization accuracy with low-cost receivers, collaboration computation has been introduced by [24] for a scalable network of single-frequency receivers. Further computation reduction can be achieved by cloud offloading [25]; the GPS receiver collects only a small number of raw GPS readings while moving the location estimation to the cloud. In fact, the raw GPS readings contain a fairly large amount of information; simply summarizing them into three-axis location indicators using model-based methods almost surely causes information loss. Our attempt to process these raw data using machine learning schemes is the first step towards their more holistic utilization.

2.2. Indoor–Outdoor Detection

Differentiating indoor and outdoor scenarios was the first application of the data analytic approach on GPS readings. In fact, earlier approaches to IO-detection [10,11] criticize using simple data analytics on the number of visible satellites as being slow and energy inefficient. As a result, the authors applied various machine learning techniques to fuse non-GPS sensor readings, while [10] applied a naive decision tree, and [11] has attempted several semi-supervised learning techniques. Nevertheless, classical machine learning techniques are too sensitive to feature selection, which can be an obstacle when facing sophisticated data inputs. Recently, SatProbe [12] revisited the number of visible satellites for IO-detection, and proposed efficient algorithms to extract this feature. However, the access to raw GNSS measurements, introduced in Android 7, provides users with much more information (e.g., the spatial relation among visible satellites) than just this counter-based feature. Moreover, existing approaches all categorize urban scenarios into three classes: outdoor, semi-outdoor, and indoor—this, apparently, oversimplifies our real-life location context.

2.3. Non-GPS Outdoor Localization

Exactly due to the unstable performance of GPS in urban areas, many have considered supplementary solutions [15,16,26,27]. Essentially, due to the proliferation of mobile and wireless communications, urban areas have been densely covered by both cellular networks [26,27] and WiFi hotspots [15,16], and these communication facilities have provided extra signal sources for location estimation. Experience tells us that fusing GPS and non-GPS signal sources properly could yield higher localization accuracy, but this can be achieved only with an adequate profiling of location context.

2.4. Mining Mobile Sensing Data

Thanks to the latest mobile devices being equipped with numerous sensors, we can obtain a huge volume of sensing data, the mining of which has enabled quite a few applications, including, notably, localization (e.g., [3,4,9,16]) and activity recognition (e.g., [28–31]). Most of these proposals follow a classical machine learning pipeline, starting by empirically selecting a few features and ending with a mapping from these features to certain known patterns. In order to relieve the reliance on pre-knowledge, Lasagna [31] was among the first to apply deep learning technique for activity recognition. Nevertheless,

the data inputs considered so far are simply time series (e.g., gyroscopes deliver intensity readings on three axes), whereas the GNSS data we shall consider have a much more complicated structure, rendering existing proposals inapplicable.

2.5. Challenges in Mining GNSS Data

While the above discussions indicate the necessities and potentials of mining GNSS data, several challenges need to be overcome to complete this task. Before a more comprehensive study of GNSS data in Section 3.1, we briefly hint at two major issues here.

- **Beyond dimensionality**: the involved data structure is fairly complicated, as shown in Table 1 for one satellite. One could deem it a high-dimensional vector and simply stack 32 of them (for all GPS satellites) into a matrix, yet such an input data structure is inefficient: many fields (features) can be highly redundant, yet we are expecting a compact and effective feature set for each satellite, which enables a learning module to efficiently capture location contexts under profiling.
- **Inherent misalignment**: No matter the data structure used, proper indexing is key to aligning relevant features. Whereas conventional mobile sensing has sensor IDs as a natural indexing, using satellite IDs for the same purpose can severely misalign the target feature: many IDs are rarely visible and the same location may witness very different satellite coverage at different points in time, while even the same number of visible satellites may form various patterns and, thus, result in very different localization performances.

Table 1. Raw GNSS data structure.

Fields Name	Description
Binary Fields	
STATE_SYMBOL_SYNC	is tracking synchronized at the symbol level?
STATE_TOW_DECODED	is the time of week known?
STATE_BIT_SYNC	is tracking synchronized at the bit level?
MultipathIndicator	has a multipath been detected?
...	...
Scalar Fields	
ReceivedSvTimeNanos	the received GNSS satellite time in ns
BiasUncertaintyNanos	the clock's bias uncertainty in ns
PseudorangeRateUncertainty...	rate uncertainty (1-sigma) in m/s.
Cn0DbHz	the carrier-to-noise density in dB-Hz.
...	...

3. From Understanding GNSS Data to Problem Definition

In order to get a better understanding of the problem and its solution's feasibility, we have performed extensive studies on the GNSS data and their corresponding location contexts (roughly defined in Section 1, but to be elaborated further). In this section, we will firstly present a few key observations, which motivates a concrete problem definition and the subsequent solution methods as well.

3.1. Case Study of GPS-Based Localization Performance

During our extensive survey in a city, we have collected the following data for various location scenarios: (i) raw GNSS measurements, (ii) Android GPS location indicators, (iii) ground truth location indicators (manually labelled), and (iv) Google Maps snapshots. We briefly report a few typical results in Figure 1, where we choose to put the overlay of (ii) and (iii) onto (iv) in the first row, along with the corresponding (photographic)

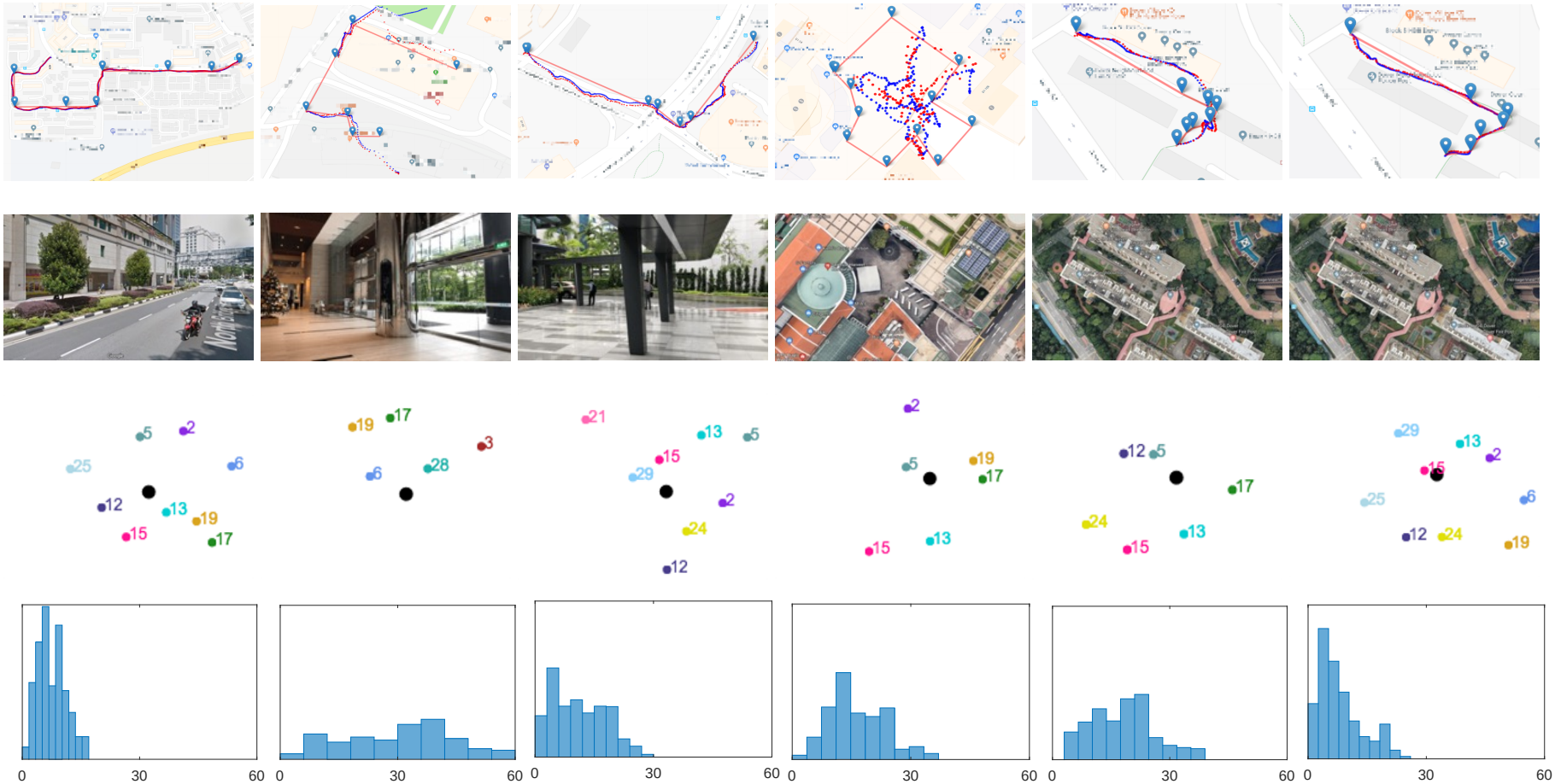
views in the second row, then (i) alone in the third row, and finally the errors distribution derived from (ii) and (iii) in the last row. As it is impossible to present all GNSS data in one figure, we show the derived location patterns for visible satellites, as inspired by constellation-based astronomical localization [32]. The following are a few key observations on these results.

- **Satellite pattern matters**: a wide street, shown in Figure 1a, enjoys a balanced satellite pattern and thus an almost perfect (for commercial GPS) average localization error of 5–10 m, but semi-exposed indoor areas Figure 1b (glass-covered building or void deck) cause a biased pattern and larger variations in GPS errors. A balanced pattern obtained for a sheltered street, shown in Figure 1c, again leads to consistent GPS performance, though the location scenario is quite close to what would be considered indoor.
- **Correlations between GNSS data and locations exist**: besides the discussions in the previous bullet, a high-rise-surrounded square or narrow street (scenarios often termed "building well" or "urban canyon"), as shown in Figure 1d, leads to a more skewed pattern, as well as a more severe multipath effect (another GNSS field not shown in the figure). Supposing we may properly differentiate Figure 1a from Figure 1d via contextual information inferred by GNSS data, adequate corrections by other complementary signal sources can be put into action. Note that such a differentiation is achieved without deriving GPS location indicators that could indicate locations erroneously.
- **Other GNSS features may help further**: differentiating Figure 1a from Figure 1d may sometimes require more features than provided by the satellite pattern itself. In addition, even when the location remains roughly the same, Figure 1e,f exhibit very different GPS tracking performance during two distinct time periods, indicating a change of contextual information beyond pure location. Nevertheless, attributing the context change solely to satellite pattern may not be sufficient; a holistic mining of several GNSS features is necessary.

Essentially, our preliminary studies have demonstrated the necessity and benefit of profiling *location context* that mainly involves a time-independent location scenario and time-related GNSS features. Moreover, the importance of satellite pattern (and its independence on satellite IDs) may have suggested an effective way to re-organize the GNSS data so as to enable effective mining.

3.2. Problem Definition

By now, we can clearly define the problem we aim to tackle in this paper. The major task is to profile location context based on raw GNSS measurements. In particular, we consider a large volume of GNSS data records collected by various GPS receivers (of capable smartphones), each *record* containing *sample*s obtained within a time segment $\mathcal{T}$ but from one of $\mathcal{N}$ visible satellites and each sample includes $\mathcal{M}$ fields. Our goal is to first process and pack all the records of dimension $\mathcal{T} \times \mathcal{M} \times \mathcal{N}$ into a valid feature map; then, we identify a *semantic representation* through unsupervised learning, aiming at compactly characterizing these seemingly sophisticated data structures.



(**a**) Outdoor along street (**b**) Semi-exposed indoor (**c**) Sheltered street (**d**) High-rise enclosing (**e**) Between buildings (**f**) Between buildings

Figure 1. Preliminary studies on GNSS data. (**a**) Outdoor, long, wide street; (**b**) inside a glass building and on a void deck, (**c**) sheltered street; (**d**) enclosed, high-rise area; (**e**) building surrounded by street; and (**f**) the same as (**e**) but at a different time. We put ground-truth and GPS traces on Google Maps snapshots in the first row, the corresponding photographic views in the second, then satellite distribution patterns (with black dot indicating the position of receiver) in the third, and finally GPS error distributions (in meters) in the last.

The outcome of this profiling process should yield a representation that satisfies the following criteria to ensure its effectiveness:

1. It can be further derived and used to estimate the accuracy level of GPS localization performance.
2. It has specific semantic meaning that can be further clustered or classified to represent various urban location contexts.
3. Its dimension should be sufficiently reduced, so as to be computationally efficient for other applications.

Although the ultimate purpose of this profiling is to assist an integrated localization service, we would refrain from being over-ambitious in this preliminary study. Instead, we use the above requirements (1) and (2) to act as two immediate applications, in order to demonstrate the usefulness of our proposed GNSS profiling.

4. GNSS-Based Location Context Profiling

In this section, we present a specially designed pipeline to transform raw GNSS measurements into compact data structures that is adapted to deep learning models. Then we engineer an autoencoder model to perform location context profiling based on these data. We further evaluate the effectiveness of this profiling with a supervised task, i.e., localization accuracy estimation, and an unsupervised task, i.e., semantic analysis on location contexts.

4.1. Feature Engineering Pipeline

We next provide the details of the whole pipeline for extracting context features.

4.1.1. GNSS Data from Android 7.0

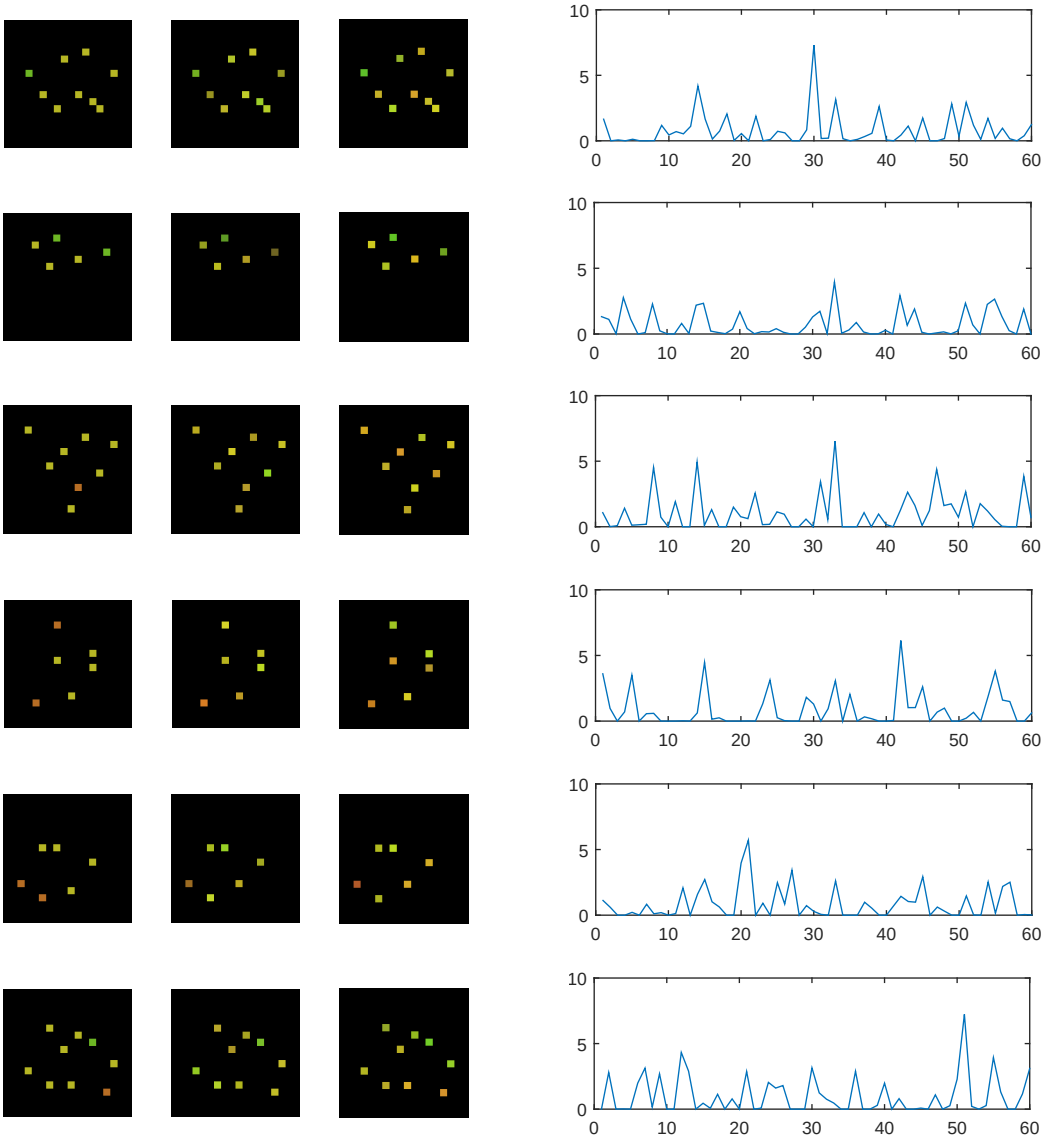
Previously, GPS modules handled all tasks, from receiving and processing signals to calculating pseudoranges and, thus, estimating locations; they only returned estimated locations and their corresponding accuracy levels to front-end applications. Starting from Android 7.0, raw GNSS measurements became available to users on a number of devices [17]. Each sample of a GNSS measurement includes three general types of fields: (i) state indicators that describe clock synchronization, multipath effects, and hardware clock continuity; (ii) raw measurements, such as carrier frequency, cycles, and phases; and (iii) computed measurements, such as the uncertainty of some raw measurements. A detailed list of fields can be found in [33]. By invoking the GNSS raw measurement API, a sample including all these fields is returned from every visible satellite. In order to facilitate our learning purpose, we regroup them into two categories of features (part of them shown in Table 1) and also derive a few new features. We briefly explain them in the following:

- **Binary features** take only boolean values and they are related to the clock status, carrier phase validity, and multipath effect. Some of these features may not be directly applicable to the profiling, yet they indicate the validity of other features.
- **Scalar features** take float scalar values and primarily include measurements concerning information quality. For example, there are measurements concerning SNR and Doppler shift, along with their respective uncertainty fields providing 1-σ values. Although these features are often used to infer localization accuracy, given a model-based approach, we intend to use them in a non-parametric manner to avoid losing information.
- **Derived features** are not directly obtained from Android API; we compute them, mainly to facilitate reshaping the input to deep learning modules. In particular, *SvPosition* is the position (under a spherical coordinate system) of a satellite when a GNSS sample is received, derived using the GPS ephemeris [34] and the receiving timestamps.

4.1.2. Satellite Position-Based Feature Packing

As we have elaborated in Section 2.5, packing samples for all satellites into one compact and meaningful representation that is highly non-trivial. Fortunately, the preliminary

studies in Section 3.1 indicate that the satellite pattern might be a good way of indexing the complex data structure. Further inspired by the fact that most deep learning models take images (i.e., 2D pixel maps) as a common input, we design a pixel-position-based packing mechanism. Essentially, we represent each feature of all visible satellites as a *snapshot*, i.e., an image with certain pixels representing the satellites and the other pixels taking their values from the features. Each snapshot is centered at the receiver's true location, and the relative satellite positions derived from *SvPosition* are normalized to determine the corresponding pixel positions. As a result, each sample (for all satellites) contains $\mathcal{M}$ snapshots and a record has $\mathcal{T} \times \mathcal{M}$ of them. We could further pack the temporal dimension into these snapshots, but that would require a much higher resolution and may also lose track of temporal correlations. We show a few snapshot samples in Figure 2a.



(**a**) Snapshots at $\tau = 1, 2, 3$ (**b**) Representation $\mathbf{v}$

Figure 2. Snapshots and corresponding representations (the x-axis is the feature index and the y-axis represents the feature magnitude, normalized to a scale of $[0, 10]$).

To ensure the scale and trends of features are consistent, normalization and re-ranking are done before packing them into snapshots. The normalization phase transforms feature values within the range $[0, 1]$—it should produce a "scattered" distribution for all features—

and we use stereographic projection to map the celestial sphere to a plain. In particular, most uncertainty features follow a log-normal distribution, so we convert them logarithmically to obtain a quasi-normal distribution. The re-ranking phase makes sure all the features at each pixel have a consistent trend in their physical meanings. Since pixel values are in greyscale (i.e., non-negative integers) and the zero value indicates no valid data presented, a smaller value should correspond to greater uncertainty. Therefore, all the uncertainty and ambiguity features are reversely ordered to maintain consistency.

4.2. Location Context Autoencoder

To learn a compact representation from the packed features (i.e., image sequences), we naturally choose an autoencoder for embedding [35], and the compact representations are extracted from its central layer. Sequence inputs are handled by incorporating a recurrent neural network (RNN) as the fundamental structure [36]. Given a sequence of inputs and a desired output sequence, the goal of the RNN is to estimate the conditional probability of this output sequence given the input sequence. Incorporating the two structures, we design our autoencoder as a sequence-to-sequence structure with images as input. Both the encoder and the decoder are implemented using gated recurrent unit (GRU) cells. As shown in Figure 3, given the snapshots from time 0 to $\mathcal{T}$, denoted as $\mathbf{x} = (x_0, \cdots, x_\tau \cdots, x_\mathcal{T})$, where $x_\tau \in \mathbb{R}^{n_i \times n_i \times \mathcal{M}}$, n_i is the size of a snapshot, and $\mathcal{M}$ is number of GNSS features, we want to get a compact representation $\mathbf{v} \in \mathbb{R}^\ell$ with ℓ being the latent dimension of GRU (shown as red in Figure 3). The encoder and decoder are defined as transformations Θ and Φ. Here, $\Theta : \mathbf{x} \to \mathbf{v}$, $\Phi : \mathbf{v} \to \mathbf{x}'$, where $\mathbf{x}'$ denotes the reconstructed sequence of $\mathbf{x}$ with the same dimension. The training is conducted with a back-propagation algorithm to find transformations Θ^* and Φ^* based on the following objective:

$$\{\Theta^*, \Phi^*\} = \arg\min_{\Theta,\Phi} \sum_{\tau=0}^{\mathcal{T}} \|x_\tau - x'_\tau\|_2^2.$$

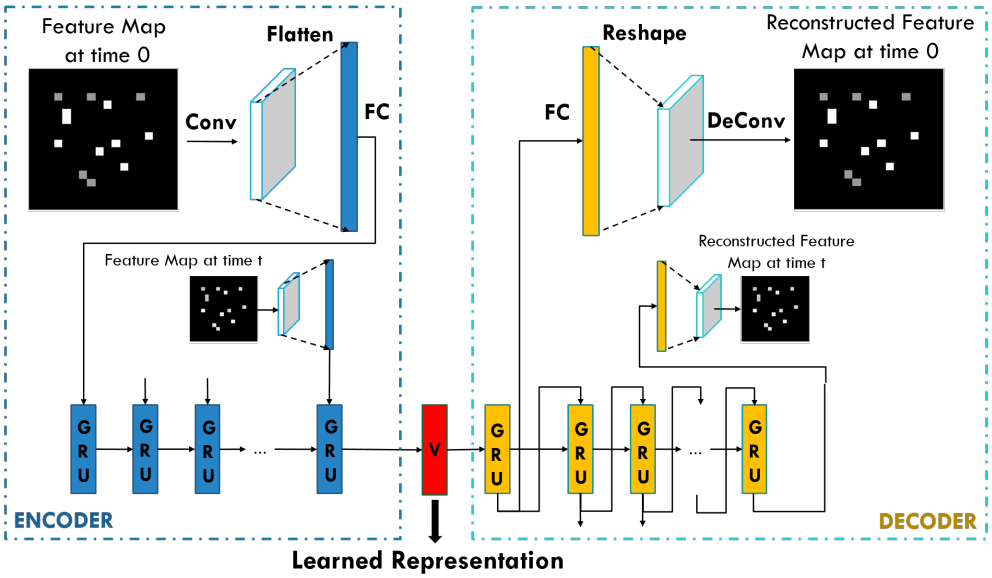


Figure 3. Autoencoder structure.

The encoder first applies a convolution layer with filter size (m_f, m_f, ℓ) to each x_t and flattens it into a vector as an input for the GRU cell. At the decoder side, the semantic vector $\mathbf{v}$ is set as the hidden state, the output generated by the RNN is reshaped, and a deconvolutional layer applied, with the same filter size as applied at the encoder side to obtain the final output x'_t. The parameters setting for m_f, ℓ and $\mathcal{T}$ will be studied in Section 5.2.1. Figure 2 shows examples of snapshots and correspondingly generated

representations, based on the six scenarios in Figure 1. Due to space limitations, we only visualize the first three steps in a sequence of length five, and select three features to be the RGB channels of images.

4.3. Applications of Location Context Profiling

In this section, we demonstrate two applications with profiling: (i) a localization error estimator, and (ii) a semantic context search.

4.3.1. Localization Error Estimator

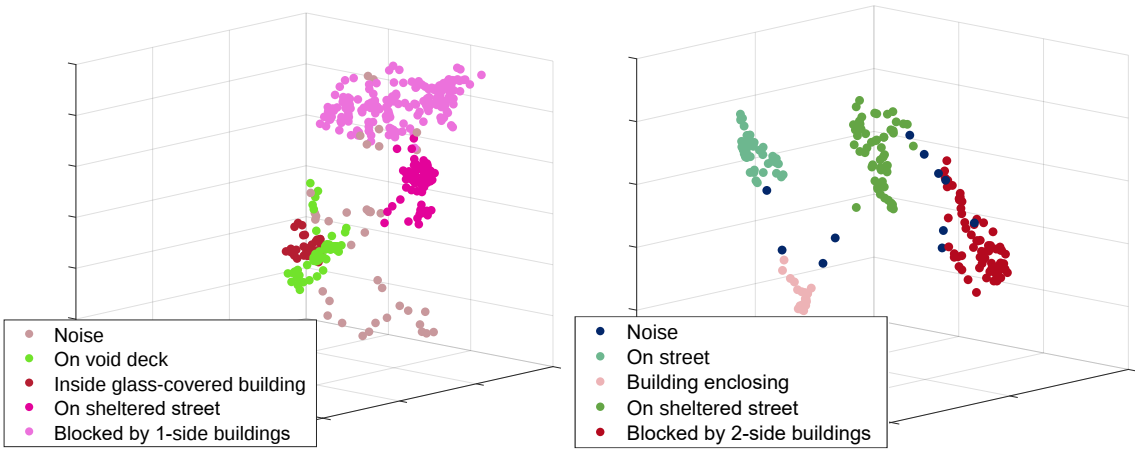
As shown in Section 3.1, different contexts can result in various error distributions. So, the semantic vector $\mathbf{v}$ can be used to estimate localization performance. Such information can yield the confidence level of a GPS localization, similar to the "blue disk" provided by Google Maps. At any time t, given the representation $\mathbf{v}_t$ generated from GNSS measurements during previous $\mathcal{T}$ time steps, an error estimation function Λ and true GPS error e_t, and the predicted error $\hat{e}_t = \Lambda(\mathbf{v}_t)$, we aim to find $\Lambda^* = \arg\min_\Lambda \|e_t - \hat{e}_t\|_2^2$. Using this objective as the loss function, we design a four-layer feed-forward deep neural network structure with a single neuron at the last layer as our error estimator model. We denote it by DNN-R hereafter.

We also implement two other deep learning structures, namely DNN-C and RNN, to compare with DNN-R. DNN-C converts the regression problem into a classification problem, in which we adopt a classification layer before the regression neuron layer, and make the final layer as the expectation on the class distribution. RNN implements GRU cells to include estimation from previous steps; it adapts to the filtering effect inside GPS that results in relatively smooth changes in location estimations and corresponding errors. Detailed settings are presented in Section 5.3.

4.3.2. Context Semantic Analysis

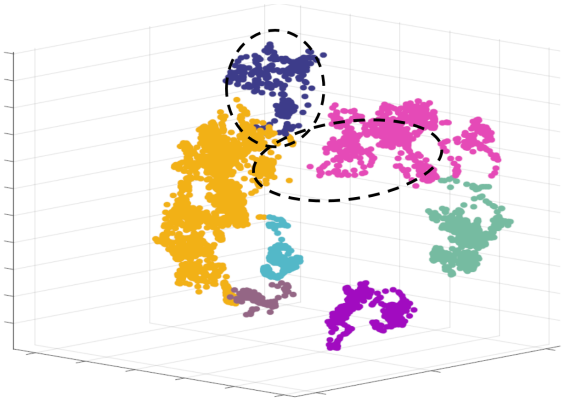
Our studies in Section 3.1 also suggest that GNSS measurements can represent the context of environments, which can be used to trigger sensor fusion for better localization. We aim to create a context database containing labelled training representations and allow a new representation to query for a context label. Since our representation is learned from records, a semantic label is assigned to a trip segment (of length $\mathcal{T}$) rather than a spot sample. Moreover, manually labeling representations can be meaningless, given the great variety in human perception, so we adopt a bottom-up clustering mechanism to build up the labeled database. Each training representation has two levels of labels.

- **Lower-level clustering**: For a set of representations, we use DBSCAN to cluster them into small clusters, assign each cluster with a predefined label as the *detailed context* and calculate the center of the cluster. Detected noise is labeled as "noise". Lower-level clustering captures scenario information, such as "on street", "blocked one one side by building", "building enclosing", and "on void deck". Figure 4a,b demonstrate two traces after lower-level clustering.
- **Higher-level clustering**: Cluster centers and "noises" from lower-level clustering are further combined into upper-level clusters, and assigned a unified label, such as *general context*. Higher-level clustering captures general information, such as satellite patterns. The noise vector detected from the upper level of clustering are treated as a separate cluster, since it may capture an infrequent satellite pattern at the moment, but can be further clustered into other groups when the database is enlarged. To illustrate the correspondence with lower-level clustering, Figure 4c shows the representations with the color label of upper-level clusters.



(**a**) Detailed context for trace 1

(**b**) Detailed context for trace 2



(**c**) General Context

Figure 4. Context semantic analysis in (unitless) t-SNE plots.

At the query stage, each representation will be firstly matched to the nearest upper-level clusters to determine its general context, and then compared with all representations inside the cluster to find its k nearest neighbors and assign the final label by majority voting. We set a threshold for finding the kNN so that queries with the nearest distance and larger than the threshold will not be assigned any label, as we expect it to be a new context representation that is not included in the training database.

Remark 1. *All the training procedures presented in Section 4 are performed offline, making the online computations (basic algebraic operations) in smartphones very light-weight.*

5. Evaluation

In this section, we firstly provide details on our system implementation, data preparation, and model parameter settings. Then, we evaluate the effectiveness of context profiling by examining the performances of the two derived applications.

5.1. System Implementation and Data Preparations

We develop applications for Android in Java, using four smartphones for data collection: Huawei P10, P10 Plus, Mate 9, and Samsung S8. The entire data processing and model training pipeline is implemented in Python, and set up on a PC with a 2.6-GHz Intel Core i7 CPU and a 16 GB 1600-MHz DDR3 memory.

5.1.1. Data Collection

Our Android application collects raw GNSS measurements and locations returned by GPS. The GPS location indicator has two sources in Android: AndroidLoc, obtained through the Android LocationManager API [37], and GoogleLoc, provided by Google's FusedLocationProviderApi [38]. We record locations and their accuracies (associated with the Location class) for both sources at a frequency of 1 Hz. The ground truth is manually labeled by marking critical points along the way, and the locations between the critical points are linearly interpolated by time, assuming users move at a constant speed. We collect 30 traces at different times covering about 9000 typical location points in the urban area, roughly summarized in the following nine scenarios:

On street	On void deck
blocked on one side by buildings	blocked on two sides by buildings
building enclosing	inside a glass-covered building
on a sheltered street	in a tunnel
inside a room with heavy walls	

We randomly select 20 traces for training purposes and assign the remaining traces as testing traces, resulting in a training dataset with about 6600 location points and a testing dataset with about 2300 location points.

5.1.2. Data Pre-Processing

We match the corresponding raw GNSS measurements, GPS locations, accuracy (from two sources), and ground truth locations by their timestamps. We further calculate the *true GPS localization error*s as the longest radius between GPS locations and their respective ground truth locations. As explained in Section 4.1.2, we use SvPosition as an index to pack features into a snapshot. We set the snapshot (image) resolution as 18×18. The following 18 features are chosen; their details can be found in [33]:

ADR_STATE_CYCLE_SLIP	HardwareClockDiscontinuityCount
ADR_STATE_RESET	BiasUncertaintyNanos
ADR_STATE_VALID	ReceivedSvTimeUncertaintyNanos
STATE_SYMBOL_SYNC	Cn0DbHz
STATE_MSEC_AMBIGUOUS	PseudorangeRateMetersPerSecond
STATE_TOW_DECODED	PseudorangeRateUncertaintyMetersPerSecond
STATE_SUBFRAME_SYNC	AccumulatedDeltaRangeUncertaintyMeters
STATE_BIT_SYNC	DeltaRange
STATE_CODE_LOCK	MultipathIndicator

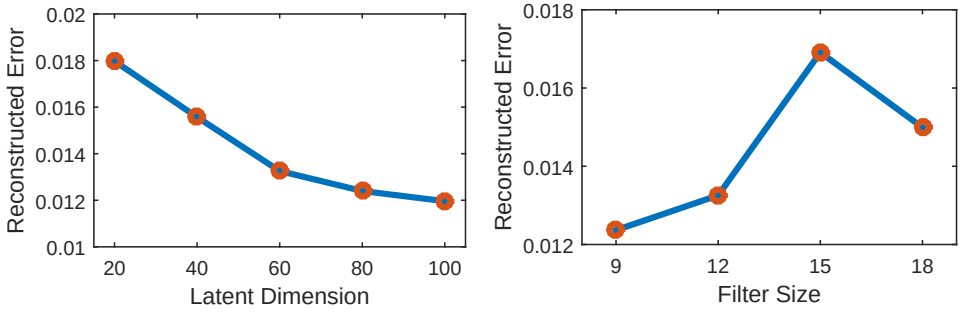
Remark 2. *We note that our current experiments are rather preliminary; the data traces collected are far from comprehensive and the GNSS features are selected largely on an intuitive basis. Nevertheless, we believe they are sufficient to demonstrate the effectiveness of mining GNSS measurements and to provoke further research on related topics. In particular, crowd sensing (e.g., used to gather WiFi data for outdoor localization [16]) can help to significantly enlarge the coverage in collecting GNSS measurements.*

5.2. Autoencoder Model Training

5.2.1. Parameter Setting

As presented in Section 4.2, there are three groups of hyperparameters to be determined: (i) input sequence steps for the sequence-to-sequence model; (ii) a latent dimension for the RNN GRU cells; and (iii) the filter size for the convolutional layer. The number of time steps is set it to five, thus, the model includes measurements from past 5 s. We set such a relatively small number as we intend to make the context relatively stable within the period so that the resulted representation can be better characterized as a stable context. As the loss for our model is the reconstruction error, we evaluate the effect of these parameters

through reconstruction error, as shown in Figure 5. The latent dimension directly affects the dimension of our representation; there is a trade-off in determining the kernel number. On one hand, a larger latent dimension allows representations to encode more information. On the other hand, a larger latent dimension brings additional storage costs, model training, and application, at later stage. To achieve a balance, we select 60 when the error begins to gradually drop. We set the convolutional filter size as (12,12) so that each patch can capture a sufficient image while achieving acceptable reconstruction performance. We apply an Adam Optimizer and a ReLU activation function during the training procedure.



(**a**) Latent dimension (**b**) Convolution filter size

Figure 5. Autoencoder parameters (in quantity) vs. reconstruction error (unitless).

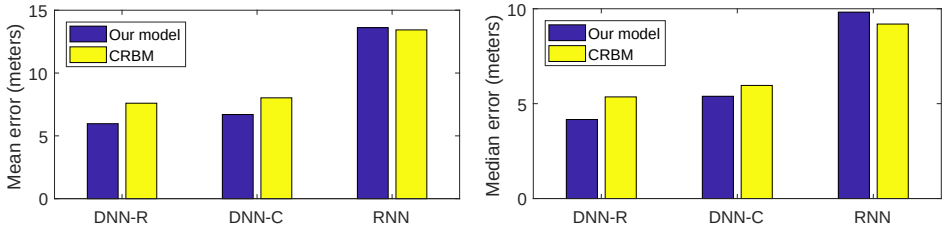
5.2.2. Comparison with Other Embedding Mechanisms

As a comparison, we also build a convolutional restricted Boltzmann machine (CRBM), as presented in [31]. To incorporate it, we treat satellites' identifiers as their dimensions (D) and various features as the channel (C). Since the feature packing is different, we omit a comparison of reconstruction error but compare the effectiveness of representation next, based on the applications discussed in Section 4.3.

5.3. Localization Error Estimator

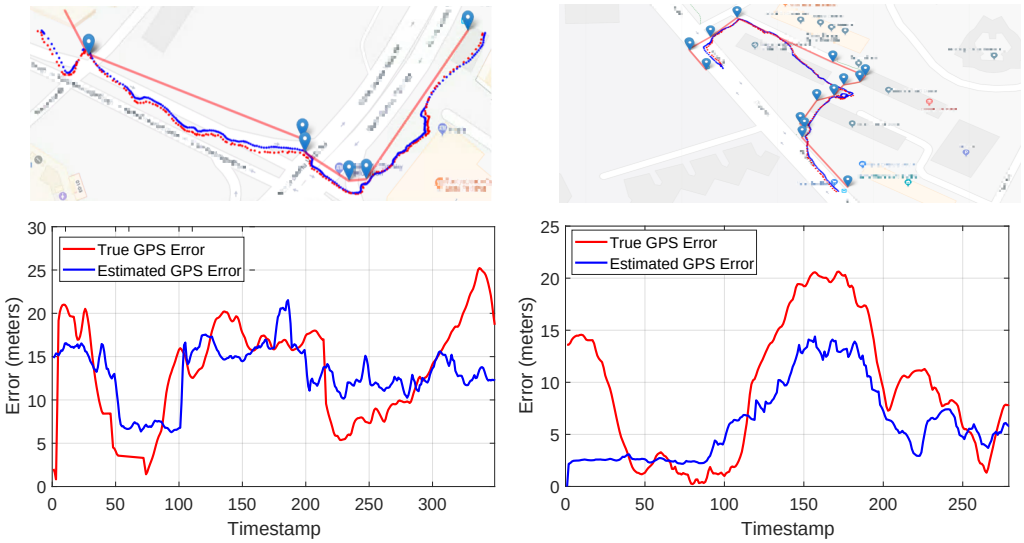
To evaluate the effectiveness of representation, we firstly examine its performance in estimating localization error. We build three types of location error estimators, as described in Section 4.3.1: DNN-R, DNN-C, RNN. They share the same loss function as the mean squared error between the regression result and true GPS error. DNN-R consists of four layers with 32, 16, 8, and 1 neuron(s), respectively, and takes ReLU as its activation function. DNN-C has a similar structure to DNN-R, with four layers, but the layer with eight neurons has an additional activation layer of soft-max, aiming to output the error range classification for eight classes. The errors are divided into eight bins, and each has a size (error range) of eight meters. The single neuron in the last layer takes the expectation for all error bins. RNN shares a similar structure as DNN-R except that the first three layers are recurrent neural layers with GRU cells.

We train the three models using representations generated from our autoencoder with the training traces, and test them against those generated with the testing traces. To compare with CRBM embedding, as discussed in Section 5.2.2, we conduct the same procedure using the CRBM embedding model. The mean and median errors for all testing data are shown in Figure 6. DNN-R performs the best out of the three models, achieving a median difference of 3.65 m and a mean difference with the true errors of 5.18 m. Our model generally performs better than CRBM, except for RNN, but both models perform the worst in this case, as they tend to over-fit the sequence pattern. To showcase the change of errors along a trace, two sample traces are also shown in Figure 7, along with the comparison between the true error sequence and the estimated one, based on our model using DNN-R. Both sequences generally capture the same error trends, though the magnitudes thereof may not perfectly match each other.



(**a**) Mean error (**b**) Median error

Figure 6. Deviations from true localization error.



(**a**) Error estimation for trace 1 (**b**) Error estimation for trace 2.

Figure 7. Trace samples (in second) with error estimation (in meter)

To further evaluate the performance of our representation in estimating GPS error, we compare it with the error estimates provided by the Android system. Instead of an absolute estimated error range, both the Android system [37] and Google [38] (yes, they are different APIs) provide an indicator called *horizontal accuracy*, defined as the radius of 68% confidence. In other words, there is a 68% probability that the true location is inside the circle centered at the reported location, with a radius indicating accuracy. To make our results comparable with it, we transform our absolute estimated error into a similar "accuracy indicator" by obtaining the 68% percentile over the error distribution within the preceding 10 s.

Based on the derived "accuracy indicator" for our model and the "horizontal accuracy" from the Android system, we compute the probability that the true GPS error is smaller than each accuracy range within a sliding window. A probability close to 68% implies an adequate accuracy indicator for localization performance, and one larger than 68% implies an over-conservative estimation, while one smaller than 68% implies an over-confident estimation. We also measure the probability with different sliding window sizes. Figure 8 reports the mean and median probabilities over all testing trips for different sliding window sizes, and it makes a comparison among accuracy indicators derived from our model, Android [37], and Google [38]. All models give a relatively stable median probability and mean probability over different window sizes, indicating that the estimators perform in a consistent manner. However, the indicator delivered by our model remains, almost always, around 68%, while the other two indicators always give over-confident estimations. This suggests that our model yields a more reliable location accuracy indicator, as compared with the other two systems.

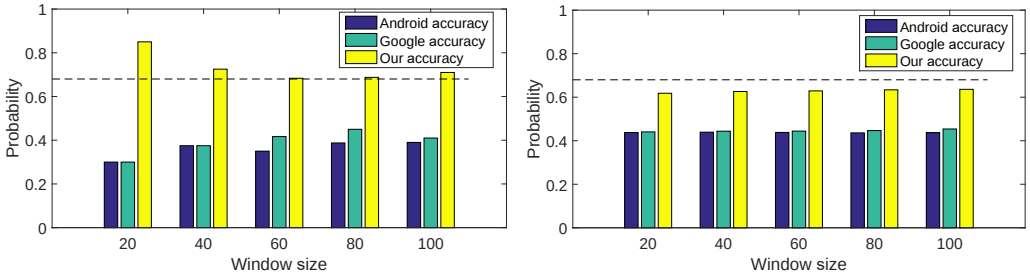
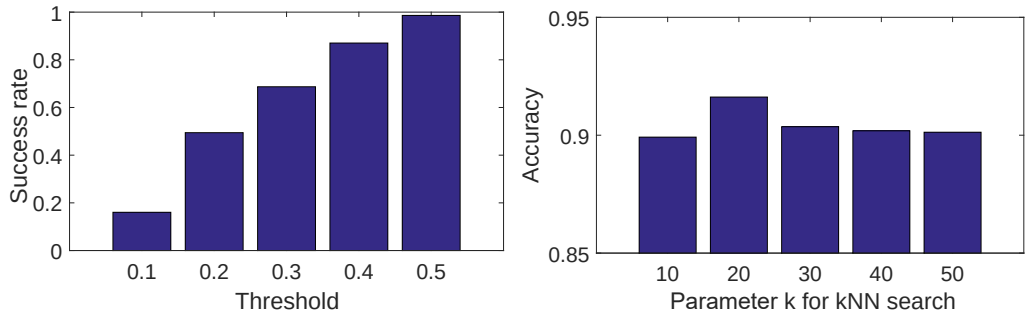


Figure 8. Performance of localization accuracy indicator. (**a**) Mean probability of error falling within the confidence range at different time resolutions; (**b**) median probability of error falling within the confidence range at different time resolutions.

5.4. Location Contexts Semantic Analysis

To evaluate the effectiveness of representation in characterizing different contexts, we firstly build up a context database of all training trips by adopting a bottom-up clustering process (i.e., lower-level clustering followed by upper-level clustering) with DBSCAN. For 20 training trips, we finally get 86 lower-level clusters, 7 upper-level clusters, and 14 small clusters remain as noise at the upper level, given their unique satellite patterns. To provide ground truth for the training set, we manually assign predefined labels, as described in Section 5.1.1, to all the representations in each cluster. Noise points detected through DBSCAN at lower-level clustering are labeled as "Noise".

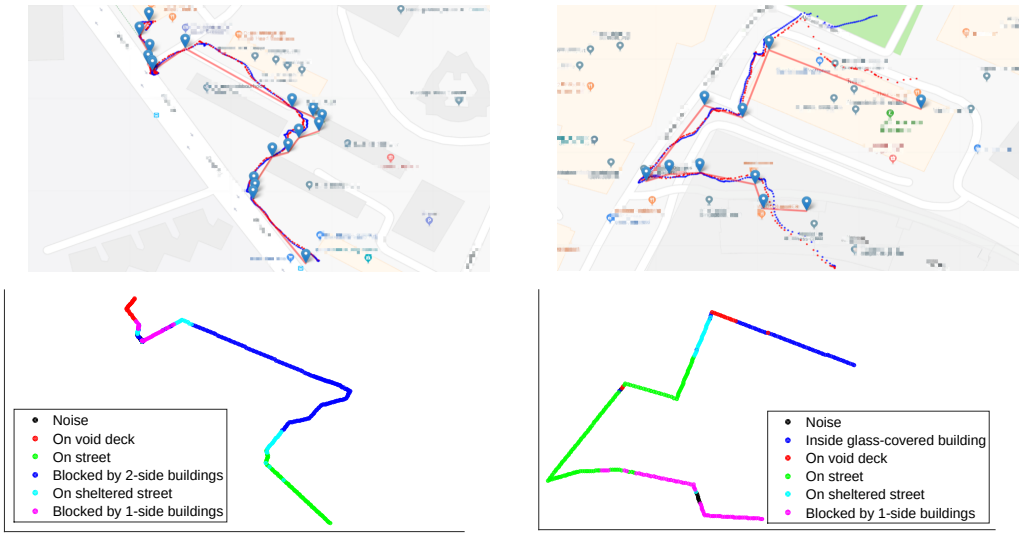
At the query phase, we conduct a top-down search within our context database. We assign each representation the same detailed context label of its k nearest neighbors, using cosine distance as a metric. If the nearest distance is larger than our predefined threshold, we record it as a "failure case". We manually examine the assigned labels based on their corresponding locations for validation. As shown in Figure 9, we firstly evaluate the success rate given different thresholds. The success rate of queries increases when the threshold increases, and a threshold of 0.5 can cover 98.6% of queries. We then evaluate classification accuracy under threshold of 0.5 for different k values, and achieve the highest classification accuracy of 91.62% when $k = 20$; the accuracies at different k values are all above 90%. The results show that our model can successfully recognize most contexts given a sufficient and well-labelled database.



(**a**) Success rate vs threshold. (**b**) Accuracy vs k for kNN search

Figure 9. Clustering performance with various threshold and k.

To better showcase the performance in the context semantic analysis, we demonstrate the result of two test traces along with their snapshots, taken from Google Maps in Figure 10. Most of the location points are correctly labeled, except some noise points and a mismatch between "on void deck" and "on sheltered street" in both traces, given that the two scenarios share quite similar contexts and, thus, it is hard to distinguish between them.



(**a**) Label result for sample trace 1 (**b**) Label result for sample trace 2

Figure 10. Trace samples with context label results.

5.5. Comparing with IO-Detectors

It is not evident how to accomplish directly comparing earlier proposals on IO-detection [10–12] in terms of detection accuracy, as we offer a much finer granularity in context profiling than just classification into three scenarios (indoor, outdoor, semi-outdoor), and our location context does not only include scenarios (see our definition in Section 3.1). Earlier GPS-free proposals [10,11] complain about the excessive delay incurred by a GPS-based approach, but the performance of commercial GPS modules has drastically improved since, and we only retrieve raw GNSS measurements. Consequently, we do not experience any excessive delay in detecting context changes during our experiments. As a result, the only meaningful comparative metric left is energy efficiency.

To this end, we compare the energy consumption between our system and GPS-free IO-detection proposals [10,11], which rely mainly on a magnetometer, cellular receiver, and light sensor. Therefore, we develop an Android application to collect magnetic field strength, cellular signal, and light intensity at a frequency of 100 Hz and install it on a Huawei P10, in order to emulate the systems used in [10,11]. By contrast, our application only collects raw GNSS measurements at a frequency of 1Hz. We measure the battery life under the following three conditions: (i) the phone stays with a user who either conducts data collection or acts normally in his/her daily life; (ii) each application keeps running until the battery level depletes to 30%; and (iii) each screen is set to the minimum brightness for checking results. As shown in Table 2, our system consumes about 20% more energy, largely comparable to the GPS-free solutions. Nonetheless, as our method achieves a much finer granularity in profiling location context compared with those earlier proposals; we believe that the slight disadvantage of our system, in terms of energy consumption, is acceptable. In fact, there is still a room to further improve the energy efficiency of GNSS sensing, given that this is only a preliminary attempt to make use of this recently available sensing ability.

Table 2. Battery consumption comparison between two different sensor sets.

Sensor Set	Battery Life (Hours)
magnetometer + cellular + light	29.2
raw GNSS measurement listener	24.2

6. Conclusions

In this paper, we have gone beyond well-studied indoor–outdoor detection and proposed holistically profiling the various location contexts in an urban area. Based on extensive studies on the raw GNSS measurements available since Android 7, we have innovated in a new method to organize this unconventional data structure so that effective mining techniques can be applied. We have then engineered an autoencoder module to extract compact representations from GNSS traces. In order to demonstrate the efficacy of this context profiling, we have showcased two applications: a localization error estimator, which is better than that provided by Android/Google, and a context semantic database, which is potentially extensible to assist other applications, including the seamless integration of indoor and outdoor localization schemes.

Our current work is still rather preliminary at this stage, mostly due to the fact that our GNSS data are only gathered from a few users. Therefore, this paper primarily seeks to raise the attention of our community: should we gather such data in a more pervasive manner (e.g., crowd sensing incentivized by cloud offloading [25]), we would be in a much better position to work towards full-fledged urban localization services.

Author Contributions: The first author, Jin Wang, worked on all the experiments and prepare the majority of the manuscript. The corresponding author, Jun Luo, came up with the idea and supervised the whole project, including revising the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The location and GNSS datasets used in this research cannot be publicly shared due the privacy concern of the volunteers.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kaplan, E.; Hegarty, C. *Understanding GPS Principles and Applications*, 2nd ed.; Artech House: Norwood, MA, USA, 2005.
2. Hassanieh, H.; Adib, F.; Katabi, D.; Indyk, P. Faster GPS via the Sparse Fourier Transform. In Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, Barcelona, Spain, 8–10 October 2012; pp. 353–364.
3. Bahl, P.; Padmanabhan, V. RADAR: An In-building RF-based User Location and Tracking System. In Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064), Tel Aviv, Israel, 26–30 March 2000; pp. 775–784.
4. Youssef, M.; Agrawala, A. The Horus WLAN Location Determination System. In Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, Seattle, WA, USA, 6–8 June 2005; pp. 205–218.
5. Xiong, J.; Jamieson, K. ArrayTrack: A Fine-grained Indoor Location System. In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, USA, 2–5 April 2013; pp. 71–84.
6. Kuo, Y.; Pannuto, P.; Hsiao, K.; Dutta, P. Luxapose: Indoor Positioning with Mobile Phones and Visible Light. In Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, Maui, HI, USA, 7–11 September 2014; pp. 447–458.
7. Zhang, C.; Zhang, X. Pulsar: Towards Ubiquitous Visible Light Localization. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, Snowbird, UT, USA, 16–20 October 2017; pp. 208–221.
8. Wang, H.; Sen, S.; Elgohary, A.; Farid, M.; Youssef, M.; Choudhury, R.R. No Need to War-drive: Unsupervised Indoor Localization. In Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, Ambleside, UK, 25–29 June 2012; pp. 197–210.
9. Zhang, C.; Subbu, K.; Luo, J.; Wu, J. GROPING: Geomagnetism and cROwdsensing Powered Indoor NaviGation. *IEEE Trans. Mob. Comput.* **2015**, *14*, 387–400. [CrossRef]
10. Zhou, P.; Zheng, Y.; Li, Z.; Li, M.; Shen, G. IODetector: A Generic Service for Indoor Outdoor Detection. In Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys 2012), Toronto, ON, Canada, 6–9 November 2012; pp. 113–126.
11. Radu, V.; Katsikouli, P.; Sarkar, R.; Marina, M. A Semi-Supervised Learning Approach for Robust Indoor-Outdoor Detection with Smartphones. In Proceedings of the 12th ACM Conference on Embedded Networked Sensor Systems (SenSys 2014), Memphis, TN, USA, 3–6 November 2014; pp. 280–294.
12. Chen, K.; Tan, G. SatProbe: Low-Energy and Fast Indoor/Outdoor Detection based on Raw GPS Processing. In Proceedings of the 36th IEEE International Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9.

13. Chintalapudi, K.; Padmanabha Iyer, A.; Padmanabhan, V.N. Indoor Localization Without the Pain. In Proceedings of the 16th Annual International Conference on Mobile Computing and Networking, Chicago, IL, USA, 20–24 September 2010; pp. 173–184.
14. Nirjon, S.; Liu, J.; DeJean, G.; Priyantha, B.; Jin, Y.; Hart, T. COIN-GPS: Indoor Localization from Direct GPS Receiving. In Proceedings of the 12th Conference on Mobile Systems, Applications, and Services, Bretton Woods, NH, USA, 16–19 June 2014; pp. 301–314.
15. Cheng, Y.C.; Chawathe, Y.; LaMarca, A.; Krumm, J. Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys 2005), Seattle, WA, USA, 6–8 June 2005; pp. 233–245.
16. Wang, J.; Tan, N.; Luo, J.; Pan, S.J. WOLoc: WiFi-only Outdoor Localization Using Crowdsensed Hotspot Labels. In Proceedings of the 36th IEEE International Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9.
17. GNSS-Android. Android Raw GNSS Measurements. 2018. Available online: https://developer.android.com/guide/topics/sensors/gnss (accessed on 18 July 2018).
18. Kjærgaard, M.B.; Langdal, J.; Godsk, T.; Toftkjær, T. EnTracked: Energy-efficient Robust Position Tracking for Mobile Devices. In Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, Kraków, Poland, 22–25 June 2009; pp. 221–234.
19. Paek, J.; Kim, J.; Govindan, R. Energy-efficient Rate-adaptive GPS-based Positioning for Smartphones. In Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, San Francisco, CA, USA, 15–18 June 2010; pp. 299–314.
20. Lin, K.; Kansal, A.; Lymberopoulos, D.; Zhao, F. Energy-accuracy Trade-off for Continuous Mobile Device Location. In Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, San Francisco, CA, USA, 14–18 June 2010; pp. 285–298.
21. LaMance, J.; DeSalas, J.; Järvinen, J. Assisted GPS: A Low-Infrastructure Approach. 2002. Available online: http://gpsworld.com/innovation-assisted-gps-a-low-infrastructure-approach/ (accessed on 18 July 2018).
22. Constandache, I.; Gaonkar, S.; Sayler, M.; Choudhury, R.; Cox, L. EnLoc: Energy-Efficient Localization for Mobile Phones. In Proceedings of the 28th Annual Conference of the IEEE Communications Society, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 2716–2720.
23. Misra, P.; Hu, W.; Jin, Y.; Liu, J.; de Paula, A.S.; Wirström, N.; Voigt, T. Energy efficient GPS acquisition with Sparse-GPS. In Proceedings of the 13th ACM/IEEE International Conference Information Processing in Sensor Networks, Berlin, Germany, 15–17 April 2014; pp. 155–166.
24. Hedgecock, W.; Maroti, M.; Ledeczi, A.; Volgyesi, P.; Banalagay, R. Accurate Real-time Relative Localization Using Single-frequency GPS. In Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, Memphis, TN, USA, 3–6 November 2014; pp. 206–220.
25. Liu, J.; Priyantha, B.; Hart, T.; Ramos, H.S.; Loureiro, A.A.F.; Wang, Q. Energy Efficient GPS Sensing with Cloud Offloading. In Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys 2012), Toronto, ON, Canada, 6–9 November 2012; pp. 85–98.
26. Thiagarajan, A.; Ravindranath, L.S.; Balakrishnan, H.; Madden, S.; Girod, L. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2011), Boston, MA, USA, 30 March–1 April 2011; pp. 267–280.
27. Wu, X.; Tian, X.; Wang, X. Large-scale Wireless Fingerprints Prediction for Cellular Network Positioning. In Proceedings of the 37th IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 1007–1015.
28. Huynh, T.; Fritz, M.; Schiele, B. Discovery of Activity Patterns Using Topic Models. In Proceedings of the 10th International Conference on Ubiquitous Computing COEX, Seoul, Korea, 21–24 September 2008; pp. 10–19.
29. Matsubara, Y.; Sakurai, Y.; Faloutsos, C. AutoPlait: Automatic Mining of Co-evolving Time Sequences. In Proceedings of the International Conference on Management of Data (SIGMOD 2014), Snowbird, UT, USA, 22–27 June 2014; pp. 193–204.
30. Parate, A.; Chiu, M.C.; Chadowitz, C.; Ganesan, D.; Kalogerakis, E. RisQ: Recognizing Smoking Gestures with Inertial Sensors on a Wristband. In Proceedings of the 12th Conference on Mobile Systems, Applications, and Services, Bretton Woods, NH, USA, 16–19 June 2014; pp. 149–161.
31. Liu, C.; Zhang, L.; Liu, Z.; Liu, K.; Li, X.; Liu, Y. Lasagna: Towards Deep Hierarchical Understanding and Searching over Mobile Sensing Data. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, New York City, NY, USA, 3–7 October 2016; pp. 334–347.
32. Groth, E. A Pattern-Matching Algorithm for Two-Dimensional Coordinate Lists. *Astron. J.* **1986**, *91*, 1244–1248. [CrossRef]
33. GNSS-Android. Android Developers Documentations. 2018. Available online: https://developer.android.com/reference/android/location/GnssMeasurement (accessed on 18 July 2018).
34. NASA. CDDIS: NASA's Archive Space Geodesy Data. 2018. Available online: https://cddis.nasa.gov/Data_and_Derived_Products/GNSS/broadcast_ephemeris_data.html (accessed on 18 July 2018).
35. Hinton, G.E.; Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507. [CrossRef]
36. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.

37. Google. Android Developers Documentations: LocationManager. 2018. Available online: https://developer.android.com/reference/android/location/LocationManager (accessed on 18 July 2018).
38. Google. Google APIs for Android: FusedLocationProviderApi. 2018. Available online: https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi (accessed on 18 July 2018).