UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Deep Learning for Trajectory-Based Transportation Mode Identification

by Christos Markos

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Assoc. Prof. Richard Yi Da Xu

University of Technology Sydney
Faculty of Engineering and Information Technology

August 2021

Certificate of Original Authorship

I, Christos Markos declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with the Southern University of Science and Technology.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: August 1st, 2021

ABSTRACT

Deep Learning for Trajectory-Based Transportation Mode Identification

by

Christos Markos

Understanding users' mobility patterns and associated transportation modes is essential for intelligent transportation management and infrastructure design. Through the ubiquity of Global Positioning System (GPS) sensors in modern smartphones and vehicles, rich spatiotemporal trajectories can be readily captured for use in intelligent transportation applications. Key among the latter is transportation mode identification, or how to infer travel modes within GPS trajectories. Although studied extensively, its real-world applicability remains limited due to several challenges.

GPS trajectories are often incomplete due to signal lapses, thereby complicating subsequent analysis. Since learning from raw GPS data restricts model generalization to the regions best covered in the training set, this thesis sets an alternative imputation target: approximate missing GPS points by learning to impute relative magnitude and angle of displacement features. The proposed Uncertainty-aware Imputation Generative Adversarial Network (UI-GAN) leverages a Bayesian generator to capture imputation uncertainty and a window-level discriminator for localized sequence structure penalization. UI-GAN produces high-fidelity GPS points and outperforms established imputation baselines.

A single GPS trajectory may encompass multiple transportation modes. Existing trajectory segmentation approaches often exhibit poor scalability and require extensive feature engineering or transportation domain knowledge. As such, this thesis reframes trajectory segmentation as timestep-level transportation mode identification. Concretely, it proposes a shuffling-based data augmentation

scheme and a *majority-vote* post-processing step to effectively train a convolutional neural network for timestep-level classification and refine the extracted segments. The proposed segmentation model is nearly twice as accurate as the best performing baseline in detecting transportation mode changes.

In reality, GPS trajectories are neither automatically annotated nor segmented by transportation mode. In addition, predictive uncertainty tied to model parameters or noise in GPS readings is typically unaccounted for. Therefore, this thesis proposes an unsupervised channel-calibrated Bayesian Temporal Convolutional Network (BTCN) trained to maximize the mutual information between neighboring feature map patches. By approximating variational inference, BTCN can both classify each input timestep and estimate its predictive uncertainty. BTCN significantly outperforms established trajectory segmentation baselines without using any labels.

Finally, this thesis proposes an unsupervised deep learning approach to transportation mode identification. First, a clustering layer maintaining cluster centroids as trainable weights is attached to the embedding layer of a convolutional autoencoder. The composite model is then trained by optimizing a weighted sum of reconstruction and clustering losses to encourage learning clustering-friendly representations. By further incorporating segment-level features, the proposed model outperforms traditional clustering and state-of-the-art semi-supervised methods without using any labels.

Dedication

I dedicate this thesis to my wonderful parents Costas and Sophia, whose constant support, love, and encouragement saw me through the difficult times of this journey. I also dedicate my thesis to my grandfather, Christos, whose unwavering pride in me boosted my self-confidence and gave me the motivation to someday be the man he saw me as. Finally, I dedicate this work to my partner Danae and her mother Lucy, who were always there to both listen to my frustrations and celebrate my accomplishments.

Acknowledgements

I am sincerely grateful to my supervisors, Professor Richard Yi Da Xu and Professor James Jian Qiao Yu (Southern University of Science and Technology), whose endless support in challenging times and meticulous approach to research were an endless source of inspiration. The extent to which they influenced me as a researcher cannot be overstated.

In addition, I thank Dr. Jason Traish for his encouragement and thought-provoking feedback during our brainstorming meetings. I am also grateful to Wei Huang, Zayne Zhang, and Dr. Caoyuan Li for our weekend explorations of the Sydney sights. These trips have left me with countless wonderful memories and at the time provided me with much-needed energy for resuming work the next day. Last but not least, I thank Chenhan Zhang, Xiaozhuang Song, Yuanshao Zhu, and Dr. Shiyao Zhang for our excellent collaboration in producing high-quality publications.

List of Publications

Journal Papers

J-1. C. Zhang, Y. Zhu, **C. Markos**, S. Yu, and J. J. Q. Yu, "Towards Crowdsourced Transportation Mode Identification: A Semi-supervised Federated Learning Approach," *IEEE Internet of Things Journal*. Under review.

J-2. S. Zhang, **C. Markos**, and J. J. Q. Yu, "Autonomous Vehicle Intelligent System: Joint Ride-Sharing and Parcel Delivery Strategy," *IEEE Transactions on Intelligent Transportation Systems*. Under review.

J-3. J. J. Q. Yu, **C. Markos**, and S. Zhang, "Long-Term Urban Traffic Speed Prediction with Deep Learning on Graphs," *IEEE Transactions on Intelligent Transportation Systems*, in press.

Conference Papers

C-1. Y. Zhu, **C. Markos**, and J. J. Q. Yu, "Improving Transportation Mode Identification with Limited GPS Trajectories," in *IEEE 24th International Conference on Intelligent Transportation Systems (ITSC)*, 2021. Under review.

C-2. Y. Zhu, **C. Markos**, R. Zhao, Y. Zheng, and J. J. Q. Yu, "FedOVA: One-vs-All Training Method for Federated Learning with Non-IID Data," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

C-3. **C. Markos**, J. J. Q. Yu, and R. Y. D. Xu, "Capturing Uncertainty in Unsupervised GPS Trajectory Segmentation Using Bayesian Deep Learning,"

in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 1, 2021, pp. 390–398.

C-4. **C. Markos** and J. J. Q. Yu, "Unsupervised Deep Learning for GPS-Based Transportation Mode Identification," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

C-5. X. Song, **C. Markos**, and J. J. Q. Yu, "MultiMix: A Multi-Task Deep Learning Approach for Travel Mode Identification with Few GPS Data," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms / Abbreviations

ACC	Accuracy
AUC	Area Under Curve
CAE	Convolutional AutoEncoder
CNN	Convolutional Neural Network
FCN	Fully Convolutional Network
GAN	Generative Adversarial Network
GIS	Geographic Information System
GMM	Gaussian Mixture Model
GPS	Global Positioning System
HAC	Hierarchical Agglomerative Clustering
ITS	Intelligent Transportation System
KL	Kullback-Leibler
KM	k-Means
k-NN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MC	Monte Carlo
MF	Matrix Factorization
MICE	Multiple Imputation using Chained Equations
mIoU	mean Intersection over Union
MLP	MultiLayer Perceptron

NMI	Normalized Mutual Information
PR	Prediction Ratio
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
SAE	Stacked AutoEncoder
SC	Spectral Clustering
SVM	Support Vector Machine
TCN	Temporal Convolutional Network
VAE	Variational AutoEncoder

Notation

Lowercase non-bold characters denote scalar quantities (e.g., x, λ); uppercase non-bold characters denote constant scalars (e.g., M, N, K); lowercase bold characters denote vectors (e.g., $\mathbf{x}$); uppercase bold characters denote matrices (e.g., $\mathbf{X}$); uppercase bold Euler characters denote tensors (e.g., $\boldsymbol{\mathcal{X}}$). We let $\mathbf{x}_i$ denote the i-th element of vector $\mathbf{x}$. Instead, $\mathbf{X}_{i,:}$ denotes the i-th row of matrix $\mathbf{X}$, while $\mathbf{X}_{i,j}$ denotes the element at row i, column j. We use $\mathbf{X} \odot \mathbf{Y}$ to denote the Hadamard or element-wise product between matrices $\mathbf{X}$ and $\mathbf{Y}$. We denote the transpose operation on matrix $\mathbf{X}$ as $\mathbf{X}^\mathsf{T}$. Finally, we use $\mathbb{R}$ and $\mathbb{Z}^+$ to denote the sets of real numbers and positive integers, respectively.

Chapter 1

Introduction

1.1 Motivation

Transportation mode identification is the problem of associating segments of users' mobility traces to transportation modes. Location-based services can leverage such knowledge to generate highly accurate, personalized suggestions that are tied not only to an individual's real-time location but also their current transportation mode [5]. Examples include electronic billboard advertisements or departure notifications and route recommendations so that users users may reach their destination in a timely manner [5]. For Intelligent Transportation Systems (ITSs) aiming to improve traffic management, transportation mode identification is critical for operations such as public transportation scheduling and travel demand analysis [6].

The effectiveness of transportation mode identification methods depends on the collection and analysis of massive mobility data from fixed-point or mobile sensors. The presence of Global Positioning System (GPS) sensors in most modern smartphones and wearable devices, capturing temporally ordered sequences of geographic coordinates while staying in constant user proximity, provides con-

siderably wider path coverage compared to standard fixed-point sensors. Nevertheless, the real-world applicability of transportation mode identification based on GPS data remains limited due to a number of open problems.

A significant challenge is that GPS trajectories may be incomplete, as signal interruption often occurs in urban areas due to tall buildings, bridges, tunnels, or extreme weather,* resulting in missing GPS data. Training models on incomplete trajectories introduces ambiguity into the learning process that may significantly impede not only transportation mode identification, but also other downstream applications such as destination prediction and travel time estimation. Yet most such works have neither investigated the effects of missing GPS data on their models' reliability and predictive performance nor attempted to impute them. In this direction, *GPS trajectory imputation* aims to estimate missing GPS points within trajectories. The seminal work of Zheng *et al.* [7] imputes sparsely-sampled trajectories by combining historical data and map-matching. Banerjee *et al.* [8] train a higher-order Markov model on trajectories represented as edge-weighted graphs, with each weight holding the probability of traveling across its corresponding road segment. To alleviate the need for underlying map information, Li *et al.* [9] instead build a junction network from aggregated trajectories and fill in GPS points based on its topology. However, all three approaches are by definition only applicable to geographic areas sufficiently covered in the training set. More recently, Wang *et al.* [10] augment GPS data with curvature information obtained via high-order polynomial fitting and train a Recurrent Neural Network (RNN) to reconstruct simulated missing points with a focus on preserving the trajectories' overall geometry. Nonetheless, this method requires complete trajectories at training time.

While it is possible to impute trajectories using traditional machine learn-

*https://www.gps.gov/systems/gps/performance/accuracy/

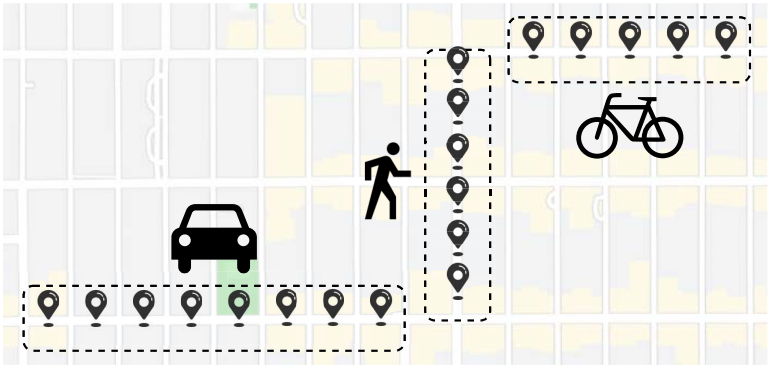

Figure 1.1 : GPS trajectory segmentation aims to extract single-transportation-mode segments from sequences of GPS points.

ing methods such as k-Nearest Neighbors (KNN) [11], Matrix Factorization (MF) [12]. Multiple Imputation using Chained Equations (MICE) [13], or missForest [14], these methods were designed with tabular rather than sequential data in mind. Recent work in general time series imputation has reported better results leveraging RNNs to model long-term temporal dependencies [15, 16]; however, both GRU-D [15] and BRITS [16] are trained jointly with a supervised downstream task, limiting their generality and potentially biasing the learned imputation. Another line of work has successfully adapted Generative Adversarial Networks (GANs) [17] for both general [18, 19] and time-series-specific [20, 21, 22] imputation. For instance, NAOMI [22] uses a non-autoregressive adversarial framework to recursively estimate missing time series values from coarse to fine-grained resolutions. Nonetheless, it cannot handle originally incomplete data such as GPS trajectories.

Another challenge is that users' trajectories are not automatically segmented by transportation mode, as required for transportation mode identification. The challenging problem of *GPS trajectory segmentation* (Figure 1.1), or how to split GPS trajectories into segments involving exactly one transportation

mode, is usually avoided by leveraging the transportation mode labels available during data preprocessing. Following the naive approach of simply extracting consecutive nonoverlapping trajectory segments of uniform length [23], duration, or distance, there may be multiple transportation modes in many of the obtained segments. Even by mapping these segments to their dominant transportation mode, one would be injecting the data with additional noise, potentially impacting the performance of transportation mode classifiers. In this direction, the literature has mostly sought to detect transportation mode change points within GPS trajectories by leveraging mobility-based heuristics [1, 24, 25, 26, 27], and more recently, optimization-based algorithms [3]. However, such approaches often exhibit poor scalability, require extensive feature engineering or transportation domain knowledge, and assume independent and identically distributed samples, respectively.

In response to this issue, we note the merit of deep semantic segmentation approaches commonly used in computer vision applications. Semantic image segmentation classifies images at pixel-level, allowing for fine-grained detection of object boundaries. Earlier work [28, 29] repurposes image-level-pretrained CNN classifiers for semantic segmentation by replacing their fully connected layers with convolutions, introducing in-network upsampling, adding skip connections from shallow to deep layers [29], and optimizing a pixel-level loss. Several studies have since leveraged these Fully Convolutional Networks (FCNs) as a backbone for developing context aggregation modules [30, 31] to recover the downsampled spatial information, often including dilated convolutions in cascades [30] or in parallel [31]. Other architectural adaptations include adding a simple decoder [32] or a symmetric succession of convolutions and upsampling layers with encoder-decoder skip connections [33]. More recently, deep semantic segmentation has been successfully applied to time series data, with U-Time [2] modifying the

architecture in reference [33] for sleep staging segmentation.

Because raw geographic coordinates are ill-suited for direct processing by machine learning models, GPS trajectories are typically preprocessed into sequences of relative distance, velocity, acceleration, and other motion features. Earlier work in *transportation mode identification*, i.e., classification of single-transportation-mode segments, has leveraged decision trees [1, 34] or random forests [35], support vector machines [36], and Bayesian networks [26], to name a few. Not only are such models suboptimal for learning from large amounts of data, but also motion features like velocity and accelerations are often noisy due to missing data, positioning errors, and traffic conditions [1]. With deep learning models recently attaining state-of-the-art results in challenging applications of computer vision [37, 38, 39] and natural language processing [40, 41], they have similarly attracted growing interest in transportation research. Several researchers have designed transportation mode identification frameworks based on multilayer perceptrons [42] and recurrent [43, 44, 45] or convolutional neural networks [3, 46]. Nonetheless, the problem of how to perform either transportation mode segmentation or identification from entirely *unlabeled* GPS data has not yet been addressed. This is crucial for real-world applications, given that user movement captured by GPS sensors does not contain any information on the corresponding transportation modes. In addition, users often hesitate to label their trajectories by transportation mode due to lack of motivation or privacy concerns, with the former being exacerbated by the difficulty of accurate and consistent annotation of such voluminous data.

In this direction, recent work in unsupervised learning has proposed several *deep clustering* methods. Some approaches optimize image-to-image distance in a lower-dimensional space typically learned via unsupervised autoencoder pretraining, by applying k-Means clustering [47] or attaching custom clustering lay-

ers [48, 49] and minimizing an associated loss. DeepCluster by Caron *et al.* [4] instead leverages a CNN pretrained on different datasets by iteratively applying k-Means clustering to the network's outputs, setting the cluster memberships as pseudo-ground-truth labels, and using them to update the network parameters. It is also possible to leverage denoising [50] or variational [51] autoencoders, as well as GANs [52]. Another body of research instead clusters data based on mutual information between pairs of samples typically generated via data augmentation, or between samples and their latent representations [53, 54, 55]. For instance, Hu *et al.* [53] train neural networks to produce similar outputs for input samples and their augmented versions, while also maximizing the mutual information between the original samples and their learned representations. Even though these *deep clustering* frameworks designed for image and text data have attained encouraging results, to the best of our knowledge, there is no evidence regarding their suitability for GPS trajectory data.

1.2 Research Questions

This section develops the Research Questions (RQs) that both motivated and guided the undertaking of this thesis. Initially resulting from our literature review presented in Chapter 2, our RQs were subsequently revised according to novel challenges discovered in the process of addressing them. To demonstrate the value of pursuing each RQ, we briefly explain the research gap it aims to fill within the existing literature.

RQ1: How to impute missing GPS points within densely-sampled trajectories using deep learning? Signal lapses often render users' GPS trajectories incomplete, thereby complicating subsequent analysis. This directly affects transportation mode identification, which is typically performed on sequences of motion features such as relative distance and bearing, extracted from consecutive

pairs of GPS points. When GPS points are missing, such features are inevitably computed with arbitrary estimation errors, thereby inserting noise to the data. Existing approaches rely on the availability of historical data and underlying map information [7, 8, 9], or require complete trajectories at training time [10]. RQ1 is addressed in Chapter 3.

RQ2: How to divide GPS trajectories that are pointwise labeled by transportation mode into single-transportation-mode segments using deep learning? Transportation mode identification is typically applied to GPS trajectory segments involving exactly one transportation mode. To split GPS trajectories into such segments, the literature has mainly used mobility-based heuristics [1, 24, 25, 26, 27] and optimization-based change point detection algorithms [3]. Nonetheless, such approaches demand extensive feature engineering or transportation domain knowledge and make the strong assumption of independent and identically distributed samples, respectively. RQ2 is addressed in Chapter 4.

RQ3: How to split unlabeled GPS trajectories into single-transportation-mode segments using deep learning? Users' trajectories are generally not labeled by transportation mode due to the considerable effort required to accurately and consistently annotate such data, in addition to privacy concerns. Consequently, real-world applicability demands that trajectory segmentation models such as the one resulting from RQ2 be able to operate on trajectories not labeled by transportation mode. RQ3 is addressed in Chapter 5.

RQ4: How can unlabeled, single-transportation-mode GPS trajectory segments be clustered by mode of transportation using deep learning? To date, deep-learning-based approaches to GPS-based transportation mode identification [56, 23, 46, 44, 57] have largely relied on the availability of labeled data for supervised learning. Recent work [58, 3, 45] has partially addressed this problem by

combining large amounts of unlabeled GPS data with relatively few labeled data in semi-supervised learning. However, these works have demonstrated varying degrees of success depending on the percentage of labeled data used during training, with results suggesting room for improvement especially when using very few labeled data [3]. In the absence of labels, researchers typically turn to clustering methods; nonetheless, clustering trajectory segments by transportation mode is non-trivial since traditional clustering methods tend to underperform when applied to high-dimensional data [49]. RQ4 is addressed in Chapter 6.

1.3 Contributions

Motivated by the research gaps identified in Section 1.1 and formalized as research questions in Section 1.2, this thesis explores novel deep learning approaches for GPS-based trajectory imputation, segmentation, and identification of transportation modes. Concretely, this thesis makes the following contributions:

1. A framework for imputing incomplete GPS trajectories based on unsupervised deep learning. Rather than operate on raw GPS coordinates and thus limit model generalization, we propose to reconstruct incomplete time series of relative magnitude and angle of displacement, from which GPS points can then be recovered. Specifically, we employ a GAN formulation where a Bayesian generator reconstructs motion feature sequences in an attempt to fool a discriminator that penalizes sequences at window-level. The proposed framework is shown to outperform existing work and generate highly precise GPS points.

2. A supervised deep learning method for dividing labeled GPS trajectories into same-transportation-mode segments. Instead of relying on heuristics and transportation domain knowledge, we propose to view trajectory seg-

mentation as timestep-level transportation mode identification performed by a convolutional neural network. We further introduce a majority-vote post-processing step and a shuffling-based data augmentation scheme to reduce timestep-level errors and boost performance. Our results indicate that the proposed framework is approximately twice as accurate as the best performing baseline.

3. An unsupervised deep learning approach to accomplishing the above task without using transportation mode labels. Again viewing trajectory segmentation as timestep-level transportation mode identification, this time a convolutional neural network learns to maximize the mutual information between nearby patches of feature maps. In addition, the network is able to approximate variational inference, thereby providing predictive uncertainty estimates for each timestep in a given trajectory. The proposed framework is shown to significantly outperform existing trajectory segmentation baselines.

4. A framework based on unsupervised deep learning for clustering medium-length GPS trajectory segments according to the five most prominent transportation modes in the literature, i.e, walk, bike, bus, car, and train, without using any labels. After pretraining a convolutional autoencoder on timestep- and segment-level motion features, a clustering layer whose trainable weights correspond to cluster centroids is attached to the autoencoder's embedding layer. The entire model is then retrained by balancing the autoencoder's reconstruction loss against the clustering loss associated with the clustering layer. Our experiments indicate that the proposed model outperforms not only semi-supervised transportation mode identification methods but also traditional clustering algorithms.

1.4 Thesis Outline

This thesis is structured as follows:

- *Chapter 2* surveys the literature on GPS trajectory imputation, transportation mode segmentation, and transportation mode identification. It also examines recent developments in semantic image segmentation and deep clustering, which have inspired the frameworks proposed in Chapters 4, 5, and 6.
- *Chapter 3* presents our proposed approach to unsupervised GPS trajectory imputation, in line with RQ1.
- *Chapter 4* introduces our proposed framework for supervised GPS-based transportation mode segmentation, following RQ2.
- *Chapter 5* details our proposed framework for unsupervised GPS-based transportation mode segmentation, in keeping with RQ3.
- *Chapter 6* presents our proposed approach to unsupervised GPS-based transportation mode identification, according to RQ4.
- *Chapter 7* summarizes our findings and recommends future research directions based on the identified research challenges.

As its structure may indicate, this thesis studies the problem of GPS-based transportation mode identification from a holistic perspective. Starting from incomplete trajectory imputation (Chapter 3), the focus then shifts on how to partition trajectories into single-transportation-mode segments with (Chapter 4) or without (Chapter 5) the use of transportation mode labels. It then examines how to leverage these trajectory segments to train a classifier for unsupervised transportation mode identification (Chapter 6).

Chapter 2

Literature Review

This thesis addresses research gaps tied to three distinct yet highly interdependent tasks within the GPS-based transportation literature. Starting from *trajectory imputation*, or how to estimate missing GPS points within users' trajectories, *transportation mode segmentation* then extracts single-travel-mode segments from trajectories of arbitrary length. Finally, *transportation mode identification* infers the travel mode used within each such segment. The following three sections survey the literature on these tasks in the same order. Note that, in this thesis, the terms *transportation mode segmentation* and *trajectory segmentation* are used interchangeably. This chapter also reviews recent work in semantic image segmentation and deep clustering, as they have driven the design of the transportation mode segmentation and identification frameworks introduced in Chapters 4, 5, and 6.

2.1 Trajectory Imputation

With GPS signal lapses frequently occurring near tall buildings, tunnels, or due to extreme weather conditions, GPS trajectories collected in urban areas are likely to be incomplete. Estimation of missing GPS points within trajectories is critical

for further mining and subsequent downstream tasks. The literature on GPS trajectory imputation can be broadly classified into knowledge-based [7, 8, 9] and deep-learning-based [10, 59].

2.1.1 Knowledge-Based Approaches

Earlier studies predominantly developed knowledge-based approaches to GPS trajectory imputation [7, 8, 9], primarily targeting sparsely-sampled* trajectories. After reducing positioning errors via map-matching and indexing all available trajectories, Zheng *et al.* [7] first decompose each target trajectory into pairs of consecutive GPS points. For each pair, the index is then queried for trajectories that intersect either of the two points within a desired radius. This action returns a set of *reference trajectories* on which inference is performed via either a traverse-graph-based or a nearest-neighbor-based algorithm to produce a set of local routes. Finally, the resulting local routes are ranked using a scoring function and combined into a global route by selecting those with the highest scores. Aiming to model the uncertainty that is inherent in sparsely-sampled trajectories, Banerjee *et al.* [8] instead represent them as edge-weighted graphs, with weights denoting the probability of traversing their associated road segments. To this end, the authors train a higher-order Markov model on graph representations of map-matched historical trajectories on the underlying road network. At inference time, the target trajectory is processed in pairs of consecutive GPS points, whereby a random walk with restarts is used to simulate routes from the first point to the second. The proposed approach was shown to outperform the one in reference [7] by approximately 50% in terms of accuracy.

A significant limitation to both frameworks described above is their reliance

*In the transportation literature, trajectories are generally considered to be sparsely-sampled when the sampling interval is 15 seconds or longer (even up to 1–2 minutes) [60, 8, 43].

on road network information. This can be problematic when handling trajectories that often deviate from roads, such as trajectories that involve walking, biking, or taking the train [9]. Aiming to alleviate the need for map matching, Li *et al.* [9] build a junction network topology from historical trajectory data and uses it for trajectory completion. Specifically, the proposed method first extracts a skeleton from the GPS point cloud forming each trajectory, resulting in a junction network graph where junction endpoints constitute vertices and traffic flows are represented by edges. Next, GPS points are projected to their nearest skeleton branch and traffic flow clusters are identified via a pairwise voting process applied to branches. Finally, trajectories are imputed using GPS points from other trajectories that belong to the same cluster.

2.1.2 Deep Learning Frameworks

More recent research has turned to deep neural networks with a focus on densely-sampled trajectory imputation [10, 59]. Wang *et al.* [10] note that GPS trajectory reconstruction often results in overly smooth segments due to not incorporating information on inflection points, i.e., points where the implicit trajectory curve has its curvature sign changed. As such, the authors first fit the latitude and longitude sequences comprising each trajectory using two high-order polynomials. For each transformed latitude-longitude pair, they then calculate the curvature radius. Next, all three features are embedded using an RNN autoencoder equipped with an attention mechanism, trained by maximizing the coefficient of determination between true and reconstructed trajectories. Finally, the latter are smoothened via a post-processing step combining a moving average algorithm and Savitzky-Golay filtering. Although the proposed imputation method effectively preserves the overall *geometry* of trajectories, it requires *complete* trajectories at training time. This limits its applicability to real-world applications, as GPS

trajectories are typically incomplete due to signal lapses. Moreover, the sensitivity of the proposed approach to hyperparameter variations was not investigated. Rather than operate on raw coordinates, Nawaz *et al.* [59] first extract point-wise motion features and then map GPS points to a two-dimensional grid. Both motion features and grid locations are then fed to a bidirectional Convolutional Long Short-Term Memory (ConvLSTM) encoder-decoder architecture equipped with an attention mechanism. Additional global and auxiliary features are also merged with the decoder's output before reaching the final, fully connected layer. While the proposed method demonstrated promising results, inference is tied to the granularity of grid cells, a practice that is not without limitations. Intuitively, if cells are too large, some may cover multiple roads; on the other hand, cells that are too small may introduce redundancy by including areas that in reality are unreachable.

Finally, it is important to emphasize how GPS trajectory imputation differs from related imputation tasks within the transportation domain. Traffic flow [61, 62] and speed [63] imputation both recover spatiotemporal measurements that are *aggregated* over multiple users; these measurements are typically captured by stationary rather than mobile sensors, such as inductive loop detectors or traffic microwave sensors. In contrast, GPS trajectory imputation estimates missing GPS points within *individual* users' trajectories, as recorded by GPS sensors in their smartphones or vehicles.

2.2 Transportation Mode Segmentation

Users' GPS trajectories may contain numerous trips, with each trip likely encompassing multiple transportation modes. For instance, reaching one's place of employment may involve first walking to the train station, taking the train, and finally boarding the bus. As such, trajectories must first be divided into trips,

from which single-transportation-mode segments can then be extracted (transportation mode *segmentation*) and subsequently classified (transportation mode *identification*) [1, 56, 3]. In this direction, Zheng *et al.* [1] detect a new trip whenever the time interval between two consecutive GPS points exceeds 20 minutes. This simple heuristic has since been predominantly used throughout the literature [23, 3, 64, 45]. Others have proposed clustering-based approaches to identifying trips: Gong *et al.* [65] extract a new trip upon formation of any cluster of points that remain within 50 meters of each other for longer than 200 seconds. Similarly, Xiao *et al.* [26] rely on point-clustering thresholds and abrupt changes of direction. Finally, Zhu *et al.* [66] use affinity propagation clustering with distance and time constraints, detecting a new trip whenever a *stay point* is found.

2.2.1 Heuristics-Based Approaches

Regarding the more challenging problem of transportation mode segmentation, a naive approach is uniform-length segmentation, whereby trajectories are partitioned into non-overlapping segments of a fixed number of GPS points (or timesteps). For instance, Dabiri and Heaslip [23] use a window equal to the median length of all labeled trajectories in the Geolife dataset. Much like any one-size-fits-all method, uniform-length segmentation offers simplicity at the expense of often producing segments involving multiple transportation modes; this is also true for approaches such as uniform-duration or uniform-distance segmentation. Such segments may introduce ambiguity when training transportation mode identification models, thereby impacting classification performance.

2.2.2 Change Point Detection Methods

Most GPS-based transportation mode segmentation methods have focused on detecting transportation mode *change points*, i.e., the discrete timesteps where

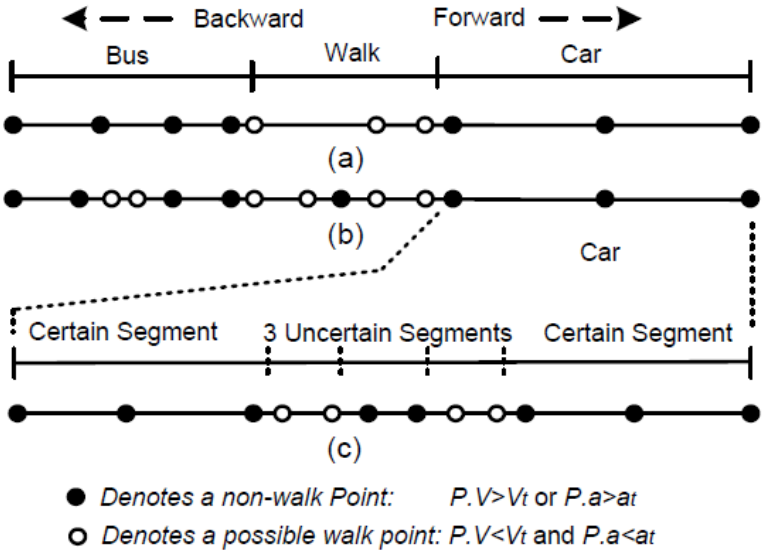

Figure 2.1 : Illustration of the transportation mode segmentation algorithm based on change point detection proposed by Zheng *et al.* [1]. Trajectories are split into *walk* and *non-walk* segments based on velocity and acceleration thresholds.

the transportation mode changes. This has been predominantly achieved via mobility-related heuristics [1, 34, 24, 25, 26, 27]. Following the intuition that walking must precede any change of transportation mode, the seminal work by Zheng *et al.* [1, 34] divides trajectories into alternating *walking* and *non-walking* segments based on distance, velocity, and acceleration thresholds. Following a similar approach, Schüssler *et al.* [24], Biljecki *et al.* [25], and Xiao *et al.* [26] detect change points at the boundaries of not only *walking* but also *gap* segments caused by signal lapses; Biljecki *et al.* [25] further identify stops exceeding a certain threshold as potential change points. Finally, Zhu *et al.* [27] extend [1] by applying a timestep-level *label revision* step to mitigate the effect of erroneous GPS points in the identified *walking* and *non-walking* segments.

More recently, Dabiri *et al.* [3] proposed an optimization-based approach to transportation mode change point detection. After applying uniform-length

segmentation to each trajectory, the authors convert the obtained segments into multivariate time series of velocity and acceleration features and feed them to an optimization-based model. The latter attempts to extract subsegments such that the homogeneity within each subsegment is maximized, while the number of identified change points is penalized by a hyperparameter-controlled linear function. Nonetheless, this method makes the strong assumption that the random variables, i.e., velocity and acceleration, are independent and identically distributed.

2.3 Semantic Image Segmentation

In the area of computer vision, semantic image segmentation is the task of pixel-level classification of images for fine-grained distinction of different objects in the image. Recent developments in semantic image segmentation have inspired the design of our transportation mode segmentation frameworks, which will be introduced in Chapters 4 and 5.

2.3.1 From Image-Level Classifiers to Fully Convolutional Networks

Long *et al.* [29] repurpose three pretrained image-level CNN classifiers for semantic segmentation by replacing their fully connected layers with convolution ones, introducing in-network upsampling, and optimizing a pixel-level multinomial logistic loss. They also add skip connections from shallow to deep layers, enabling local predictions to benefit from global information. The proposed modifications allowed the resulting Fully Convolutional Networks (FCNs) to produce dense, pixel-level predictions for arbitrary-sized inputs, resulting in an mIoU of 62.7% on PASCAL VOC 2012. Yu *et al.* [30] argue that image-level CNN classifiers subsequently adapted to pixel-wise predictions may contain redundant, potentially sub-optimal components. They demonstrate this in two ways: first, they craft a front-end module by removing the pooling layers of a pretrained VGG-16

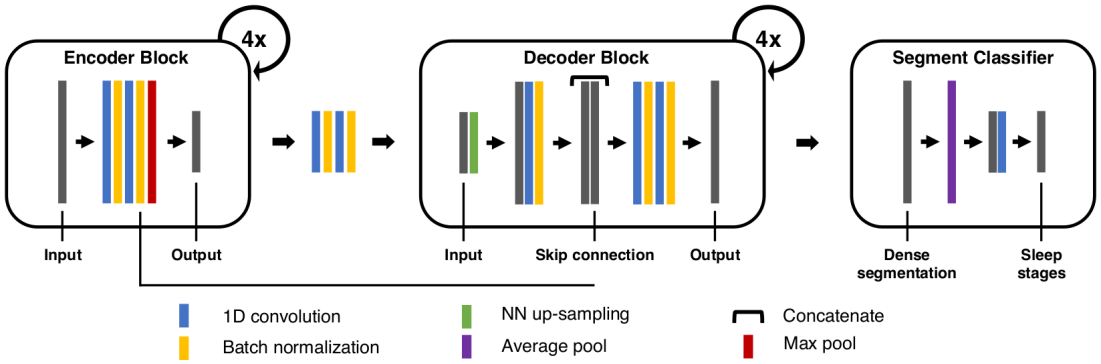

Figure 2.2 : The architecture of U-Time [2], a recent U-net variant proposed for sleep staging.

network and replacing the standard convolutions with dilated ones. Dilated convolutions help maintain high coverage and resolution as network depth increases by exponentially expanding the network's receptive field. The front-end module alone significantly outperformed [29] on the PASCAL VOC 2012 dataset. The same authors also propose a simplified context aggregation module consisting of successive dilated convolutions, attached to the front-end module. Combined with the front-end module, it achieved the highest mIoU among all evaluated methods. Ronneberger *et al.* [33] extend the contracting FCN architecture [29] with a symmetric succession of convolution and upsampling layers. The proposed U-net architecture includes a skip connection from each layer in the contracting path to its counterpart in the expansive sub-network. U-net is trained by optimizing the pixel-wise cross entropy loss. In their experiments, the authors showed that U-net significantly outperformed the evaluated baselines on a cell tracking dataset in terms of mIoU. They also found that applying data augmentation, in this case through elastic deformations of the training samples, significantly improved U-net's capacity to learn from few data.

The U-net architecture has since inspired numerous semantic segmentation

works. Jegou *et al.* [67] use a similar architecture to U-net, instead replacing convolution layers with DenseNet blocks due to their parameter efficiency, implicit deep supervision, and feature reuse properties. To avoid parameter explosion in the upsampling path, the input of dense blocks is not concatenated with their respective output; information at a given spatial resolution is preserved via skip connections from the contracting path. The proposed modifications to the upsampling path resulted in this framework outperforming U-net in terms of mIoU and global accuracy. It also fared better than [29], who leveraged transfer learning by fine-tuning neural networks pretrained on image-level classification tasks, as well as [28], who used a conditional random field to encourage structure consistency in the produced segmentations. Perslev *et al.* [2] propose U-Time, an adaptation of the U-net architecture to time series segmentation, specifically the task of sleep staging. U-Time uses encoder-decoder skip connections as in [33], along with dilated convolutions in its encoder. It processes physiological time series sampled at 30-second intervals, producing dense predictions which are finally averaged over fixed intervals to obtain the final outcomes. U-Time achieved state-of-the-art results in all evaluated sleep staging datasets.

2.3.2 DeepLab Model and Variants

Papandreou *et al.* [68] train a deep CNN with a fully connected conditional random field on few, pixel-level-annotated images, combined with a large number of weakly-labeled ones. The latter consist of either bounding boxes or image-level labels, from which pixel-level segmentation information is extracted using expectation-maximization methods. While only training on image-level labels was found to not produce highly accurate results, the proposed approach almost matched the segmentation performance of fully-supervised, pixel-level segmentation models on the PASCAL VOC 2012 dataset. Papandreou *et al.* also leveraged

weak and pixel-level annotations from two related datasets, reporting a final mIoU of 73.9%. Chen *et al.* [31] identify three challenges in performing semantic image segmentation using deep CNNs originally designed for image-level classification: lower feature resolution due to downsampling operations, existence of objects at diverse scales, and lower boundary detection capacity due to the inherent translation invariance of CNNs. To address these issues, they propose Atrous Spatial Pyramid Pooling (ASPP) for multi-scale object segmentation, paired with a conditional random field to improve boundary localization. The main component of ASPP is *atrous convolution*, i.e., convolution with pixel skipping, which was independently investigated by Yu *et al.* [30] as *dilated convolution*. The authors replace the fully connected layers within the VGG-16 and ResNet-101 networks with convolutional layers as in [29], and combine atrous convolutions with bilinear interpolation for upsampling instead of using deconvolutions. The final model, DeepLabv2, achieved an mIoU of 79.7% on PASCAL VOC 2012.

Follow up work by the same authors introduces DeepLabv3 [69], which leverages a cascade of dilated convolutions followed by an improved ASPP module incorporating image-level features. Specifically, the new ASPP module consists of a single 1×1 and three 3×3 convolutions, while image-level features are obtained through global average pooling, a 1×1 convolution, and upsampling of the model's final feature map. DeepLabv3 significantly outperformed DeepLabv2, attaining an mIoU of 85.7% on the PASCAL VOC 2012 test set without using conditional random fields at post-processing time as was done in [31]. DeepLabv3+ [32] adopts an encoder-decoder architecture by using DeepLabv3 as an encoder and attaching a decoder module to recover the downsampled spatial information and improve segmentation near object boundaries. The authors also apply depthwise dilated convolution to the ASPP and decoder modules, splitting regular convolution into a faster sequence of depthwise and pointwise convolutions.

DeepLabv3+ was shown to outperform its predecessor, DeepLabv3, reaching an mIoU performance of 89.0% on PASCAL VOC 2012.

2.3.3 Optimizing Computational Requirements

Wu *et al.* [70] introduce a specialized upsampling module in place of the more computationally complex dilated convolutions often used in semantic segmentation. Using their proposed module to upsample an FCN's last three feature maps, they noted no decrease in segmentation performance. In addition, they reported a three-fold reduction in computation and memory requirements. Noting the reliance of deep semantic segmentation models on the availability of large annotated data, Zhu *et al.* [71] propose a video-based method for synthesis of training samples. This method leverages the ability of video prediction models to predict future frames for label generation. Specifically, the authors explore two approaches: label propagation, which forms new samples by pairing propagated labels with their corresponding original future frames, and joint image-label propagation, which instead combines propagated labels with their related propagated images. The experimental results showed that the latter training set augmentation technique improved prediction accuracy for segmentation networks. Arguing that color and texture should be examined separately from shape information for semantic image segmentation, Takiwawa *et al.* [72] propose Gated-SCNN, which processes these two streams of information in parallel and combines them through gated convolution layers. During training, the binary and standard cross entropy losses are jointly optimized to address the suggested duality between segmentation and boundary prediction. In the authors' experiments, this dual loss was shown to indeed improve the accuracy of boundary detection up to 3%.

2.4 Transportation Mode Identification

Given single-transportation-mode segments, obtained either via ground-truth labels or either of the trajectory segmentation methods discussed in Section 2.2, transportation mode identification is the task of classifying each segment by transportation mode.

2.4.1 Supervised Machine Learning Approaches

Most GPS-based studies, especially earlier ones, have used supervised machine learning for transportation mode identification. The seminal work of Zheng *et al.* [1] classifies transportation modes by combining the output of a decision tree classifier with the conditional probability of transitioning from the previous transportation mode to the predicted one. In their follow-up work, the authors instead model the probability distribution of travel modes by extracting a graph of density-based transportation change point clusters [34]. They also introduce three additional handcrafted features, namely the heading change rate, stop rate, and velocity change rate. Combining decision tree predictions with the knowledge of the transportation modes' distribution resulted in a higher accuracy-by-distance than the authors' previous work. Sun *et al.* [36] train an SVM to classify cars and trucks using features related to acceleration and deceleration. Since the data for the two classes were collected in the same city at different locations and time periods, these features were shown to be more discriminative than velocity-based ones. Although the authors reported a relatively low misclassification rate, their self-collected dataset has several non-negligible limitations which we briefly discuss in Section 2.6. Xiao *et al.* [26] train a Bayesian network classifier on six handcrafted features and reports high accuracy on the authors' self-collected dataset. Xiao *et al.* [73] train an eXtreme Gradient Boosting (XGBoost) ensemble classifier on GPS-track-level global features and segment-level local features.

Soares *et al.* [64] feed 60-second chunks of trajectory segments to a random forest classifier. However, the sample size of the proprietary dataset used in the authors' experiments may be too small to draw safe conclusions. Guo *et al.* [74] train a decision-tree-based ensemble classifier termed *deep forest* on 72 statistical features of velocity, acceleration, turning angle, and sinuosity.

Other machine-learning-based works have complemented their models with Geographic Information System (GIS) information. Stenneth *et al.* [35] train a random forest classifier on features including Euclidean distances from rail lines, bus stops, and real-time bus locations. This not only improved the accuracy of distinguishing between various motorized modes of transport, but also allowed for identifying which bus a given passenger had taken. Although the proposed approach attained high precision and recall, it was evaluated on a dataset whose total duration is just three hours. Bolbol *et al.* [60] use an SVM to estimate the transportation mode for consecutive overlapping triples of GPS points and identifies change points whenever the mode of two successive triples is different. Shah *et al.* [75] train a decision tree classifier on time- and frequency-domain features of accelerometer data to distinguish between motorized and non-motorized motion. After computing motion features from GPS data in conjunction with transit route and location information, they train another decision tree to classify above-ground transportation modes. Finally, a rule-based classifier is used to identify mixed-mode and underground travel. Simoncini *et al.* [76] explore a low-frequency GPS sampling scenario where GPS points are sampled once every 90 or 120 seconds. They train an SVM to classify light- and heavy-duty vehicles using the same features as reference [36] together with distance- and velocity-related ones. Using recursive feature elimination, the authors identified 69 highly discriminative features, using them to train an SVM on their self-collected dataset. After classifying each track individually and then aggregating the results of mul-

tiple tracks for each vehicle, they achieved an even higher Area Under Curve (AUC). Although the experimental results seem promising, the final aggregation step and the need for reverse geocoding to calculate the road type feature might hinder the performance of the proposed framework in applications with real-time requirements.

2.4.2 Supervised Deep Learning Methods

With the exception of Gonzalez *et al.* [77], the literature has only recently begun to explore deep-learning-based transportation mode identification. Gonzalez *et al.* [77] train an MLP on subsets of the original GPS points consisting of *critical points*, selected such that the original GPS traces could be approximately recreated from them. This work was one of the first to effectively apply neural networks to transportation mode identification. However, it is limited by only considering three transportation modes, using a rather small dataset, and requiring manual trajectory segmentation. Endo *et al.* [56] generate image-like representations for each labeled trajectory segment by clipping a rectangular grid of GPS points sampled at a fixed time interval and setting the pixel value of each grid according to the number of its GPS points. These images are then used to train a stacked denoising autoencoder, with the learned features of its last hidden layer being concatenated with the handcrafted ones proposed in reference [34] to finally train a logistic regression classifier. Mäenpää *et al.* [78] investigate the usefulness of several features for training an MLP on sparsely sampled GPS data. They found spectral bins to be the most significant features, while auto- and cross-correlations, kurtoses, and skewnesses of velocities and accelerations were deemed relatively ineffective. Wang *et al.* [42] design point-level features to train a sparse autoencoder and use sequences of the learned point-level features to train a CNN and learn trajectory-level features. They finally combine the latter

with handcrafted trajectory-level features to train a 2-hidden-layer MLP classifier. After dividing trajectories into fixed-size segments and extracting pointwise motion features, Dabiri and Heaslip [23] train an ensemble of seven CNNs and averages their predictions for the final classification. Following a similar approach, Yazdizadeh *et al.* [79] train a CNN ensemble as a base learner and uses a random forest classifier as a meta-learner. Zhang *et al.* [46] combine the DenseNet convolutional architecture with the attention mechanism to learn both transportation modes and speeds. To capture information in different granularities, the authors map GPS trajectories to multi-scale grids, using coarser and finer ones separately to train the two sub-networks of their neural architecture. Yu [44] preprocesses GPS trajectories into multivariate time series of motion features and estimates their frequency-domain counterparts via discrete Fourier and wavelet transforms, using both sets of features to train an LSTM classifier and achieving 92.7% accuracy on Geolife [1, 34].

A number of deep-learning-based works have also integrated GIS information. For instance, Zhu *et al.* [66] incorporate features measuring closeness to bus lines and subway stops before performing unsupervised training of a Stacked AutoEncoder (SAE), finally attaching a softmax layer for supervised learning. Song *et al.* [80] train a deep Long Short-Term Memory (LSTM) architecture on GPS and GIS data to jointly learn users' movements and selected transportation modes. In the proposed architecture, the input encoding layer is followed by two shared hidden layers and an output decoding layer. Simoncini *et al.* [43] train a deep LSTM architecture to classify three vehicle weight classes using low-frequency GPS data. In addition to the standard LSTM layers, the authors include input-to-recurrent and recurrent-to-output feedforward layers. To handle class imbalance, the loss function is weighted by the inverse of the class sizes. While including GIS information can help improve performance, it may also not

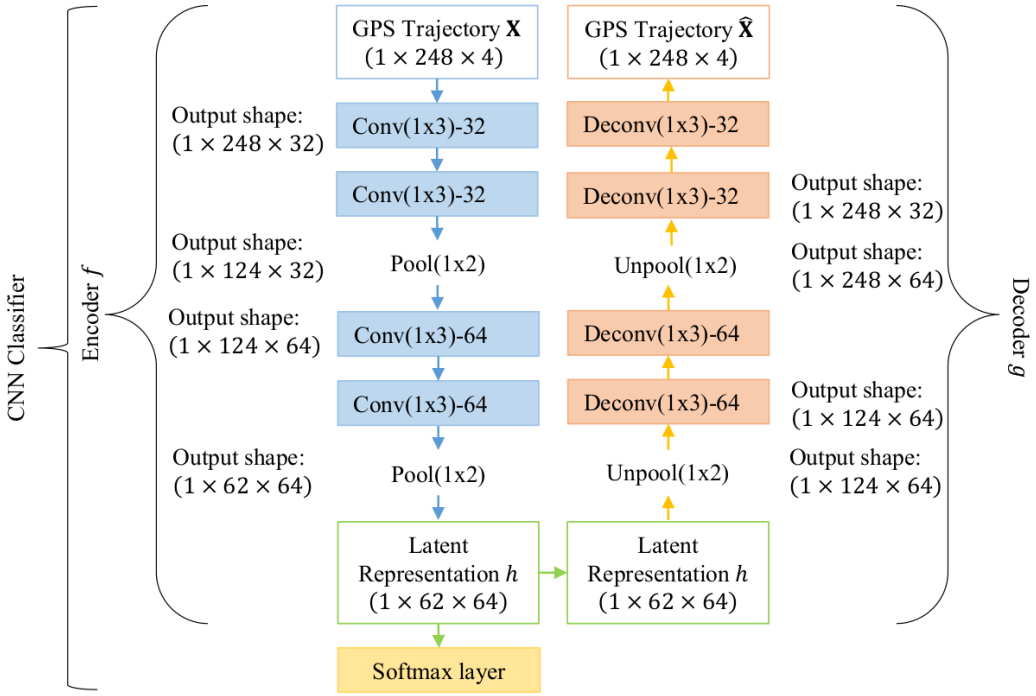

Figure 2.3 : The SECA architecture [3] jointly trains a convolutional autoencoder and a CNN classifier.

always be up to date or even available for certain areas.

2.4.3 Semi-Supervised and Unsupervised Deep Learning Frameworks

A major limitation to the real-world applicability of transportation mode identification is the lack of GPS trajectories labeled by transportation mode. This can be attributed in part to the fact that GPS sensors do not readily capture travel mode information. Some users may be unwilling to manually label their own trips, as doing so both accurately and consistently requires significant individual effort. For others, such a process may raise privacy concerns. To the best of our knowledge, only a few works have attempted to alleviate the dependency of transportation mode identification methods on the availability of labeled data [81, 58, 3, 45, 57].

Some have investigated semi-supervised deep learning approaches [58, 3, 45]. Using a label propagation algorithm with the k-Nearest Neighbors (k-NN) kernel, Rezaie *et al.* [58] achieved competitive accuracy using fewer than 30% of labels. A limitation to this work is that each segment was assigned its predominant travel mode during preprocessing, a practice that simplifies trajectory segmentation at the expense of producing noisy[†] segments. Dabiri *et al.* [3] jointly train a convolutional autoencoder and a CNN classifier for semi-supervised transportation mode identification by optimizing the weighted sum of their losses. The proposed SEmi-supervised Convolutional Autoencoder (SECA) architecture is illustrated in Figure 2.3. At preprocessing time, arbitrary-length GPS trajectories are first split into non-overlapping fixed-size segments and converted into velocity and acceleration time series. Next, these are fed to an optimization-based algorithm for travel mode change point detection. After the identified change points are used for trajectory resegmentation, four velocity- and acceleration-based features are extracted from each resulting GPS segment and used to train the two networks within SECA. During training, their losses are initially balanced until convergence, with the supervised loss gradually being assigned more weight. Compared to supervised machine and deep learning baselines, the proposed method consistently achieved higher accuracy when using as few as 20% of the labeled data. Yu [45] extracts the same features as in reference [44] before feeding them to a stacked LSTM to extract their latent representations. The resulting three sets of features are finally used in multi-view semi-supervised training of an LSTM ensemble. The latter consists of a *main* network trained on all three feature sets and three *view* networks trained on different subsets thereof. During training, originally unlabeled trajectory segments are assigned proxy labels if either all three

[†]In the context of transportation mode identification, trajectory segments are referred to as *noisy* if they involve multiple travel modes.

view networks predict the same travel mode or at least one of them matches the *main* network's prediction. The proposed framework significantly outperformed reference [3], achieving an accuracy of nearly 85% on Geolife [1, 34] using just 1% of labeled data.

Another way to address the lack of labeled GPS trajectories involves leveraging generative models to create synthetic annotated samples. For example, Li *et al.* [57] first train a GAN conditioned on transportation mode classes to balance the Geolife dataset [1] via upsampling. Both ground truth and synthetic motion features are then used in supervised training of a CNN classifier. Nonetheless, the reported deviation between the original and generated feature distributions suggests room for improvement of the data generation process.

To the best of our knowledge, only the work of Patterson *et al.* [81] has investigated fully unsupervised learning in the context of transportation mode identification. The proposed method represents street maps as graphs and integrates information on bus stops and routes, using graph-constrained particle filters to estimate both transportation modes and routes in an unsupervised manner. Although it achieved promising accuracy on the authors' self-collected GPS dataset, this framework depends on the availability of up-to-date GIS information and cannot generalize to locations beyond those adequately covered in the training set.

We note that closely related to transportation mode identification is the task of vehicle classification, which aims to classify vehicles into fine-grained classes. For instance, some works attempt to distinguish between cars and taxis [56, 46], trains and subways [56, 73, 46], or even various types of ships [46]. Although methods used in one research area may be useful for the other [36], features typically used for transportation mode identification may not be as effective for

vehicle classification [43]. Nonetheless, we have summarized a few such studies due to their relevance to this research.

2.5 Deep Clustering

The frameworks proposed in Chapters 5 and 6 perform unsupervised transportation mode segmentation and identification of GPS trajectories by clustering their associated motion feature sequences at timestep- and sequence-level, respectively. This methodology is tied to a set of techniques for clustering data via unsupervised deep learning, referred to in the literature as *deep clustering*.

2.5.1 Autoencoder-Based Approaches

Earlier deep clustering studies first perform autoencoder pretraining using the original high-dimensional inputs and cluster the learned lower-dimensional representations by merely minimizing a clustering-specific objective. Yang *et al.* [47] apply agglomerative clustering to autoencoder-learned features. The proposed method starts with a large number of clusters, gradually merging them on the forward pass and learning the embedding on the backward pass. In their seminal work, Xie *et al.* [48] propose Deep Embedded Clustering (DEC) to simultaneously learn a low-dimensional feature space and cluster assignments in a self-supervised manner. After pretraining a 4-layer denoising Stacked AutoEncoder (SAE) to obtain an initial target distribution, they replace the decoder with a clustering layer maintaining cluster centroids as trainable weights, initialized using the k-Means algorithm. At each iteration, the clustering layer assigns soft labels to each training sample using Student's t-distribution formula; these soft labels are then used to estimate the target distribution and calculate the clustering loss using Kullback-Leibler (KL) divergence. Instead, Li *et al.* [49] attach a soft k-Means layer in place of the decoder and retrain the composite model with

samples of ascending (estimated) clustering difficulty.

More recently, joint optimization of the autoencoder reconstruction loss together with a clustering loss has generated better results. Unlike reference [48], Guo *et al.* [82] maintain the decoder to preserve the representation space learned by minimizing the reconstruction loss at pretraining time. They then add a weighted clustering loss term to the objective function and retrain the composite model. Similarly to Xie *et al.* [48], the clustering loss estimates the KL divergence between the cluster assignments and the target distribution. The authors' experimental results using a 4-layer denoising SAE trained on MNIST showed that maintaining the decoder indeed improved the clustering accuracy. Guo *et al.* [83] expand on their previous work by replacing the SAE with a Convolutional AutoEncoder (CAE). However, the reported clustering accuracy of the proposed Deep Convolutional Embedded Clustering (DCEC) on MNIST was not significantly higher than merely training the CAE and applying k-Means clustering to the encoded data. Yang *et al.* [84] instead estimate the clustering loss based on the mean squared error and update the network weights, cluster memberships, and centroids separately. The proposed Deep Clustering Network (DCN) handles class imbalance by updating cluster assignments more rigorously for clusters having fewer members than others. Despite showing promising results, DCN often did not significantly outperform the baseline of pretraining the SAE and applying k-Means clustering to the learned latent representations. To address class imbalance, Ghasedi *et al.* [50] design a denoising autoencoder and instead minimize the KL divergence between predicted and target distributions (as in reference [48]) with the addition of a uniform prior controling the frequency of cluster assignments.

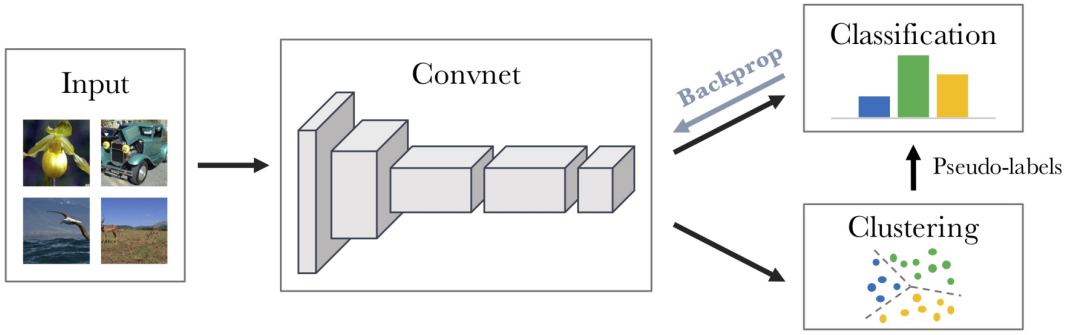

Figure 2.4 : The DeepCluster architecture [4] iterates between clustering and leveraging cluster assignments as pseudo-labels to train the model.

2.5.2 Traditional Clustering on Pretrained Network Outputs

Caron *et al.* [4] propose DeepCluster, a framework trained by iteratively using k-Means to group the output of a convolutional neural network (pretrained using supervision on other datasets) and leveraging the obtained cluster assignments as ground-truth labels to optimize its parameters. If a cluster becomes empty, the proposed method first selects one of the remaining centroids at random, then adds some noise to it in order to generate a new one, and finally distributes all points of the former centroid among the two. Moreover, to handle class imbalance, samples are selected using a uniform distribution over the pseudo-labels. DeepCluster, an overview of which is shown in Figure 2.4, demonstrated promising performance on established image datasets using the AlexNet and VGG-16 CNN architectures. However, DeepCluster relies on the availability of CNNs that have been pretrained using supervised learning, and applies k-Means to features reduced via Principal Component Analysis (PCA) rather than include dimensionality reduction in the learning process.

2.5.3 Generative Modelling

Another body of deep clustering research leverages generative models such as variational autoencoders [51] and GANs [52]. Jiang *et al.* [51] propose Variational Deep Embedding (VaDE), whereby a Gaussian Mixture Model (GMM) first selects a cluster from which an embedding is generated and subsequently decoded by a Variational AutoEncoder (VAE) into an observation. Compared to frameworks based on standard autoencoders, like DEC [48], its generative nature allows VaDE to produce and cluster new high-fidelity samples. Noting that traditional GAN formulations do not involve encoder networks and thus do not directly allow for clustering in the latent space, Mukherjee *et al.* [52] train a GAN together with an encoder network by incorporating a clustering loss in the standard GAN objective function.

2.5.4 Mutual Information Maximization

It is also viable to cluster data by exploiting the mutual information between samples and their generated augmentations or learned latent representations [53, 54, 55]. Hu *et al.* [53] encourage neural networks to output similar feature maps for input samples and their augmented versions, while also maximizing the mutual information between samples and their latent representations. Hjelm *et al.* [54] instead maximize the mutual information between latent representations and local input regions, at the same time matching the former to a prior distribution. Finally, Ji *et al.* [55] maximize the mutual information between the latent representations of pairs of samples and their augmentations, improving performance by simultaneously training an auxiliary overclustering component.

2.6 Conclusions

For GPS trajectory data to be safely used in downstream applications, naturally missing GPS points due to signal lapses must first be imputed. This is especially true for the task of transportation mode identification, which is typically performed on sequences of motion features extracted from consecutive GPS points in a pairwise manner: the more GPS points are missing, the noisier the estimated motion features. Nonetheless, the effectiveness of existing approaches is conditioned on either the availability of historical trajectories and knowledge of the underlying map [7, 8, 9], which may not always be up to date, or on access to complete trajectories at model training time [10].

Before performing transportation mode identification, it is necessary to first identify trips within users' GPS trajectories from which single-transportation-mode segments can then be extracted [56, 3]. While detecting trips has been shown to be more straightforward, with simple time-based heuristics often producing satisfactory results [1], transportation mode segmentation remains an open problem, especially in the absence of transportation mode labels. To date, researchers have predominantly approached trajectory segmentation as transportation mode change point detection, using mobility-based heuristics [1, 24, 25, 26, 27] and more recently optimization-based algorithms [3]. However, such methods depend on extensive feature engineering or transportation domain knowledge and assume independent and identically distributed samples, respectively. In addition, heuristics alone may not sufficiently account for the complexity of all possible traffic scenarios.

To date, GPS-based transportation mode identification has been tackled using a wide range of machine learning models, including but not limited to Bayesian networks [26], decision trees [1, 34], random [35] or deep forests [74], and

support vector machines [36]. More recently, deep learning has increasingly gained popularity in the transportation domain, with researchers successfully applying MLPs [42] and convolutional [23, 3, 46, 79] or recurrent neural networks [43, 44, 45] to transportation mode identification. With the exception of references [81, 75, 3, 45], however, most of the above studies have used models trained in fully supervised settings. As such, they have largely relied on the availability of ground-truth labels, which is often limited in real-world applications, at the same time failing to incorporate the typically much larger amounts of available unlabeled data.

As a final remark, it is worth noting that several transportation mode segmentation and identification studies were evaluated on self-collected datasets, which makes it difficult to draw safe conclusions when comparing the effectiveness of their proposed methods. Self-collected datasets also risk containing bias; for instance, Sun *et al.* [36] identified bias in their dataset due to the trucks' GPS devices being automatically turned off when velocity fell below 2 m/s. Furthermore, many such datasets are limited by either their small sample size or total duration [60]. To avoid the above issues, the methods proposed in this thesis are primarily evaluated on the openly available Geolife dataset [34, 1] released by Microsoft Asia Research, as it has been established as a benchmark in transportation research.

Chapter 3

Uncertainty-Aware Generative Trajectory Imputation

In this chapter, we propose to *indirectly* learn how to estimate missing GPS points by setting an alternative target: imputing point-wise motion features measuring magnitude and angle of displacement, i.e., relative distance and bearing (see Figure 3.1). To this end, we introduce an Uncertainty-aware Imputation Generative Adversarial Network (UI-GAN) trained on incomplete motion features extracted from incomplete GPS trajectories. In UI-GAN, the Bayesian generator G performs Monte Carlo (MC) dropout sampling [85] for multiple imputation and uncertainty estimation, while the discriminator D examines real and imputed sequences at window-level. Inspired by recent work in computer vision [86, 87], both G and D leverage missingness-gated temporal convolutions to handle originally missing data. UI-GAN is shown to generate high-fidelity GPS points from imputed motion features, outperforming competitive baselines on a large, real-world dataset.

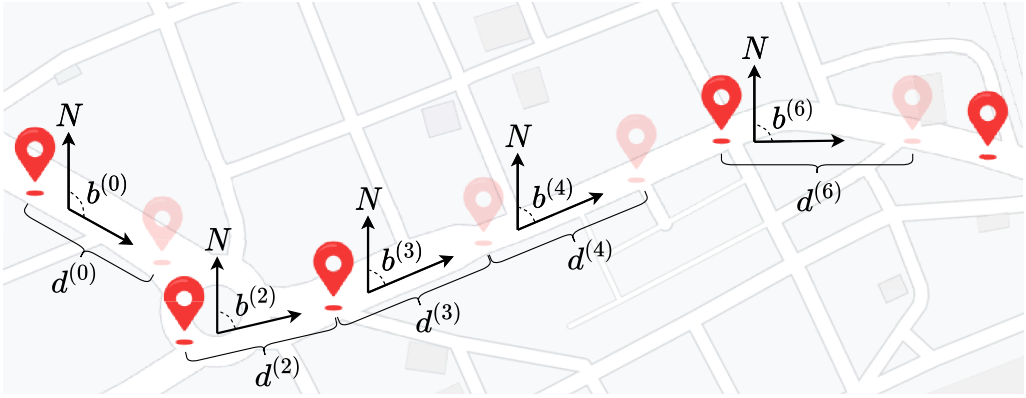

Figure 3.1 : Missing (transparent) GPS points are estimated from observed (opaque) or previously recovered ones after imputing relative distance d and angle b between initial direction and true north.

3.1 Preliminaries

This section first defines domain-related terms and formalizes the problem to be addressed in this chapter, before introducing uncertainty quantification via Bayesian deep learning.

3.1.1 Incomplete GPS Trajectory Reconstruction via Motion Feature Imputation

Definition 3.1 (GPS Point). We denote GPS point $\mathbf{p}$ as a 4-tuple $\langle \mathrm{t}, \mathrm{lat}, \mathrm{lon}, \mathrm{mode} \rangle$, where t measures the decimal number of days since a reference date, $-90 \leq \mathrm{lat} \leq 90$ and $-180 \leq \mathrm{lon} \leq 180$ are latitude and longitude coordinates in decimal degrees, and $\mathrm{mode} \in \{\mathrm{walk}, \mathrm{bike}, \mathrm{bus}, \mathrm{car}, \mathrm{train}\}$ denotes the transportation mode label associated with $\mathbf{p}$. More details regarding the target transportation mode classes will be presented in Section 4.3.1.

Definition 3.2 (GPS Trajectory). We represent GPS trajectory $\mathbf{T}$ as a temporally ordered sequence $\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\|\mathbf{T}\|-1)}\}$ of arbitrary length $\|\mathbf{T}\| \in \mathbb{Z}^+$. Since the frameworks proposed in this thesis are all based on deep convolutional

neural networks (thus requiring fixed-size inputs), we partition $\mathbf{T}$ into $\lfloor \|\mathbf{T}\|/L \rfloor$ non-overlapping sequences of fixed length $L > 0$.

Definition 3.3 (Motion Feature Sequence). Directly training machine or deep learning models on raw GPS data inadequately captures spatiotemporal information and biases generalization towards the locations best covered in the training set. As such, we follow established transportation mode identification literature [1, 3, 44] in preprocessing each GPS trajectory $\mathbf{T}$ into a motion feature sequence $\mathbf{X} = \{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\|\mathbf{T}\|-1)}\}$ by transforming each $\mathbf{p} \in \mathbf{T}$ into a vector $\mathbf{x} \in \mathbb{R}^N$ of N motion features. This process will be detailed in Section 4.2.1.

Definition 3.4 (Masking Matrix). Incomplete GPS trajectories $\mathbf{T}$ produce incomplete motion feature sequences $\mathbf{X}$. For each $\mathbf{X} \in \mathbb{R}^{L \times N}$, we construct a masking matrix $\mathbf{M} \in \{0, 1\}^{L \times N}$ such that $\mathbf{M}_{l,n} = 1$ if $\mathbf{X}_{l,n}$ is observed or $\mathbf{M}_{l,n} = 0$ otherwise. Unobserved GPS points and their associated motion features in $\mathbf{X}$ are identified and inserted based on elapsed time and velocity thresholds, as per Section 5.3.1.

Problem 3.1 (GPS Trajectory Reconstruction). Given an incomplete GPS trajectory $\mathbf{T} \in \mathbb{R}^{L \times 3}$ preprocessed into a sequence of incomplete motion features $\mathbf{X} \in \mathbb{R}^{L \times N}$, we propose to impute the missing motion features indexed by masking matrix $\mathbf{M} \in \{0, 1\}^{L \times N}$ to produce the complete sequence $\tilde{\mathbf{X}} \in \mathbb{R}^{L \times N}$ and use it to estimate the missing GPS points in $\mathbf{T}$, thus obtaining the reconstructed trajectory $\tilde{\mathbf{T}}$. The process of determining raw GPS points based on imputed motion features and previously observed or imputed points will be detailed in Section 3.2.2.

3.1.2 Predictive Uncertainty Quantification for Bayesian Neural Networks

Bayesian modelling considers *epistemic* uncertainty tied to model weights and heteroscedastic *aleatoric* uncertainty due to input noise that varies for different observations. Capturing the epistemic uncertainty of a neural network

requires placement of a prior distribution over its weights followed by calculation of the posterior. The latter is typically intractable, as it requires integration with respect to the space of all network parameters which often lacks a closed form [85].

In the proposed framework, the generator estimates the posterior by approximating variational inference using MC dropout sampling, and captures aleatoric uncertainty by regressing observation noise $\mathbf{S} \in \mathbb{R}^{L \times N}$ as an additional model output [85]. The former corresponds to injecting the generator's otherwise deterministic weights with noise sampled from a Bernoulli distribution, hence creating their stochastic counterparts. In practice, we apply dropout with probability of dropping connections p_{drop} after each convolution layer except for the output at both training and inference time. For inference, we perform S stochastic forward passes to obtain imputations $\hat{\mathbf{X}} \in \mathbb{R}^{S \times L \times N}$, which are averaged to produce the predictive mean $\mathbf{X}' \in \mathbb{R}^{L \times N}$.

Given S MC dropout samples, the total predictive uncertainty for the l-th timestep $\hat{\mathbf{X}}_{:,l,:}$ can be approximated by its variance as follows [85]:

$$\text{Var}(\hat{\mathbf{X}}_{:,l,:}) \approx \underbrace{\frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{X}}_{s,l,:}^2 - \left(\frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{X}}_{s,l,:}\right)^2}_{\text{epistemic}} + \underbrace{\frac{1}{S}\sum_{s=1}^{S} \hat{\mathbf{S}}_{s,l,:}^2}_{\text{aleatoric}}. \tag{3.1}$$

3.2 Proposed Framework

This section first introduces the Missingness-Gated Temporal Convolution (MGT-Conv) block proposed for operating on time series with missing data. Next, it describes the generator and discriminator networks that form our GAN-based approach to GPS trajectory imputation. Finally, it explains how the generator approximates variational inference to both estimate predictive uncertainty and reduce imputation error.

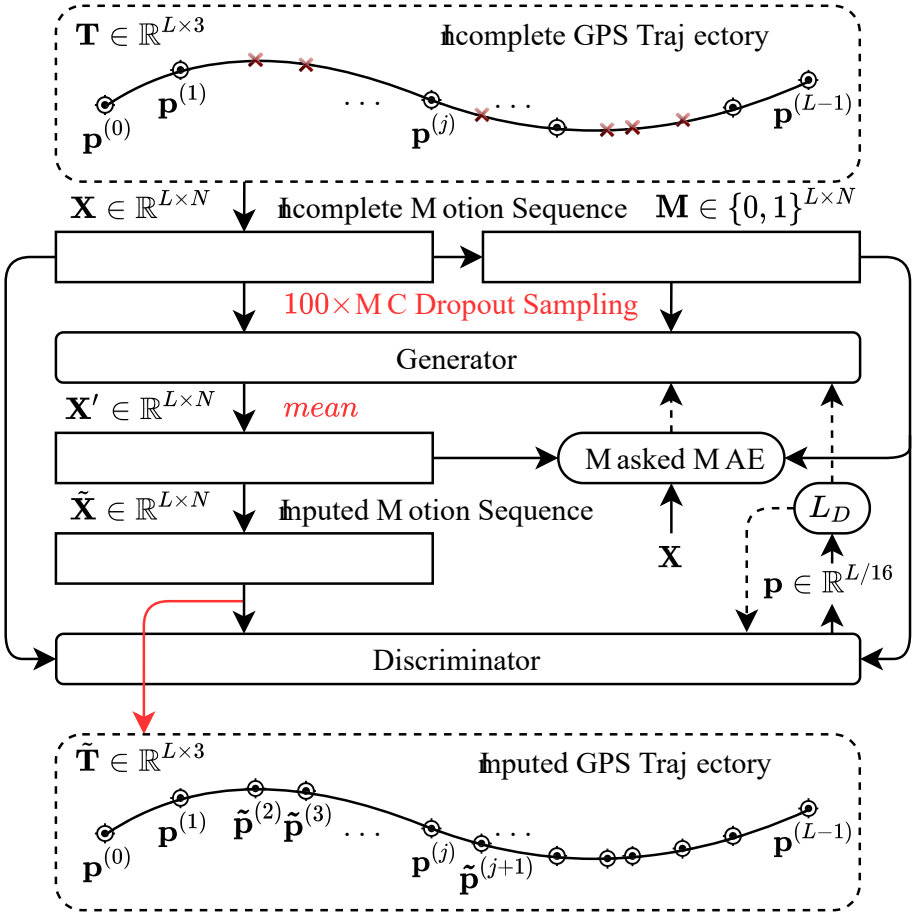

Figure 3.2 : Overview of the proposed UI-GAN. Dashed arrows indicate gradient flow, while operations in red only occur at test time.

An overview of the proposed UI-GAN is given in Figure 3.2. At training time, incomplete time series of motion features are extracted from incomplete GPS trajectories and fed to the generator G, which uses MGTConv blocks to handle unobserved values. G learns to produce better imputations by reconstructing the observed parts of its input and by convincing the discriminator D that the reconstructed sequences are real. D penalizes sequences at window-level, leveraging MGTConv blocks to process missing features within real and imputed data. At inference time, incomplete motion feature sequences are similarly extracted and then sampled from the trained G multiple times while dropout is activated to approximate variational inference. These aggregated outputs are averaged to obtain the final imputation, while their variance is used to estimate aleatoric and epistemic uncertainties. Finally, imputed relative distances and bearings are used to estimate raw GPS points.

3.2.1 Missingness-Gated Temporal Convolutions

Feature learning based on standard temporal convolutions involves sliding the same trainable filters over the entire input sequence. Such practice is reasonable when every timestep holds valid data, but becomes problematic when feature values are missing: the latter must still be numerically represented with placeholder constants. Applying the same convolutional filters to both valid and invalid features is undesirable, as it creates ambiguity during training [86]. Although one could simply use RNNs, where masked processing is supported by design, temporal convolutional networks benefit from high parallelism, lower memory requirements, and have even outperformed recurrent architectures* in established sequence modelling tasks [88].

Inspired by recent work on image inpainting [89, 86] and occluded pose es-

*This excludes Transformers, as they do not involve recurrence.

timation [87], we design missingness-gated temporal convolution blocks for the generator (Figure 3.3(a)) and discriminator (Figure 3.3(b)) networks in UI-GAN to effectively process incomplete time series. Note that, while $\mathbf{X}$ contains missing features, $\mathbf{M}$ does not: the *indices* where data are available are in fact fully observed. Therefore, our intuition is that $\mathbf{M}$ should be processed separately from $\mathbf{X}$, so that the fully observed information encoded in $\mathbf{M}$ is preserved in deeper layers. In addition, using intermediate feature maps of $\mathbf{M}$ to gate those of $\mathbf{X}$ at the l-th block may attenuate the influence of missing values on the next block and ultimately on the generated motion features.

Excluding bias terms for readability, the discriminator's MGTConv block is defined as:

$$\bar{\mathbf{X}}^{(l)} = \phi(\mathbf{X}^{(l-1)} * \mathbf{W}_f^{(l)}), \tag{3.2}$$

$$\mathbf{M}^{(l)} = \sigma(\mathbf{M}^{(l-1)} * \mathbf{W}_g^{(l)}), \tag{3.3}$$

$$\mathbf{X}^{(l)} = \bar{\mathbf{X}}^{(l)} \odot \mathbf{M}^{(l)}, \tag{3.4}$$

where $\mathbf{W}_f^{(l)}$, $\mathbf{W}_g^{(l)}$ are convolution filters, ϕ, σ are the Leaky Rectified Linear Unit (LReLU) and sigmoid activation functions, $\bar{\mathbf{X}}^{(l)}$ is the intermediate output of the l-th block before gating, and $\odot$ is the Hadamard product that produces the final gated output $\mathbf{X}^{(l)}$. The generator's version of the MGTConv block is described in Section 3.2.2.

3.2.2 Bayesian Generator

Architecture The generator G is built by stacking MGTConv blocks (Figure 3.3(a)). For G, the convolutions in Eqs. (2), (3) are dilated, applying filters over a larger area than their own by skipping input values with step d, otherwise known as dilation rate. This allows convolution layers to expand their receptive fields as the network depth grows, preserving input resolution without need for

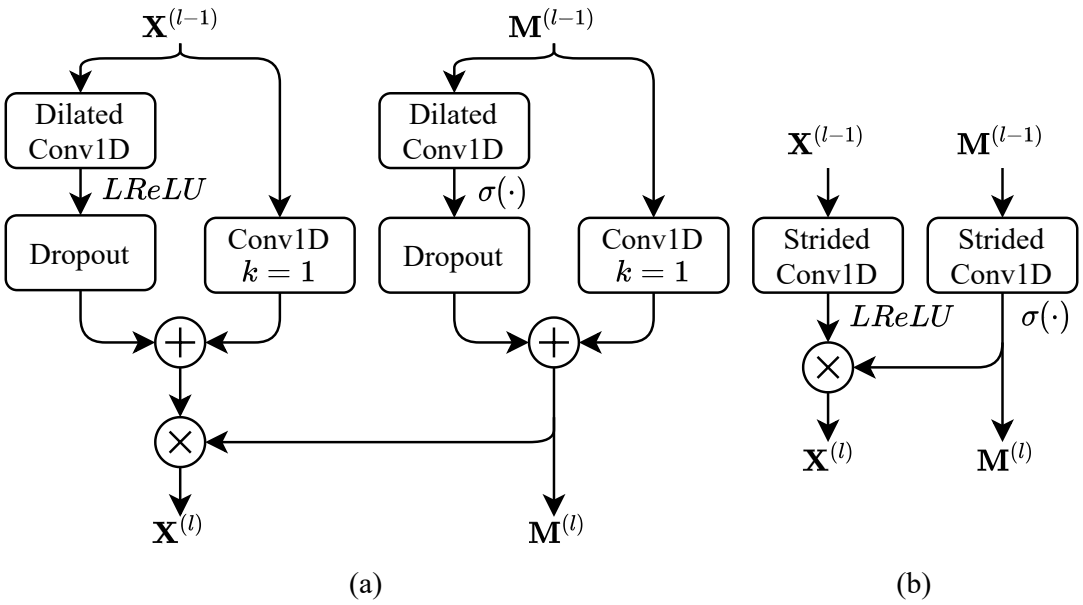

Figure 3.3 : (a) The generator's MGTConv block applies dilated 1D convolutions to capture high-resolution temporal information without need for downsampling. Always-on dropout is used to approximate variational inference via MC dropout sampling. (b) In the discriminator's MGTConv block, strided convolutions downsample the input and enlarge D's receptive field.

downsampling. Skip connections are also added to facilitate gradient flow. Note that both residual paths have an optional convolution with unit kernel size applied to $\mathbf{X}^{(l-1)}$ to ensure that the number of input channels equals the number of output channels; this is only required for the block following the input layer. Finally, dropout layers are inserted to approximate variational inference at test time, as per Section 3.1.2.

Missing values in the incomplete input sequence $\mathbf{X}$ are estimated by feeding it to G and superimposing $G(\mathbf{X}) = \mathbf{X}'$ over $\mathbf{X}$ at the indices where features are missing, as indicated by $\mathbf{M}$, producing the complete (imputed) sequence $\tilde{\mathbf{X}}$ as follows:

$$\tilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{M} + \mathbf{X}' \odot (\mathbb{1} - \mathbf{M}), \tag{3.5}$$

where $\mathbb{1}$ is a matrix of ones with the same dimensionality as $\mathbf{M}$ and $\odot$ is the Hadamard product.

The generator is trained by optimizing the weighted sum of the adversarial hinge loss [90] and the Mean Absolute Error (MAE) between true and generated features at the indices where true ones are observed:

$$L_G = -\mathbb{E}_{\tilde{\mathbf{X}} \sim P_{\tilde{\mathbf{X}}}}[D(\tilde{\mathbf{X}})] + \frac{\lambda}{L} \sum_{l=1}^{L} \mathbf{M}_{l,:} \left(\frac{1}{2} \exp(-\log \hat{\mathbf{S}}^2_{l,:}) \left| \mathbf{X}_{l,:} - \mathbf{X}'_{l,:} \right| + \frac{1}{2} \log \hat{\mathbf{S}}^2_{l,:} \right), \tag{3.6}$$

where $P_{\tilde{\mathbf{X}}}$ is the distribution of imputed motion feature sequences and hyperparameter λ controls the influence of the MAE loss. Note that we regress $\log \hat{\mathbf{S}}^2_{l,:}$ instead of $\hat{\mathbf{S}}^2_{l,:}$ for numerical stability.

Estimating GPS Points from Imputed Motion Features After imputing the relative geodesic distance $d^{(i)}$ and bearing $b^{(i)}$ between $\mathbf{p}^{(i)}$ and its missing successor

$\mathbf{p}^{(i+1)}$ as per Section 5.3.1, the latter's coordinates are estimated as follows:

$$\begin{aligned}\text{lat}^{(i+1)} &= \arcsin(\sin(\text{lat}^{(i)}) * \cos(d^{(i)}/R) \\ &\quad + \cos(\text{lat}^{(i)}) * \sin(d^{(i)}/R) * \cos(b^{(i)})), \qquad (3.7a)\\ \text{lon}^{(i+1)} &= \text{lon}^{(i)} + \arctan2(\alpha, \beta), \qquad (3.7b)\\ \alpha &= \sin(b^{(i)}) * \sin(d^{(i)}/R) * \cos(\text{lat}^{(i)}), \qquad (3.7c)\\ \beta &= \cos(d^{(i)}/R) - \sin(\text{lat}^{(i)}) * \sin(\text{lat}^{(i+1)})), \qquad (3.7d)\end{aligned}$$

where all coordinates are in radians, and $d^{(i)}$, R (the Earth's radius) are in kilometers. Note that $\mathbf{p}^{(i)}$ may either be observed or previously imputed.

3.2.3 Window-level Discriminator

Should D penalize entire motion feature sequences at once, or is it more informative to consider smaller segments independently? The answer may depend on both the temporal dependencies between mobility patterns encoded in each trajectory, as well as the percentage of missing GPS points. The latter implies that, since trajectories with fewer missing points are easier to impute and harder for D to detect, it may benefit D to examine sequences in shorter windows.

Therefore, we design D as a modification to the PatchGAN discriminator [91] originally proposed for patch-level image-to-image translation. By replacing all standard convolutions with MGTConv blocks as per Section 3.2.1 (Figure 3.3(b)), D can now handle incomplete GPS trajectories. In each block, D applies strided convolutions for downsampling, thus increasing the window length (receptive field) W_D by which the input is processed. W_D is a function of the stride s, the kernel size k, and the number of layers; for instance, a 4-layer architecture with $s = 2$, $k = 5$, will result in D examining sequences of length L using $W_D = 61$ and output shape $\mathbb{R}^{L/16}$.

The discriminator receives real sequences $\mathbf{X}$ and imputed sequences $\tilde{\mathbf{X}}$. Cru-

cially, both $\mathbf{X}$ and $\tilde{\mathbf{X}}$ are masked at the indices where features are originally missing in $\mathbf{X}$ (as indexed by its associated $\mathbf{M}$), to ensure that D discriminates based on originally observed data only. Thus, D optimizes the hinge version of the adversarial loss [90]:

$$L_D = \mathbb{E}_{\mathbf{X}\sim P_{\mathbf{X}}}[\max(0, 1 - D(\mathbf{X}))] + \mathbb{E}_{\tilde{\mathbf{X}}\sim P_{\tilde{\mathbf{X}}}}[\max(0, 1 + D(\tilde{\mathbf{X}}))], \tag{3.8}$$

where $P_{\mathbf{X}}$, $P_{\tilde{\mathbf{X}}}$ are the distributions of input and imputed motion feature sequences.

3.3 Experiments

After detailing our experimental setup, this section presents and analyzes our results.

3.3.1 Dataset and Simulation Setup

We conducted our experiments on a server equipped with an Intel Xeon Silver 4210 CPU clocked at 2.20GHz and NVIDIA GeForce RTX 2080Ti GPUs with 11 GB of GDDR6 memory. The reported results were averaged over 5 runs.

Dataset We evaluate the proposed approach on the GAIA open dataset[†] by DiDi Chuxing. GAIA contains ride-hailing drivers' trajectories obtained in November 2016 in Chengdu, China at sampling intervals of $2-4$ seconds, totalling nearly 2 billion GPS points; this corresponds to about 66 million GPS points per day. As such, we select all data from November 1 for our experiments. After preprocessing them according to the following subsection and dividing them into fixed-size segments of length $L = 128$, we obtain $120,000$ samples and generate training, validation, and test sets using a 80/10/10 split.

[†]Available at https://gaia.didichuxing.com.

Although evaluation on an additional dataset would be ideal, to the best of our knowledge no other real-world GPS dataset exists that is both openly available and densely sampled.[‡] While the Geolife dataset [1, 34] by Microsoft Research Asia satisfies these requirements and includes numerous trajectories involving several transportation modes, it does not contain enough trajectories of a *single* transportation mode to train deep neural networks on. Geolife will, however, be used in Chapters 4, 5 and 6 to evaluate the transportation mode segmentation and identification frameworks that will be introduced.

Preprocessing Missing GPS points in trajectory $\mathbf{T}$ with mode sampling interval t_m are identified and inserted via placeholders after $\mathbf{p}^{(i)}$ when both $2\text{m/s} < v^{(i)} < 50\text{m/s}$ and $\lfloor((t^{(i+1)} - t^{(i)}) - t_m)/t_m\rfloor > 1$. We thereby estimate that nearly 10% of GPS points in our training set are missing. For each GPS point $\mathbf{p}^{(i)}$, we then compute the relative geodesic distance $d^{(i)} = \text{Geodesic}(\text{lat}^{(i)}, \text{lon}^{(i)}, \text{lat}^{(i+1)}, \text{lon}^{(i+1)})$ (in meters), velocity $v^{(i)} = d^{(i)}/(t^{(i+1)} - t^{(i)})$, acceleration $a^{(i)} = (v^{(i+1)} - v^{(i)})/(t^{(i+1)} - t^{(i)})$, and jerk $k^{(i)} = (a^{(i+1)} - a^{(i)})/(t^{(i+1)} - t^{(i)})$. After converting latitudes and longitudes to radians, we also calculate the bearing $b^{(i)} \in [0, 360]$:

$$b^{(i)} = \arctan(\gamma, \delta), \tag{3.9a}$$

$$\gamma = \cos(\text{lat}^{(i)}) * \sin(\text{lat}^{(i+1)}) - \sin(\text{lat}^{(i)}) * \cos(\text{lat}^{(i+1)}) * \cos(\text{lon}^{(i+1)} - \text{lon}^{(i)}), \tag{3.9b}$$

$$\delta = \sin(\text{lon}^{(i+1)} - \text{lon}^{(i)}) * \cos(\text{lat}^{(i+1)}). \tag{3.9c}$$

Following the above preprocessing steps, GPS point $\mathbf{p}^{(i)}$ is associated with motion feature vector $\mathbf{x}^{(i)} = \{d^{(i)}, v^{(i)}, a^{(i)}, k^{(i)}, b^{(i)}\} \in \mathbb{R}^5$. For every $\mathbf{p} \in \mathbf{T}$ marked as missing, all motion features that involve it in their computation are marked as

[‡]In the transportation literature, trajectories are typically considered "densely-sampled" when GPS points are collected every 1–5 seconds on average [23].

missing and set to zero in the corresponding matrices $\mathbf{M}$ and $\mathbf{X}$, respectively. Each motion feature is then linearly scaled to $[-1, 1]$ based on observed values.

We note that, when GPS points are consecutively missing, their associated motion features will also be consecutively missing. In addition, we clarify that relative distance and bearing are selected for the purpose of missing GPS point estimation. The remaining motion features are empirically selected following established travel mode identification literature [1, 44], as learning from relative distance alone can be misleading when sampling intervals are not consistent.

As a final remark, the proposed framework focuses on imputing densely sampled trajectories with missing GPS points. For datasets where trajectories are not incomplete but rather sparsely sampled (e.g., every 15 or 30 seconds), one may desire to artificially lower the mode sampling interval by inserting placeholders between observed GPS points and performing motion feature imputation. However, the route uncertainty between originally observed GPS points may be too high for motion feature reconstruction be accurate. We leave the investigation of how to extend our proposed framework for the above use case as future work.

Model Configuration G leverages 6 MGTConv blocks. Within the i-th block, convolution layers use a kernel size $k = 3$ with 256 channels and exponentially increasing dilation rates $d = 2^i$, $i \in \{0, \ldots, 5\}$. Each of G's two outputs is obtained by applying a standard temporal convolution with 1 filter and unit stride. The dropout probability is set to $p_{\text{drop}} = 0.5$ and maintained at test time for MC dropout sampling, from which $S = 100$ samples are obtained. For L_G, we tested $\lambda \in \{0.1, \ldots, 1.0\}$ and ultimately set $\lambda = 0.3$. D has 4 MGTConv blocks with 256 channels, $k = 5$, and stride $s = 2$, corresponding to window width $W_D = 61$; the output layer applies a standard temporal convolution with

1 filter and unit stride. All convolutions within UI-GAN are spectral-normalized [90], while G and D are trained in a $1:1$ alternating fashion using the default Adam optimizer with learning rates 0.0001 and 0.0004. UI-GAN is trained with a batch size of 512 for about 200 epochs, using early stopping conditioned on the validation set. The final model was developed using TensorFlow 2.3.0, has 3.9M parameters, and takes 0.28s per batch at training time, with G taking 0.6s per MC sample per sequence at inference time. All hyperparameters were empirically selected via trial-and-error based on the mean absolute error on the validation set. Our ablation studies in Table 3.1 (bottom half) and Figure 3.4 (right subplot) also provide guidelines into hyperparameter selection for the proposed UI-GAN.

Baselines and Evaluation Metrics Since our dataset contains naturally incomplete trajectories resulting in incomplete time series of motion features, we do not include baselines that cannot handle originally missing data [22, 10] or are ill-suited for time series [18, 19]. Therefore, UI-GAN is evaluated against (1) **mean** and (2) **forward** (last value) imputation, (3) **k-Nearest Neighbors (KNN)** [11], (4) **Matrix Factorization (MF)** [12], (5) **Multiple Imputation using Chained Equations (MICE)** [13], (6) **MissForest** [14], (7) **BRITS** [16], and (8) **E^{2}GAN** [21]. BRITS and E^{2}GAN were selected because they are the state-of-the-art in multivariate time series imputation; while both leverage recurrent neural networks, the proposed UI-GAN is instead based on CNNs for their computational efficiency and demonstrated success with time series data [88]. While E^{2}GAN resembles our proposed framework in that both employ a GAN formulation, E^{2}GAN does not account for predictive uncertainty. Both BRITS and E^{2}GAN were implemented using their publicly available code, while the rest were provided by the `fancyimpute` and `missingpy` libraries for Python. For k-NN, we found $k = 10$ to perform best.

In the following experiments, we report the MAE calculated over all imputed and artificially dropped (but originally observed) motion features in the normalized test set.

3.3.2 Results

Motion Feature Imputation To evaluate UI-GAN against the baselines on motion feature imputation, we report the MAE after artificially dropping 20%, 40%, 60%, and 80% of observed GPS points (and their corresponding motion features) by sampling from a Bernoulli distribution. Our experimental results, summarized in Table 3.1, show that UI-GAN consistently and significantly outperformed all evaluated methods for all percentages of missing GPS points. In the case of 80% missing data, UI-GAN attained an 18% improvement over E^2GAN, which was the best performing baseline. The subpar performance of BRITS may be in part due to the lack of class labels in GAIA; this method was designed to jointly impute and classify time series based on a downstream task. Among the machine learning baselines, MissForest achieved the lowest MAE on average, followed by MICE for all but 80% of missing data.

Ablation Study To empirically validate the contribution of dropout variational inference, MGTConv blocks, and window-level D to the imputation performance of UI-GAN, we repeated our experiments by varying $W_D \in \{13, 29, 61, 125\}$, as well as by removing either component while maintaining the other. Note that the above range of window widths resulted from setting $s = 2$, $k = 5$ for D and varying the number of layers in $\{2, 3, 4, 5\}$. As shown in Table 3.1, either using a deterministic G or non-gated convolutions resulted in considerable increase in MAE when more than 40% of data were missing. This is not surprising, as a deterministic G cannot account for the growing uncertainty caused by having fewer

Table 3.1 : Imputation results (MAE, lower is better) for percentages of artificially dropped GPS points.

	Missing Data			
Method	20%	40%	60%	80%
Mean	0.198	0.200	0.201	0.202
Forward	0.101	0.108	0.120	0.146
KNN	0.107	0.113	0.126	0.177
MF	0.141	0.146	0.148	0.156
MICE	0.087	0.095	0.117	0.187
MissForest	0.073	0.080	0.092	0.124
BRITS	0.079	0.086	0.098	0.135
E^2GAN	0.057	0.066	0.085	0.120
Non-Bayesian G	0.048	0.058	0.079	0.122
Non-gated Conv.	0.047	0.061	0.088	0.134
$D_W = 13$	**0.046**	0.057	0.074	0.114
$D_W = 29$	0.048	**0.056**	0.076	0.111
$D_W = 61$	0.049	0.057	**0.073**	**0.100**
$D_W = 125$	0.050	0.058	0.075	0.108

observations available during training, while standard convolutions are applied to more features with invalid (zero) values. Moreover, smaller discriminator windows D_W performed better for sequences with lower percentages of missing data, while $D_W = 61$ worked better for higher percentages. This supports our intuition towards designing a window-level rather than one-size-fits-all discriminator.

Uncertainty-filtered Imputation The ability to selectively impute values based on some measure of confidence can be highly beneficial, as inaccurate estimations may adversely affect subsequent data analysis and downstream applica-

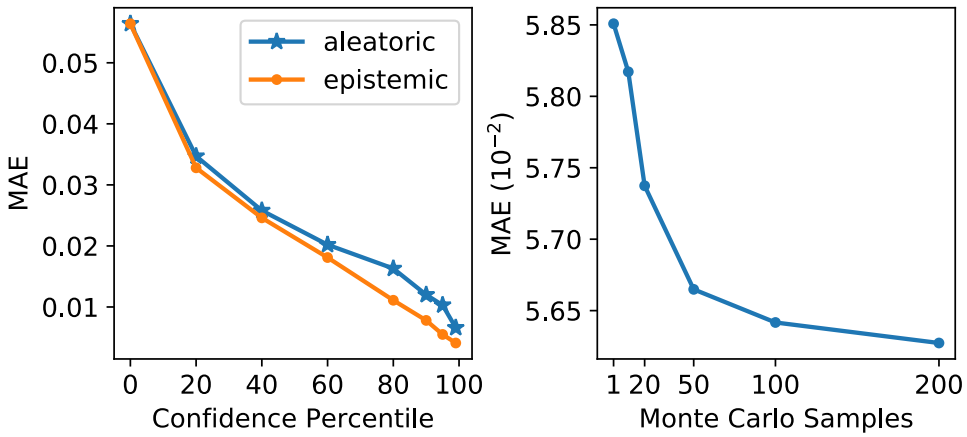

Figure 3.4 : MAE when only imputing motion feature timesteps that exceed confidence thresholds (left). Sensitivity of MAE to number of MC Samples S (right). In both cases, 40% of observed GPS points have been purposefully discarded.

tions. Defining confidence as the negative aleatoric or epistemic uncertainty, the left subplot of Figure 3.4 shows that both were effective predictors of MAE. Finally, the right subplot of Figure 3.4 shows the sensitivity of MAE to the number of MC samples S. Since MAE demonstrated marginal improvement for $S > 100$, we selected this value throughout our experiments.

GPS Point Estimation While UI-GAN imputes motion features, the ultimate aim is to estimate raw GPS points. Figure 3.5 visualizes our results for exemplar trajectory segments exhibiting challenging patterns in change of direction. Observed GPS points are in black, while dropped and estimated ones are in green and red, respectively. In cases where GPS points were consecutively missing, each was estimated using the last imputed point as the origin in eqs. (3.7a – 3.7d). Our qualitative results suggest that the proposed approach can generate high-fidelity GPS points. We note that the median distance between imputed and artificially dropped GPS points in the test set was 5.6 meters, or 18.4 feet.

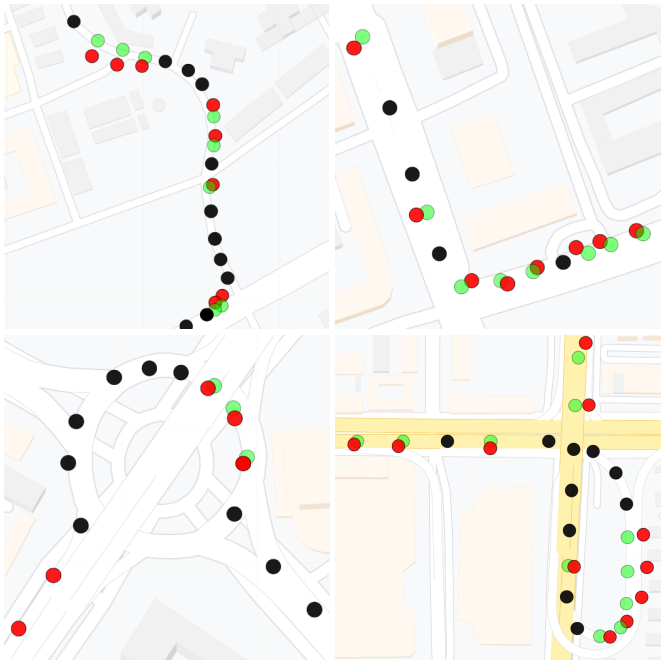

Figure 3.5 : Examples of incomplete trajectories imputed by UI-GAN. Observed, dropped, and estimated GPS points are in black, green, and red. Note that UI-GAN uses no underlying map information.

3.4 Summary

In this chapter, we viewed GPS trajectory imputation from a novel perspective to address the shortcomings of related work in learning from incomplete trajectories and capturing imputation uncertainty. Specifically, we first extracted incomplete sequences of motion features measuring magnitude and angle of displacement; we then trained an Uncertainty-aware Imputation GAN (UI-GAN) to impute them and finally used the recovered motion features to estimate missing GPS points. We also conducted an ablation study to empirically validate the necessity of UI-GAN's components, showing that its Bayesian generator effectively captured uncertainty by producing lower MAE for motion features with lower aleatoric and epistemic uncertainties.

Chapter 4

Supervised Trajectory Segmentation by Transportation Mode

Since GPS trajectories may contain multiple transportation modes, it is standard practice to first segment them by transportation mode before feeding the resulting segments to a classifier. In this chapter, we leverage recent developments in semantic image segmentation to propose a supervised deep learning approach for extraction of same-transportation-mode segments from GPS trajectories of arbitrary length. We achieve 71.7% mIoU on Microsoft's Geolife dataset [1, 34] without significantly over- or under-predicting transportation mode change points. Compared to the best performing baseline, the proposed segmentation model attains a nearly 2× reduction in mean absolute error between true and predicted change points. We also introduce (1) a simple yet highly effective data augmentation technique tailored to the above method, and (2) a *majority-vote* post-processing step to smoothen the predicted segments, thereby minimizing false timestep-level predictions. Both techniques are shown to significantly improve segmentation performance.

The rest of this chapter is structured as follows. Section 4.1 formulates

the problem of GPS trajectory segmentation. Section 4.2 describes the proposed trajectory segmentation framework, including our data preprocessing steps, as well as the proposed augmentation and post-processing techniques. Section 4.3 then presents our simulation setup and analyzes our experimental results. Section 2.3 examines the literature on semantic image segmentation, which inspired the proposed trajectory segmentation approach. Finally, Section 4.4 concludes this chapter.

4.1 Problem Formulation

In this section, we formulate the problem of partitioning GPS trajectories such that each segment involves exactly one transportation mode. This is often a necessary preprocessing step before transportation mode identification [56, 23]. For the definitions of GPS point $\mathbf{p}$, GPS trajectory $\mathbf{T}$, and motion feature sequence $\mathbf{X}$, we refer the reader to Definitions 3.1, 3.2, and 3.3 in Section 3.1.1, respectively.

Definition 4.1 (GPS Trajectory Segment). Definitions 3.1 and 3.2 mean that trajectory $\mathbf{T}$ may involve the use of multiple transportation modes. As such, $\mathbf{T}$ can be partitioned into $M > 0$ consecutive non-overlapping segments $\{\mathbf{G}^{(0)}, \mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(M-1)}\}$ such that each segment contains exactly one transportation mode. Defined formally, segment $\mathbf{G}^{(k)}$ must satisfy the condition that $\text{mode}^{(i)} = \text{mode}^{(j)} \forall (\mathbf{p}^{(i)}, \mathbf{p}^{(j)}) \in \mathbf{G}^{(k)}$.

Definition 4.2 (Motion Feature Sequence Segment). Following Definitions 3.3 and 4.1, we convert single-transportation-mode trajectory segment $\mathbf{G}^{(k)}$ into a motion feature sequence segment $\mathbf{F}^{(k)}$. The extraction of single-transportation-mode trajectory segments (and associated motion feature sequence segments) is required for our proposed shuffling-based data augmentation technique, which will be presented in Section 4.2.2.

Problem 4.1 (Transportation Mode Segmentation). Given K target transportation mode classes and GPS trajectory $\mathbf{T}$ of length $\|\mathbf{T}\|$ preprocessed into a multivariate time series of N motion features $\mathbf{X} \in \mathbb{R}^{\|\mathbf{T}\| \times N}$, predict the transportation modes $\mathbf{y} \in \{0, \ldots, K-1\}^{\|\mathbf{T}\|}$ for each timestep t, where $0 \leq t \leq \|\mathbf{T}\|-1$.

4.2 Proposed Framework

This section starts by detailing our data preprocessing pipeline, including feature extraction, outlier deletion, feature standardization, and appropriate resegmentation or truncation of single-transportation-mode segments for use with the proposed trajectory segmentation model. It then introduces the architecture of the proposed trajectory segmentation model, as well as two techniques that we devise in order to improve its performance: a preprocessing, shuffling-based data augmentation technique, and a *majority-vote* post-processing step.

4.2.1 Data Preprocessing

We take the Geolife dataset [1, 34] by Microsoft Research Asia as an illustrative example of real-world GPS trajectories in practice. It consists of both labeled and unlabeled GPS trajectories, many of which involve multiple transportation modes. For this reason, we first perform trajectory segmentation at preprocessing time, identifying a new segment whenever there is a change in transportation mode or the elapsed time between a pair of consecutive GPS points is larger than twenty minutes. This threshold was introduced by Zheng *et al.* [1] and has since been used by multiple studies [34, 3].

Following trajectory segmentation, we convert the available raw GPS data to motion feature sequences. Concretely, we begin by calculating the relative geodesic distance $d^{(i)}$ between $\mathbf{p}^{(i)}$ and $\mathbf{p}^{(i+1)}$ (in meters) using the established Vincenty formula [92]; we also obtain the elapsed time $t^{(i+1)} - t^{(i)}$ (in seconds).

We then estimate higher-order derivatives of distance, including velocity $v^{(i)}$, acceleration $a^{(i)}$, and jerk $k^{(i)}$:

$$d^{(i)} = \text{Geodesic}(\text{lat}^{(i)}, \text{lon}^{(i)}, \text{lat}^{(i+1)}, \text{lon}^{(i+1)}), \tag{4.1}$$

$$v^{(i)} = \frac{d^{(i)}}{t^{(i+1)} - t^{(i)}}, \tag{4.2}$$

$$a^{(i)} = \frac{v^{(i+1)} - v^{(i)}}{t^{(i+1)} - t^{(i)}}, \tag{4.3}$$

$$j^{(i)} = \frac{a^{(i+1)} - a^{(i)}}{t^{(i+1)} - t^{(i)}}. \tag{4.4}$$

We follow previous transportation mode identification literature [3, 46] as well as our own preliminary experiments in selecting velocity, acceleration, and jerk to train the proposed transportation mode segmentation model. We then remove GPS points deemed unrealistic based on a set of heuristics [3]. Concretely, we eliminate GPS points satisfying one or more of the following conditions:

- Either of its coordinates is invalid, i.e., $\text{lat}^{(i)} \notin [-90, 90]$ or $\text{lon}^{(i)} \notin [-180, 180]$;
- Its timestamp is greater or equal than its successor's, i.e., $t^{(i)} \geq t^{(i+1)}$;
- Its velocity or acceleration is unreasonably high [3] based on its associated travel mode.

Next, we discard any GPS point whose velocity, acceleration, or jerk exceeds the 99^{th} percentile. We also remove GPS points if their acceleration or jerk do not surpass the 1^{st} percentile. Observing that the distribution of velocities is highly skewed towards near-zero values, we apply a cubic root transformation, before finally standardizing all motion features to zero mean and unit variance.

Given that the above motion features are calculated for each GPS point, it is important to address the common problem of missing data, indicated by

abnormally long distances between pairs of consecutive GPS points. While several data imputation techniques could be applied to rectify this issue, including the one proposed in Chapter 3, we note that reasonable amounts of missing data might not be detrimental to transportation mode identification. Intuitively, suppose that a user enters a tunnel where the GPS signal is blocked until the user finally exits, resulting in a pair of GPS points separated by a 250-meter gap; it is expected that the velocity of the first point with regards to the second would not be overly anomalous, since their relative distance would increase proportionately to the elapsed time. It is partly to address such cases that we do not employ relative distance as a motion feature. We also note that, although the trajectory imputation framework proposed in Chapter 3 could be leveraged to estimate missing GPS points prior to motion feature calculation, this would condition the performance of the proposed trajectory segmentation framework on that of the imputation framework, thereby affecting the evaluation of the former.

4.2.2 Trajectory Segmentation Model

This section first describes the formatting of the preprocessed GPS data for use with the proposed trajectory segmentation model, as well as the simple shuffling-based data augmentation technique that we devised to improve segmentation performance. Next, it presents the architecture of our trajectory segmentation model and analyzes its main components. Finally, this section concludes with the post-processing step that we apply to the segmentation model's timestep-level predictions in order to refine the extracted segments.

Input Data and Augmentation

As will be discussed in the sequel, the proposed trajectory segmentation model leverages a supervised deep convolutional architecture for timestep-level classification. To learn segmentation boundaries between transportation modes,

it requires samples containing sequences of multiple single-mode segments. Therefore, after extracting such segments from raw GPS trajectories according to Section 4.2.1, we first concatenate them into a single sequence before resegmenting them into non-overlapping chunks of L timesteps, producing samples $\mathbf{X} \in \mathbb{R}^{M \times N}$ where $N = 3$ is the number of features. Since the segmentation model outputs timestep-level predictions, the class labels for each segment are one-hot encoded and repeated according to the segment's length, resulting in labels $\mathbf{y}$ of shape (M, K), where $K = 5$ is the number of classes.

We hypothesize that there is an inherent limitation to the standard practice of merely shuffling the training set during training, stemming from the nature and size of our labeled dataset, as well as the segmentation task itself. Specifically, the model could be biased towards learning transitions (i.e., segmentation boundaries) between transportation modes occurring more frequently in the labeled dataset. To mitigate this issue, we devise the following shuffling-based data augmentation technique: after every training epoch, we shuffle the single-transportation-mode segments within the training set and resegment them into new non-overlapping chunks $\mathbf{X} \in \mathbb{R}^{M \times N}$ together with their corresponding labels $\mathbf{y}$. In our experiments, this simple modification was shown to significantly improve segmentation performance.

Model Architecture

Learning to detect high-precision boundaries between arbitrary-length segments, each involving a single transportation mode, requires a segmentation model that can perform dense, timestep-level classification. Following seminal work originally applied to images [29, 33, 30], as well as a recent related approach to segmenting time series data [2], we design a fully convolutional encoder-decoder architecture for trajectory segmentation. Figure 4.1 offers a high-level view of the

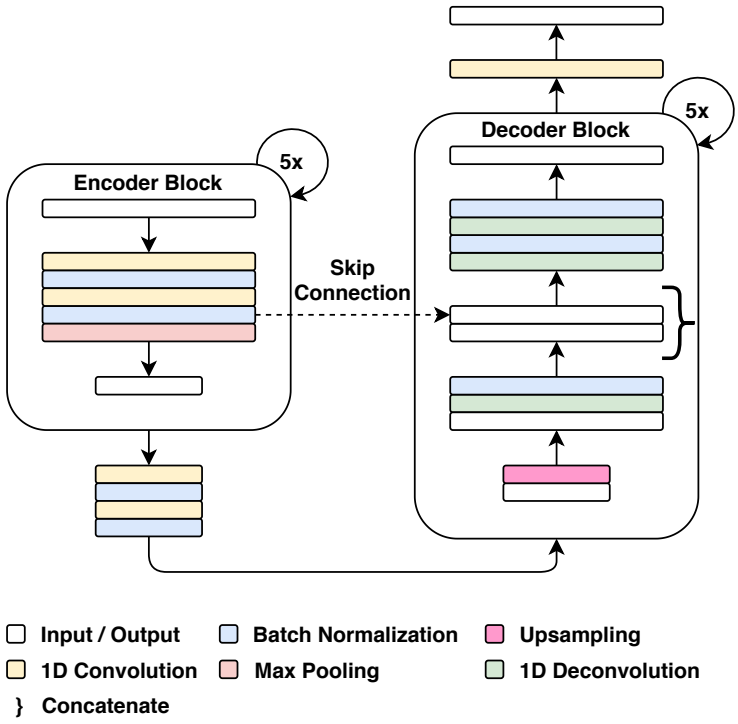

Figure 4.1 : The proposed trajectory segmentation model is built by stacking five encoder and five decoder blocks. At the i-th encoder block, convolutions use an exponential dilation rate of 2^i to increase the network's receptive field as the input dimensionality is downsampled. This information is passed to the decoder side via skip connections and merged with the upsampled feature maps. The output of the last decoder block is fed to a standard softmax-activated convolution layer with K filters of unit length.

proposed architecture, inspired by U-net [33].

For the most part, the encoder sub-network resembles a standard contracting CNN: each encoder block halves its input dimensions while doubling the number of channels. To do so, it applies two zero-padded 1D convolutions and a max pooling operation for downsampling. Each convolution layer is batch-normalized to accelerate training convergence and reduce overfitting.

Where the encoder side mainly differs from traditional CNN architectures is the use of *dilated* convolutions [30, 31]. Dilation allows convolutions to skip input locations with step (or rate) $d \in \mathbb{Z}^+$, thus requiring fewer layers to increase the network's receptive field compared to standard convolutions. With fewer layers implying fewer trainable parameters, such practice further reduces the risk of overfitting. The two convolution layers at the i-th encoder block use a dilation rate of 2^i; for instance, the first encoder block corresponds to $d = 2^0 = 1$, which reduces to standard convolution.

The decoder sub-network attempts to gradually recover the resolution of the original input, aided by the incorporation of encoder-downsampled information at each scale. This is achieved via the use of skip connections from each encoder block to its same-scale decoder block. Specifically, each decoder block first applies a nearest-neighbor upsampling operation to its input, followed by a 1D deconvolution and batch normalization, then merges the resulting feature map with the one from the encoder side by concatenating them along the channel axis. Next, the decoder block applies two more 1D deconvolutions, each followed by a batch normalization operation. Finally, a point-wise, standard 1D convolution with K scalar filters is applied to the feature map of the final decoder block to produce K scores for each timestep of the input. Note that deconvolutions at the decoder side are not dilated; while this was implemented at the encoder size

to expand the convolutions' receptive fields, this information is readily available to the decoder blocks due to the presence of the corresponding skip connections.

To train deep semantic segmentation models, most image-based studies optimize the pixel-level cross entropy loss [28, 33, 67, 70]. Depending on the properties of the data at hand, however, different objective functions may be more suitable. For instance, to handle class imbalance within samples, [2] minimizes the generalized dice loss; a viable alternative is to optimize the pixel-level cross entropy loss with sample re-weighting. As discussed in Section 4.2.2, we train the proposed segmentation model on samples consisting of concatenated variable-length trajectory segments. Therefore, to avoid producing a model biased towards longer segments, we optimize the generalized dice loss, defined as:

$$L_{\text{dice}}(\mathbf{Y}, \hat{\mathbf{Y}}) = 1 - \frac{2}{K} \frac{\sum_{k}^{K} \sum_{l}^{L} \mathbf{Y}_{k,l} \hat{\mathbf{Y}}_{k,l}}{\sum_{k}^{K} \sum_{l}^{L} \mathbf{Y}_{k,l} + \hat{\mathbf{Y}}_{k,l}}, \tag{4.5}$$

where $\mathbf{Y}$, $\hat{\mathbf{Y}}$ are matrices containing predicted and ground-truth labels, respectively, and K is the number of classes. Indeed, during our preliminary experiments, we empirically found that minimizing the above loss rather than the timestep-level cross entropy loss produced significantly better results.

Post-processing

After training the segmentation model, one could obtain single-transportation-mode segments by simply performing inference on the test set and concatenating consecutive timesteps classified into the same class. However, timestep-level errors could result in either fragmented or noisy segments, as empirically supported by our experiments in Section 4.3.2. Such segments could subsequently complicate transportation mode identification.

To reduce timestep-level errors and smoothen the extracted segments, we propose a simple *majority-vote* post-processing step, which is applied in an over-

lapping sliding window manner to the segmentation model's flattened output following inference. Moving from left to right, the predicted class at timestep t is set to the modal value of the $W/2$ classes preceding and the $W/2$ classes following timestep t, where hyperparameter W is defined as the post-processing window width. When there are fewer than $W/2$ timesteps before or after t, as may be the case near the beginning or ending of the model's flattened output, we simply calculate the modal value using as many timesteps as available.

4.3 Experiments

In this section, we first introduce the dataset used to evaluate the proposed trajectory segmentation model and the hardware on which we performed our experiments. Then, we provide a description of our segmentation model configuration, as well as the established baselines and performance metrics that we use to evaluate it. After analyzing our experimental findings, we finally assess the necessity of the segmentation model's main components through an ablation study.

4.3.1 Dataset and Simulation Setup

Our experiments were conducted on the same computing server as described in Section . We report the averaged results of five independent executions for all evaluation metrics.

Dataset

We evaluate the proposed framework on the real-world, openly available Geolife dataset [1, 34], which has been established as a benchmark in transportation research. Geolife encompasses $18,670$ GPS trajectories obtained from 182 individuals during five years, totaling nearly 25 million GPS points and 1.3 million kilometers. Most trajectories were sampled every $1-5$ seconds in Beijing,

China. Only 69 users partially annotated their trajectories by transportation mode. For model evaluation purposes, only labeled trajectories are used in our experiments. We note that it would be possible to also include unlabeled trajectories, for instance, by having the trained segmentation model perform inference on them and then incorporating the extracted segments in training of the clustering model. However, the selected standard evaluation metrics demand ground-truth labels, and hence may only be applied to labeled samples. Learning from unlabeled trajectories is also possible via semi-supervised or unsupervised training of the segmentation model. To the best of our knowledge, such methods have not yet been explored in the context of GPS trajectory segmentation, and are left for future work.

Regarding the transportation modes included in Geolife, we follow Zheng *et al.* [1, 34] in treating taxis and private cars as a single class, *car*; we do the same for trains and subways, which are merged into the *train* class. We follow standard practice [1, 34, 3] and only keep the transportation mode classes sufficiently represented in the dataset, i.e., *walk*, *bike*, *bus*, *car*, and *train*. For the segmentation model, the labeled trajectories are then divided into training, validation, and test sets using an 80/10/10 split. The resulting trajectories are then preprocessed into sequences of motion features as described in Section 4.2.1 and prepared for the segmentation model as per Section 4.2.2.

Model Configuration

Our trajectory segmentation model first downsamples its input via five encoder blocks, whose convolution layers have 16, 32, 64, 128, and 256 filters, respectively; their dilation rates are 1, 2, 4, 8, and 16. Each convolution layer uses *same* padding and a kernel with length 3, and is connected to a batch normalization layer. The decoder leverages five decoder blocks to gradually restore the

original dimensionality through nearest-neighbor upsampling and deconvolution layers with 256, 128, 64, 32, and 16 filters. All deconvolution layers use *same* padding and a kernel of length 3, and are connected to a batch normalization layer. We apply the Rectified Linear Unit (ReLU) activation function to all convolutions and deconvolutions, except for the convolution having unit kernel length that follows the final decoder layer, which has 5 filters and produces timestep-level predictions using the softmax function. The segmentation model is trained for 400 epochs using the Adam optimizer with learning rate 10^{-4} and default hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$; this was empirically found to produce the best results on the validation set. Regarding the sample length L, although $L \in \{128, 256, 512\}$ worked reasonably well in preliminary experiments, we set $L = 2048$ for improved computational efficiency during training. For the selected value of L, the optimal post-processing window width W was determined to be 256 using the same validation set, following the experimental results visualized in Figure 4.2.

Baselines

To assess the accuracy of the proposed trajectory segmentation model, we evaluate it against the four most prominent segmentation methods among the relatively few used in the GPS-based transportation mode identification literature:

1. **Uniform segmentation.** A naive approach to trajectory segmentation is to extract fixed-size, non-overlapping chunks of timesteps from each trajectory. Dabiri and Heaslip [23] use a window of 200 points, which they found to be the median length of the labeled GPS trajectories in Geolife.

2. **Heuristics-based change point detection.** Following the intuition that walking must precede any change of transportation mode, Zheng *et al.* [1] pro-

pose a four-step method that divides trajectories into alternating *walk* and *non-walk* segments, based on distance, velocity, and acceleration thresholds. Xiao *et al.* [26] additionally identify gaps caused by GPS signal interruption, extracting change points at the boundaries of both *walk* and *gap* segments. Note that we omit references [24, 25, 27] from our experiments since they are largely hyperparameter variations of Zheng *et al.* [1]; in our preliminary experiments, they only attained marginal improvements.

3. **Optimization-based change point detection.** The two-step trajectory segmentation method proposed by Dabiri *et al.* [3] first applies uniform segmentation to each trajectory, and then converts the obtained segments into multivariate time series of velocity and acceleration features before feeding them to an optimization-based model. The latter attempts to produce subsegments such that the homogeneity within each subsegment is maximized, while the number of change points is penalized by a hyperparameter-controlled linear function.

We implement each baseline as per the description in its cited study, including any hyperparameters or data preprocessing steps. For fair comparison, all evaluated methods are assessed on the same test set, which comprises 10% of the labeled data. We note that, while other semantic image segmentation frameworks like DeepLab [31] may also be adapted to transportation mode segmentation, our aim is to evaluate the proposed framework against existing work designed specifically for transportation mode segmentation of GPS trajectories.

Evaluation Metrics

The proposed semantic segmentation model labels each timestep by transportation mode. On the contrary, the above baselines detect transportation mode change points, i.e., the discrete timesteps where the transportation mode changes,

without knowledge of the specific mode corresponding to each segment. Since these methods are label-agnostic, evaluation metrics commonly used in the semantic segmentation literature, such as the dice coefficient or the mean Intersection over Union, are not applicable to them. To allow for direct comparison with our model, we simply scan its output from left to right, extracting change points as the timesteps where the predicted transportation mode changes. We then evaluate the above baselines using the following evaluation metrics:

- **Mean Absolute Error (MAE)**. As the number of predicted and ground-truth change points may differ, we first identify for each predicted change point $\hat{\mathbf{c}}_i \in \hat{\mathbf{c}}$ its nearest ground-truth $\mathbf{c}_j \in \mathbf{c}$. As most trajectories in Geolife have been sampled at a rate of $1-5$ seconds, we preserve generality by measuring the distance between two change points by the number of timesteps separating them. Then, given $\|\hat{\mathbf{c}}\|$ predicted change points $\hat{\mathbf{c}}$ and their nearest ground-truth change points $\mathbf{c}'$, the MAE is calculated as:

$$\text{MAE} = \frac{1}{\|\hat{\mathbf{c}}\|} \sum_{i=0}^{\|\hat{\mathbf{c}}\|-1} |\mathbf{c}'_i - \hat{\mathbf{c}}_i|. \tag{4.6}$$

  We select this metric as it estimates the average magnitude of the error without considering direction. The underlying assumption is that the number of timesteps between $\mathbf{c}'_i$ and $\hat{\mathbf{c}}_i$ matters more than whether $\hat{\mathbf{c}}_i$ precedes or comes after $\mathbf{c}'_i$. Please note that the definition of MAE here is different from the one in Chapter 3.

- **Prediction Ratio (PR)**. The PR is computed as:

$$\text{PR} = \frac{\|\hat{\mathbf{c}}\|}{\|\mathbf{c}\|}. \tag{4.7}$$

  When PR < 1 or PR > 1, the segmentation model is said to under- or over-predict change points. Either case could complicate transportation mode identification, even if the MAE is low: too few change points could mean

Table 4.1 : Trajectory segmentation evaluation results in terms of MAE, PR, and mIoU. Lower MAE, PR ≈ 1, and higher mIoU values are better.

Segmentation Method	MAE	PR	mIoU
Dabiri and Heaslip [23]	728.4	3.12	-
Dabiri *et al.* [3]	526.1	1.23	-
Xiao *et al.* [26]	287.7	2.34	-
Zheng *et al.* [1]	234.5	1.91	-
Proposed	**119.4**	**1.02**	**0.717**

noisy segments with more than one transportation mode, while too many would result in short segments that might be harder to classify. Therefore, a segmentation model should strive for values of PR ≈ 1.

- **Mean Intersection over Union (mIoU)**. The mIoU metric, typically used in evaluating semantic image segmentation models [29, 31], quantifies the overlap between predicted and ground-truth segments as:

$$\text{mIoU} = \frac{1}{K} \sum_{i} \frac{\mathbf{N}_{i,i}}{\sum_j \mathbf{N}_{i,j} + \sum_j \mathbf{N}_{j,i} - \mathbf{N}_{i,i}}, \tag{4.8}$$

where K is the number of classes, $\mathbf{N}_{i,j}$ is the number of ground-truth timesteps in class i predicted as belonging to class j, and $\sum_j \mathbf{N}_{i,j}$ is the total number of ground-truth timesteps in class i.

4.3.2 Results

Our experimental results are presented in Table 4.1. The proposed trajectory segmentation model consistently outperformed all evaluated baselines,

demonstrating the lowest MAE of 119.4 while at the same time not significantly over- or under-predicting change points.

Although each of the selected baselines has its own merit, they are not without limitations. Uniform segmentation [23] offers simplicity at the expense of producing noisy segments likely to contain multiple transportation modes. Indeed, in our experiments, uniform segmentation attained both the highest MAE and PR among all methods. Guided by domain-specific knowledge, the heuristics-based segmentation baselines [1, 26] both produced much lower MAEs, although they still predicted approximately twice the number of change points compared to the ground truth. This might be due to such methods not always being able to account for unexpected traffic conditions [1].

Scoring the second highest MAE of 526.1, despite having the second best PR, the sub-par performance of optimization-based segmentation [3] is not surprising: this method is applied to each fixed-size segment individually, thereby failing to consider the entire data distribution. However, the assumption of samples being independent and identically distributed may not hold for trajectory data collected from numerous users with likely different road behaviors, such as the data at hand.

Ablation Study and Hyperparameter Tuning

We posit that the promising performance of our trajectory segmentation model depends on its following distinguishing components: (1) shuffling-based data augmentation, (2) dilated encoder convolutions, (3) encoder-decoder skip connections, and (4) segmentation post-processing. To quantify the influence of these components on the selected evaluation metrics, we iteratively remove one of them and train our model from scratch while maintaining the rest. As such, the corresponding ablations are defined as follows:

Table 4.2 : Segmentation model ablation results in terms of MAE, PR, and mIoU.

Ablation	Post-processing	MAE	PR	mIoU
1	✗	242.4	9.94	0.557
	✓	186.7	1.23	0.591
2	✗	233.6	4.45	0.694
	✓	167.7	1.36	0.703
3	✗	203.0	2.06	0.675
	✓	149.7	1.08	0.682
Proposed	✗	225.9	2.82	0.713
	✓	**119.4**	**1.02**	**0.717**

1. **Shuffling-based data augmentation.** Once the training set for the segmentation model is generated as per Section 2.2.2, the samples are merely shuffled into new batches on every epoch following standard neural network training practice.

2. **Dilated encoder convolutions.** The exponentially dilated convolutions at each encoder block are replaced with standard convolutions.

3. **Encoder-decoder skip connections.** The skip connections from encoder blocks to their counterparts at the decoder side are removed.

Table 4.2 shows the experimental results of the ablation study, with each ablation being evaluated with and without the post-processing step introduced in Section 4.2.2. It is evident that not including the proposed shuffling-based data augmentation scheme, i.e., ablation 1, caused the largest degradation in segmentation performance. In addition, ablations 2 and 3 both demonstrated worse

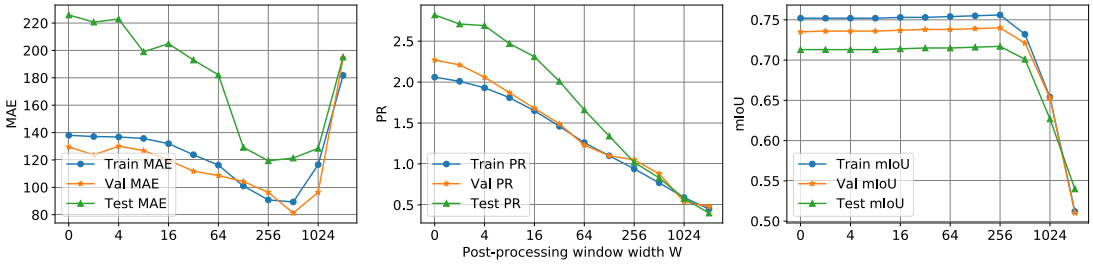

Figure 4.2 : Sensitivity of the proposed segmentation model's MAE, PR, and mIoU to a wide range of post-processing window widths W. Lower MAE, PR ≈ 1, and higher mIoU values are better.

results than the proposed method, especially without the proposed *majority-vote* post-processing step. While ablation 3 achieved somewhat comparable results to our segmentation model when the post-processing step was applied to both, the former required approximately 150 more training epochs until convergence.

Crucially, the proposed post-processing step seems to considerably improve segmentation performance for all ablations, provided that its window width W is tuned accordingly. Figure 4.2 plots the values of the selected evaluation metrics on the training, validation, and test sets as W ranges from 0 to 2048. It is evident that $W = 256$ produced the best combination of scores for the selected evaluation metrics on the validation set; this might be linked to the median segment length, which is 291 following our data preprocessing steps in Section 4.2.1. While $W < 256$ had little effect on mIoU, it resulted in much higher MAE and change point over-prediction. On the other hand, $W > 256$ did not significantly affect the MAE, except when $W = 2048$. Nonetheless, it caused a significant reduction in mIoU, as well as under-prediction of change points. We note that larger values for W may include less related predictions in the voting scheme; in this direction, we leave the exploration of *weighted majority-vote* schemes for future work.

4.4 Summary

Learning to detect transportation modes within segments of users' often unlabeled GPS trajectories is essential for travel demand analysis, transportation management, and infrastructure design. In this chapter, we proposed a deep learning framework to address the challenging task of GPS-based trajectory segmentation by transportation mode. Following extraction of motion features from raw GPS data, we first trained a deep convolutional segmentation architecture for timestep-level classification, drawing inspiration from recent developments in semantic image segmentation. We then obtained single-mode segments by merging consecutive timestep-level instances of the same transportation mode in the model's output. Our results showed that the proposed trajectory segmentation approach considerably outperformed existing baselines, with our shuffling-based data augmentation and *majority-vote* post-processing techniques further refining the predicted segments to achieve 71.7% mIoU on Geolife, without over- or under-predicting transportation mode change points.

Chapter 5

Bayesian Unsupervised Trajectory Segmentation

In the previous chapter, we proposed a transportation mode segmentation framework based on supervised deep learning. With GPS trajectories typically not being labeled by transportation mode due to privacy concerns or lack of motivation, it is imperative to design a transportation mode segmentation framework that is not dependent on label availability. Drawing inspiration from recent developments in semantic segmentation, deep clustering, and Bayesian deep learning, in this chapter we reframe GPS trajectory segmentation as timestep-level transportation mode identification; the latter follows our methodology in Chapter 4. As such, we design a channel-calibrated Bayesian Temporal Convolutional Network (BTCN) for unsupervised, uncertainty-aware GPS trajectory segmentation. BTCN extends standard TCNs, recently proposed as a sequence modelling alternative to recurrent neural networks [88], with (1) Squeeze-and-Excitation (SE) blocks [93] to encourage learning interdependencies between channels, and (2) Monte Carlo (MC) dropout sampling as an approximation of variational inference [85] to not only capture predictive uncertainty but also use it to refine

predictions. In our experiments on Microsoft's Geolife dataset [1, 34], BTCN achieved 65.8% timestep-level accuracy without using any labels, outperforming established baselines as well as its non-Bayesian variant.

The rest of this chapter is organized as follows. Section 5.1 introduces Bayesian deep learning. Section 5.2 details the architecture of BTCN, including how it performs dropout variational inference and quantifies predictive uncertainty, as well as the unsupervised segmentation objective function that is optimized during training. Section 5.3 then presents our simulation setup and analyzes our experimental results. Section 2.5 examines the literature on deep clustering, a body of works that perform clustering using deep neural networks, which inspired the unsupervised segmentation technique used to train the proposed framework. Finally, Section 5.4 concludes this chapter. For definitions of key GPS-trajectory-related terms used in this chapter, as well as our formulation of the transportation mode segmentation problem, we refer the reader to Section 4.1. The content of this chapter is taken from a conference paper [94] that has been published in the Proceedings of the 35$^{\text{th}}$ AAAI Conference on Artificial Intelligence.

5.1 Bayesian Deep Learning

Let $\mathcal{D} = \{(\mathbf{X}^{(i)}, \mathbf{y}^{(i)})\}_{i=0}^{n-1}$ be a dataset of observations $\mathbf{X}$ and targets $\mathbf{y}$. For a standard fully connected neural network with L stacked hidden layers and parameters $\theta = \{(\mathbf{W}^{(l)}, \mathbf{b}^{(l)})\}_{l=0}^{L}$, the ReLU-activated output of the l-th hidden layer can be written as:

$$\mathbf{H}^{(l)} = \text{ReLU}(\mathbf{W}^{(l)}\mathbf{H}^{(l-1)} + \mathbf{b}^{(l)}). \tag{5.1}$$

For K-class classification, the neural network's output logits are typically activated using the softmax activation function. The model likelihood is then given

by:

$$p(\mathbf{y} = \mathbf{e}^{(j)}|\mathbf{X}, \theta) = \frac{\exp\Big(\mathbf{W}^{(L)}_{j,:}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}\Big)}{\sum_{k \in K} \exp\Big(\mathbf{W}^{(L)}_{k,:}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}\Big)}, \tag{5.2}$$

where $\mathbf{e}^{(j)}$ is a one-hot encoded vector, i.e., $\mathbf{e}^{(j)}_j = 1$ and all other elements are zeros.

In contrast, Bayesian neural networks place a prior distribution $p(\theta)$ on their weights and biases, resulting in the posterior distribution:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{\prod_{i=0}^{n-1} p(\mathbf{y}^{(i)}|\mathbf{X}^{(i)}, \theta)p(\theta)}{p(\mathcal{D})}, \tag{5.3}$$

and the following predictive distribution for new inputs $\mathbf{X}'$, $\mathbf{y}'$:

$$p(\mathbf{y}'|\mathbf{X}', \mathcal{D}) = \int_{\Theta} p(\mathbf{y}'|\mathbf{X}', \theta)p(\theta, \mathcal{D})\, d\theta. \tag{5.4}$$

However, analytical estimation of the posterior is typically intractable, as it requires integration with respect to Θ, i.e., the space of all parameters, for which a closed form often does not exist. One way to approximate the posterior is through variational inference, which optimizes the Kullback-Leibler divergence between the posterior $p(\theta|\mathcal{D})$ and a variational distribution $q_\omega(\theta)$ with parameters ω [95]. Another approach is to approximate variational inference itself by applying stochastic regularization techniques like dropout to non-Bayesian neural networks. As will be detailed in the following section, we approximate variational inference via MC dropout sampling [85].

5.2 Proposed Framework

This section first introduces the Bayesian temporal convolutional network proposed for GPS trajectory segmentation. Next, it explains how we approximate variational inference and capture uncertainty to refine model predictions. Finally, it presents the objective function that is optimized to learn the unsupervised segmentation task.

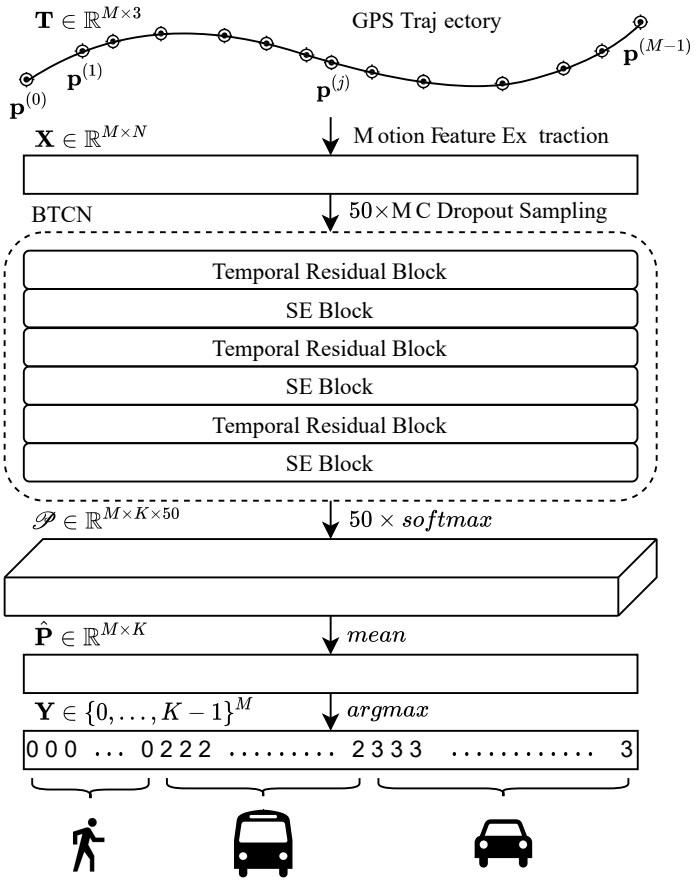

Figure 5.1 : Overview of the proposed trajectory segmentation framework. At test time, GPS trajectories are preprocessed into sequences of motion features and repeatedly fed to the proposed BTCN while dropout remains activated. The mean of these aggregated softmax probabilities is taken as the final predictions, while their variance is used to quantify predictive uncertainty.

A high-level architectural view of BTCN is given in Figure 5.1. BTCN fuses TCNs with SE blocks [93] to enrich the learned temporal trajectory representations with knowledge of feature map interdependencies. At inference time, a sequence of motion features is extracted from each GPS trajectory and sampled from BTCN multiple times while dropout is activated. Variational inference is approximated by the mean of these aggregated softmax probabilities constituting the final predictions, while their variance is used to estimate aleatoric and epistemic uncertainties.

5.2.1 Bayesian Temporal Convolutional Network

Given GPS trajectory $\mathbf{T}$ converted into a sequence of motion features $\mathbf{X}$, one way to approach trajectory segmentation is via standard sequence modelling practice, i.e., using recurrent neural networks. However, recurrent architectures can be cumbersome to train, owing to reduced parallelism, vanishing or exploding gradients, as well as high memory requirements [88]. On the other hand, recent work has demonstrated that TCNs can effectively address the above issues and even outperform recurrent architectures in established sequence modelling tasks [88]. As such, the architecture of the proposed BTCN is based on TCNs.

Architecture

Similarly to TCNs, the basic component of BTCN is the temporal residual block. As shown in Figure 5.2(a), the residual path first performs an optional 1D convolution with unit kernel size to ensure that the number of input features matches the desired number of output channels. Crucially, the latter is set equal to the length of the input sequence, allowing TCNs to produce sequences of the same length as their input. Next, the block applies two consecutive 1D convolutions that are dilated, ReLU-activated, and batch-normalized, before feeding their output to a final dropout layer. Contrary to standard convolutions, dilated

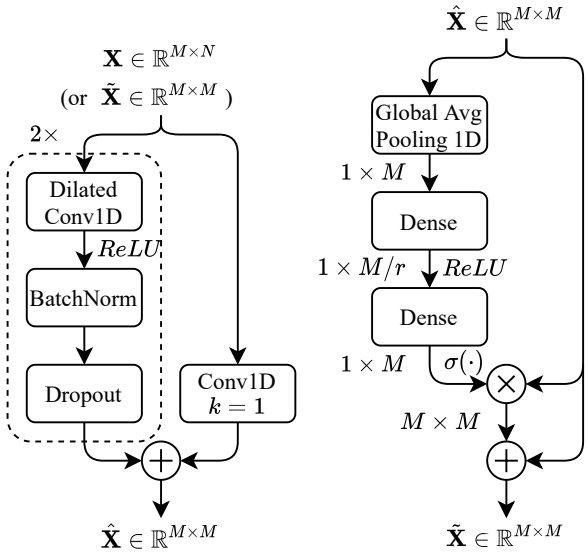

Figure 5.2 : The main components of the proposed BTCN. Temporal residual blocks leverage dilated 1D convolutions to capture high-resolution temporal information without need for downsampling. Always-on dropout layers are inserted to approximate variational inference via MC dropout sampling. The feature maps produced by each temporal residual block are subsequently recalibrated via an SE block.

ones apply filters over a larger area than their own by skipping input values with step d, otherwise known as dilation rate. As a result, dilation allows convolution layers to expand their receptive fields as the network depth grows, preserving input resolution without need for downsampling.

While stacking temporal residual blocks allows TCNs to uncover temporal dependencies among timesteps, we posit that it may not sufficiently account for dependencies within the channels themselves. Consequently, we follow recent work in this direction [93] and incorporate SE blocks after each temporal residual block. As shown in Figure 5.2(b), the SE block first embeds channel informa-

tion in $\hat{\mathbf{X}} \in \mathbb{R}^{M \times N}$ across the temporal dimension using global average pooling. Specifically, a vector $\mathbf{z} \in \mathbb{R}^{M}$ is built such that:

$$\mathbf{z}_m = \frac{1}{M} \sum_{m=0}^{M-1} \mathbf{c}_m. \tag{5.5}$$

Then, the SE block scales down the resulting feature maps by a factor of $r \in \mathbb{Z}^+$, before rescaling them and applying a self-gating mechanism. This is achieved using two fully connected (or dense) layers with weights $\mathbf{W}^{(0)} \in \mathbb{R}^{M/r}$ and $\mathbf{W}^{(1)} \in \mathbb{R}^{M}$; the operations can be formally described as follows:

$$\hat{\mathbf{z}} = \sigma(\mathbf{W}^{(1)}(\mathrm{ReLU}(\mathbf{W}^{(0)}\mathbf{z}))), \tag{5.6}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function. The gated output $\hat{\mathbf{z}}$ is finally multiplied with the block's input to construct the final output $\tilde{\mathbf{X}} \in \mathbb{R}^{M \times N}$, effectively promoting the most useful channels.

Dropout Variational Inference

As mentioned earlier, BTCN approximates variational inference via dropout. This corresponds to injecting each deterministic $\mathbf{W}^{(l)} \in \theta$ with noise following a Bernoulli distribution to create its stochastic counterpart. In practice, dropout is applied after each convolution in the network with probability of dropping connections p_{drop}, except for the output layer. Importantly, dropout is enabled not only during training but also at inference time; we perform S stochastic forward passes of $\mathbf{X}$ to obtain class probabilities $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{M \times K \times S}$, which are averaged to produce the posterior distribution of class probabilities $\hat{\mathbf{P}} \in \mathbb{R}^{M \times K}$. This process is known as MC dropout sampling [85].

Predictive Uncertainty Quantification

Given S MC dropout samples with class probabilities $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{M \times K \times S}$ reshaped into $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{S \times M \times K}$, the total predictive uncertainty can be approximated

by the variance $\sigma[\boldsymbol{\mathcal{P}}]^2$ of $\boldsymbol{\mathcal{P}}$, defined by Ribeiro *et al.* [96] as:

$$\begin{aligned}\sigma[\boldsymbol{\mathcal{P}}]^2 \approx\ & \mathrm{tr}(\underbrace{\mathbb{E}[\mathrm{diag}(\boldsymbol{\mathcal{P}}) - \boldsymbol{\mathcal{P}}\boldsymbol{\mathcal{P}}^{\mathsf{T}}]}_{\text{aleatoric}} + \underbrace{\mathbb{E}[\boldsymbol{\mathcal{P}}^2] - \mathbb{E}[\boldsymbol{\mathcal{P}}]^2}_{\text{epistemic}}) \\ =\ & \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{E}),\end{aligned} \tag{5.7}$$

where matrices $\mathbf{A}$ and $\mathbf{E}$ correspond to aleatoric and epistemic uncertainties, respectively. Their traces are then taken to produce a single aleatoric and epistemic uncertainty value per timestep.

5.2.2 Segmentation Objective Function

For unsupervised trajectory segmentation, we leverage the mutual-information-based deep clustering method in reference [55] for its demonstrated effectiveness in semantic image segmentation over centroid-based clustering approaches. It takes the K-dimensional softmax predictions for each timestep in the input sequence and attempts to maximize the mutual information between neighboring pairs of timestep patches.

Note that the original image-based work obtained pairs of neighboring patches not only within each image in the dataset, but also from synthetic images generated via multiple data augmentations; however, time series data augmentation is non-trivial, especially for the problem at hand. Intuitively, it would be hard to verify whether perturbed motion features would still realistically correspond to the same transportation mode. Therefore, considering only adjacent pairs of timestep patches centered at u and $u + t$, where $t \in T \subset \mathbb{Z}^+$ is a small displacement, the original clustering objective can be written as:

$$\max_f \frac{1}{|T|} \sum_{t \in T} I(P_t), \tag{5.8a}$$

$$P_t = \frac{1}{n|\Omega|} \sum_{i=0}^{n-1} \sum_{u \in \Omega} f_u(\mathbf{X}^{(i)}) \cdot [f(\mathbf{X}^{(i)})]_{u+t}^{T}, \tag{5.8b}$$

where f is a neural network, in this case BTCN, and $f_u(\mathbf{X}^{(i)})$ is its output for patch $u \in \Omega = \{0, \ldots, M-1\}$ centered at timestep u within the input motion feature sequence $\mathbf{X}^{(i)}$.

5.3 Experiments

5.3.1 Dataset and Simulation Setup

We conducted our experiments on the hardware detailed in . Deep learning models were developed using TensorFlow 2.3.0. Each baseline was implemented as per the description in its cited study. The reported values for all evaluation metrics were averaged over five executions.

Dataset

We evaluate BTCN and the selected baselines (see below) on Geolife [1, 34], a description of which was provided in Section 4.3.1. In accordance with Zheng *et al.* [1, 34], we view taxis and private cars as a single *car* class, while subways and trains are merged into the *train* class. We follow the literature [1, 34, 3] in only selecting trajectories that involve segments of the following classes: *walk*, *bike*, *bus*, *car*, and *train*. For evaluation purposes, we only use labeled trajectories; no ground-truth labels are involved in model training. All trajectories are split into chunks of length M and divided into training and test sets using a 90/10 ratio.

Preprocessing

For each trajectory, we first discard any GPS point whose timestamp exceeds that of the following point, or whose geographic coordinates fall outside of valid ranges. Next, we extract velocity, acceleration, and jerk features using eqs.

(4.1 – 4.4). We also estimate the turn $u^{(i)}$ as follows:

$$u^{(i)} = \tan^{-1} \frac{\text{Geodesic}(\text{lat}^{(i)}, \text{lon}^{(i)}, \text{lat}^{(i-1)}, \text{lon}^{(i)})}{\text{Geodesic}(\text{lat}^{(i)}, \text{lon}^{(i)}, \text{lat}^{(i)}, \text{lon}^{(i-1)})} - \tan^{-1} \frac{\text{Geodesic}(\text{lat}^{(i)}, \text{lon}^{(i)}, \text{lat}^{(i+1)}, \text{lon}^{(i)})}{\text{Geodesic}(\text{lat}^{(i)}, \text{lon}^{(i)}, \text{lat}^{(i)}, \text{lon}^{(i+1)})}. \tag{5.9}$$

Note that we empirically select these four motion features following established transportation mode identification literature [1, 3, 44]. The features are finally standardized by removing the mean and scaling to unit variance. We empirically set the sample length M to 128, such that $\mathbf{X} \in \mathbb{R}^{128\times4}$.

Model Configuration

BTCN uses 3 consecutive pairs of temporal residual and SE blocks. Within the i-th temporal residual block, both convolutions combine a larger kernel size of 8 with exponentially increasing dilation rates $d = 2^i$, $i \in \{0, 1, 2\}$. The dropout probability p_{drop} is set to 0.2 for all dropout layers, as in reference [96]; it is maintained at test time as required for MC dropout sampling, from which $S = 50$ samples are obtained. For all SE blocks, the reduction ratio r is set to 8. We train BTCN for 200 epochs using the Adam optimizer with a learning rate of 10^{-5} and default hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$.

Baselines

BTCN is evaluated against the four most prominent trajectory segmentation methods among the relatively few found in the GPS-based transportation mode identification literature:

- **Uniform segmentation.** Dabiri and Heaslip [23] follow a simple approach to trajectory segmentation by partitioning trajectories into fixed-size, non-overlapping segments. Specifically, they use a window of 200 points, which

they reported as the median length of the labeled GPS trajectories in the Geolife dataset.

- **Heuristics-based change point detection.** Zheng *et al.* [1] follow the intuition that walking segments must separate any other transportation modes, proposing a method that divides trajectories into alternating *walk* and *non-walk* segments based on distance, velocity, and acceleration thresholds. Xiao *et al.* [26] build on this seminal approach by also identifying gaps due to GPS signal interruption as change points. We do not include references [24, 25, 27] in our experiments since they constitute to a considerable extent hyperparameter variations of Zheng *et al.* [1].

- **Optimization-based change point detection.** After applying uniform segmentation to each trajectory, Dabiri *et al.* [3] convert the resulting segments into multivariate time series of velocity and acceleration and pass them to an optimization-based model. This model produces subsegments such that the homogeneity within each is maximized, while the number of change points is managed via a hyperparameter-controlled linear function.

We note that, while other deep clustering frameworks such as the ones in references [53, 54] may also be adapted to transportation mode segmentation, our aim is to evaluate the proposed framework against existing work designed specifically for transportation mode segmentation of GPS trajectories.

Evaluation Metrics

Following standard clustering evaluation practice [48, 55], all classes predicted by BTCN and its variants are first mapped to the ground-truth classes using linear assignment [97]. Segmentation performance is then measured by timestep-level accuracy (ACC).

Although BTCN assigns transportation mode labels at timestep-level, the evaluated baselines are not endowed with label information. Instead, they only detect transportation mode *change points* (the discrete timesteps where the mode changes), without explicit knowledge of the previous or the next transportation mode. To enable direct comparison with the proposed BTCN, we simply scan its predictions from left to right and detect change points at the timesteps where the predicted transportation mode changes. We finally evaluate BTCN against the selected baselines using the following evaluation metrics:

- **Mean Absolute Error (MAE)**. Since the number of predicted and ground-truth change points is likely to differ, we first map each predicted change point $\hat{\mathbf{c}}_i \in \hat{\mathbf{c}}$ to its nearest ground-truth $\mathbf{c}_j \in \mathbf{c}$. With most trajectories in Geolife having been sampled every 1–5 seconds, we maintain generality by measuring the distance separating two change points by the number of discrete timesteps between them. Given $\|\hat{\mathbf{c}}\|$ predicted change points $\hat{\mathbf{c}}$ and their nearest ground-truth change points $\mathbf{c}'$, we then calculate the MAE as $\text{MAE} = \frac{1}{\|\hat{\mathbf{c}}\|} \sum_{i=0}^{\|\hat{\mathbf{c}}\|-1} |\mathbf{c}'_i - \hat{\mathbf{c}}_i|$. Our choice of this metric is under the assumption that the number of timesteps between $\mathbf{c}'_i$ and $\hat{\mathbf{c}}_i$ matters more than whether $\hat{\mathbf{c}}_i$ precedes or comes after $\mathbf{c}'_i$.

- **Prediction Ratio (PR)**. Let $\|\hat{\mathbf{c}}\|$, $\|\mathbf{c}\|$ denote the total numbers of predicted change points $\hat{\mathbf{c}}$ and true change points $\mathbf{c}$, respectively. We compute the PR as $\text{PR} = \|\hat{\mathbf{c}}\|/\|\mathbf{c}\|$. When $\text{PR} < 1$, the segmentation model is said to under-predict change points, resulting in noisy segments with multiple transportation modes. In the case of over-prediction ($\text{PR} > 1$), the segmentation model tends to produce shorter segments that are less informative and thus harder to classify. Either case may complicate transportation mode identification even if the MAE is low.

Table 5.1 : GPS trajectory segmentation evaluation results. Where applicable, accuracies are reported for each class, with ACC denoting global accuracy. Higher ACC, lower MAE, and PR ≈ 1 is better. Training time is in minutes.

Segmentation Method	*Walk*	*Bike*	*Bus*	*Car*	*Train*	ACC	MAE	PR	Train (min)
Dabiri and Heaslip [23]			N/A			N/A	728.4	3.12	N/A
Dabiri *et al.* [3]			N/A			N/A	526.1	**1.23**	0.02
Xiao *et al.* [26]			N/A			N/A	287.7	2.34	N/A
Zheng *et al.* [1]			N/A			N/A	234.5	1.91	N/A
TCN (non-Bayesian, no SE)	0.821	0.755	0.416	0.376	0.726	0.619	31.5	3.51	33.89
TCN (non-Bayesian, $r = 8$)	0.799	0.782	0.441	0.367	**0.812**	0.640	26.1	2.92	45.12
BTCN (no SE)	0.792	0.752	0.414	0.391	0.775	0.625	29.7	2.84	33.67
BTCN ($r = 2$)	**0.827**	0.727	0.349	0.478	0.753	0.627	25.8	2.04	45.14
BTCN ($r = 4$)	0.809	0.773	0.353	**0.485**	0.744	0.632	28.3	2.43	45.21
BTCN ($r = 8$)	0.820	**0.811**	0.396	0.475	0.789	**0.658**	**24.7**	1.97	45.15
BTCN ($r = 16$)	0.773	0.799	**0.454**	0.416	0.751	0.638	28.1	2.44	44.93

5.3.2 Results

Performance Evaluation

Our experimental results are summarized in Table 5.1. BTCN consistently and significantly outperformed all evaluated methods, with its lowest MAE of 24.7 constituting a nearly 10× improvement over the best-performing baseline [1]. Recall that our definition of MAE measures the mean number of discrete timesteps between true and predicted change points; for unit GPS sampling rate, our result means BTCN would only take approximately 25 seconds on average to identify a change in transportation mode, while the above baseline would require nearly 4 minutes. Although the proposed method did demonstrate change point over-prediction on par with the baselines, it is important to note that BTCN is designed for timestep-level classification rather than change point detection. In

fact, we formalized trajectory segmentation based on mutual information maximization alone, without explicitly penalizing over-segmentation. This is an open problem, especially in the absence of labels, and is left for future work.

Regarding the baselines, we find that despite its simplicity of implementation, uniform segmentation [23] ultimately resulted in noisy segments involving multiple transportation modes. This is empirically supported by our experiments, where uniform segmentation attained not only the highest MAE but also the highest PR. The heuristics-based baselines [1, 26] both fared significantly better in terms of MAE, but still predicted nearly twice the number of change points compared to the ground truth. This is likely because hard thresholds are unable to account for a wide range of unexpected traffic conditions [1]. We also observe that optimization-based segmentation [3] achieved the best PR yet had the second highest MAE of 526.1. This result is within our expectations, as this method is individually applied to uniformly-segmented samples and does not consider the entire distribution. In other words, it makes the strong assumption that the samples are independent and identically distributed. Such an assumption is unlikely to be true for trajectories collected from numerous users at different times and traffic or weather conditions.

Ablation Study

To quantify the contribution of dropout variational inference and SE blocks to the performance of BTCN, we evaluate the latter when both components are disabled, as well as by removing either component while maintaining the other. As shown in Table 5.1, the non-Bayesian variant without SE blocks achieved the lowest global and per-class accuracy. Adding MC dropout pushed the global accuracy from 61.9% to 62.5%, while enabling SE blocks with $r = 8$ instead boosted it to 64.0%, and combining both components resulted in the highest global accu-

Table 5.2 : Performance and uncertainty metrics for BTCN over number of Monte Carlo samples.

S	ACC	MAE	PR	Aleatoric	Epistemic
1	0.6763	32.6	4.03	1.047E-02	0.0
2	0.6769	31.0	3.23	1.050E-02	3.902E-01
5	0.6775	29.3	2.76	1.043E-02	6.401E-01
10	0.6776	28.3	2.41	1.042E-02	7.162E-01
20	0.6777	27.2	2.22	1.048E-02	7.516E-01
50	0.6778	26.5	2.08	1.044E-02	7.746E-01
100	0.6778	26.1	2.02	1.047E-02	7.818E-01

racy at 65.8%. In fact, the best accuracy for four out of five classes was attained by the full BTCN, albeit with different reduction ratios r, while removing either of the Bayesian or SE components resulted in nearly twice the over-prediction of change points. We note that, while some ablations were relatively close in terms of MAE, the results for ACC and PR were much more varied, thus painting a clearer picture of how each ablation influenced segmentation performance.

Finally, Table 5.2 shows the sensitivity of evaluation and uncertainty metrics to the number of MC samples S. For this experiment, we used the best-performing BTCN out of 5 executions. Although global accuracy did not seem to drastically improve as S increased, the MAE dropped by about 19% and the PR was nearly halved. No noticeable improvement to accuracy was observed when $S > 50$, which is the value we used throughout our experiments.

Uncertainty-filtered Segmentation

For safety-critical intelligent transportation applications involving GPS trajectory segmentation, the ability of a model to selectively make timestep-level predictions based on some measure of confidence could be beneficial. Figure 5.3 shows how global and per-sample accuracy varies under different confidence percentiles for the best-performing BTCN out of 5 executions. In this context, confidence was simply defined as the negative aleatoric or epistemic uncertainty.

It appears that selectively classifying timesteps with higher aleatoric confidence consistently achieved higher global and per-class accuracy. On the contrary, we noted minimal global accuracy improvement in the case of epistemic confidence; there were almost no variations until the 90^{th} percentile, followed by a sharp drop for the 95^{th} and 99^{th} percentiles. It is hypothesized that the poor performance of epistemic confidence as a measure of accuracy in our work is tied to the unsupervised, mutual-information-based objective function that BTCN was trained with.

5.4 Summary

In this chapter, we introduced a novel GPS trajectory segmentation approach to address the shortcomings of related GPS-based work in learning from unlabeled data and capturing predictive uncertainty. Viewing trajectory segmentation through the scope of semantic image segmentation, the proposed BTCN reached 65.8% timestep-level accuracy on Microsoft's Geolife dataset, significantly outperforming established GPS-based baselines. We also conducted an ablation study to empirically validate the necessity of its components, and showed that BTCN effectively captured uncertainty by producing higher accuracy for input timesteps with lower aleatoric uncertainty.

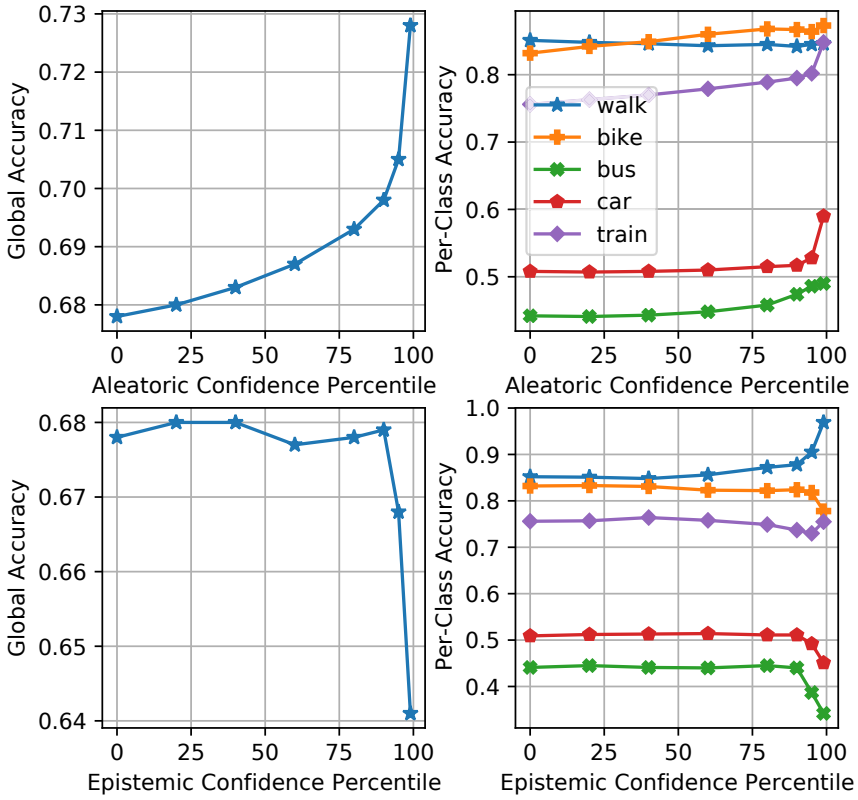

Figure 5.3 : Global and per-class accuracy of BTCN when only classifying timesteps that exceed confidence thresholds.

Chapter 6

Unsupervised Trajectory Transportation Mode Identification

In the previous chapter, we introduced a transportation mode segmentation model based on unsupervised deep learning to handle use cases where transportation mode labels are not available. Here, we focus on scenarios where trajectories are again unlabeled but have already been segmented by transportation mode. This can be achieved via any of the existing segmentation methods based on heuristics or change point detection; please refer to Section 2.2 for more details. Concretely, this chapter develops an unsupervised deep learning framework for transportation mode identification based on GPS data. First, we pretrain a Convolutional AutoEncoder (CAE) on fixed-length, single-transportation-mode trajectory segments converted in sequences of *local* motion features. Next, we connect the embedding layer of the CAE to a custom clustering layer whose trainable weights correspond to cluster centroids, following Guo *et al.* [83]. We finally resume training the resulting clustering model by balancing its clustering and reconstruction losses such that clustering-friendly structure is encouraged in the learned input representations. Since label information is not used at training

time, we also integrate motion-related statistics computed over each segment as *global* features. The proposed clustering framework attains 80.5% accuracy on Geolife [1, 34] without depending on ground-truth labels. As far as we know, this constitutes the first approach to grouping GPS trajectory segments by mode of transportation based on unsupervised deep learning.

The remainder of this chapter is organized as follows. Section 6.1 formulates the problem of clustering segments of GPS trajectories by travel mode. Section 6.2 presents the proposed framework, including our data preprocessing pipeline, the segment-level or *global* features that we incorporated to boost clustering accuracy, and the components of the clustering model. Section 6.3 describes our experimental setup and ablation study results, with Section 6.4 concluding this chapter. For definitions of key GPS-trajectory-related terms used in this chapter, please refer to Section 4.1. The content of this chapter is taken from a conference paper [98] that has been published in the proceedings of the IEEE 23rd International Conference on Intelligent Transportation Systems.

6.1 Problem Formulation

Problem 6.1 (Clustering by Transportation Mode). Given GPS trajectory $\mathbf{T}$ transformed into motion feature sequence $\mathbf{X}$, we frame the problem of clustering the single-transportation-mode segments in $\mathbf{X}$ into K clusters as the minimization of a custom clustering loss. Concretely, we aim to assign each single-transportation-mode segment to one of K disjoint clusters such that the probability distance between an approximated auxiliary distribution $\mathcal{Q}$ and a *true* distribution $\mathcal{P}$ of neural-network-encoded trajectory segments is optimized. For further details on the proposed approach, the reader is referred to Section 6.2 .

6.2 Proposed Framework

This section first describes the CAE that we use to learn lower-dimensional embeddings of high-dimensional input data. Then, it specifies the custom clustering layer holding learnable weights corresponding to cluster centroids that we connect to the embedding layer of the CAE. Finally, it introduces the two objective functions that fuse the clustering layer with the CAE, resulting in the proposed clustering model.

6.2.1 Convolutional Autoencoder

Autoencoders are a class of neural networks that aim to reconstruct their input under deliberate constraints that are set to promote the extraction of *meaningful* embeddings or latent representations. Without sacrificing generality, an autoencoder comprises an encoder network that embeds $\mathbf{X}$ to a latent representation $\mathbf{H}$, based on which a decoder network produces $\hat{\mathbf{X}}$ in an attempt to recover $\mathbf{X}$. As a special case of autoencoders, CAEs leverage convolutions to process their inputs. When the dimensionality of the input is larger than that of the autoencoder's embedding layer, we refer to the autoencoder as *undercomplete*.

We draw inspiration from deep clustering approaches (see Section 2.5) involving joint optimization of the reconstruction and clustering losses [83, 84], due to their balance between simplicity and effectiveness. To learn preliminary input embeddings, a fully convolutional undercomplete autoencoder is first pretrained; the term *fully convolutional* means that no fully connected (or dense) layers are employed, which is unlike related work in CAE-based deep clustering [84, 48, 83]. For this adaptation, we leverage convolutions with unit kernel length and shape transformation layers. We design the encoder and decoder networks to be symmetrical, with the former being built by stacking convolution layers connected to max pooling layers; that is, with the exception of the encoder's embedding layer,

which is its last convolution layer and is not connected to a max pooling layer. The decoder network stacks deconvolution and upsampling layers, instead. We configure convolution and deconvolution layers to use *same* padding so that their outputs maintain the input length. Each convolution and deconvolution layer (except for the output layer) is batch-normalized to accelerate training convergence.

Following our trajectory preprocessing steps in Section 4.2.1, all trajectories are converted to multivariate motion feature sequences of velocity, acceleration, and jerk. We then partition these sequences using the available labels so that each of the resulting segments involves exactly one mode of transportation. The resulting single-transportation-mode segments vary significantly in length. Yet our CAE only accepts *fixed-size* sequences of shape (L, N), where L denotes the number of GPS points (or timesteps) in a segment of N motion features. For the proposed framework, we have empirically selected $L = 128$, $N = 3$ based on preliminary experiments. We therefore partition all single-transportation-mode segments into non-overlapping slices $\mathbf{X} \in \mathbb{R}^{L \times N}$ and eliminate any segment with fewer than L remaining GPS points.

Note that a simple way to circumvent the CAE's requirement for fixed-size inputs is to use recurrent autoencoders. Nonetheless, recurrent neural networks are significantly slower to train and often suffer from gradient vanishing or explosion issues [88]. Besides, setting a minimum number of GPS points per segment follows the intuition that short segments, especially those where a large portion of their GPS points have near-zero velocity, may not be informative enough to cluster confidently. For instance, Dabiri *et al.* [3] set a much smaller minimum number of 20 GPS points per segment of length 248 and perform zero-padding; however, zero-padded features are still processed by CNNs, and the authors did not mask or otherwise account for these features when designing the cost function.

To avoid biasing the learned trajectory representations, in contrast to Dabiri *et al.* [3], we do not implement zero-padding and do not make additional assumptions such as minimum segment distance or duration to preserve generality.

We pretrain the CAE by minimizing the reconstruction loss L_r, measured by the mean squared error between original sample $\mathbf{X} \in \mathbb{R}^{L \times N}$ and its reconstruction $\mathbf{X} \in \hat{\mathbb{R}}^{L \times N}$ as follows:

$$L_r = \frac{1}{LN} \sum_{i,j} (\mathbf{X}_{i,j} - \hat{\mathbf{X}}_{i,j})^2. \tag{6.1}$$

We note that our CAE operates on fixed-size, medium-length inputs. This ensures that the segments to be subsequently clustered each correspond to exactly one transportation mode and are long enough to be informative, at the expense of discarding smaller amounts of otherwise potentially useful data.

6.2.2 Clustering Layer

After the CAE is pretrained, we connect its embedding layer to a custom clustering layer as in reference [83]. The latter holds cluster centroids as trainable weights $\mathbf{A}$, where $\mathbf{A}_{j,:}$ denotes the j-th centroid's coordinates and K refers to the number of desired clusters. In this study, we set $K = 5$ to equal the number of classes that we aim to recognize in the dataset, as will be discussed in Section 6.3.1.

During forward propagation, we employ Student's t-distribution to estimate probabilities of cluster membership $\mathbf{Q}_{i,j}$ for the i-th embedded sample $\mathbf{Z}^{(i)}$ according to:

$$\mathbf{Q}_{i,j} = \frac{(1 + \|\mathbf{Z}^{(i)} - \mathbf{A}_{j,:}\|^2)^{-1}}{\sum_j (1 + \|\mathbf{Z}^{(i)} - \mathbf{A}_{j,:}\|^2)^{-1}}. \tag{6.2}$$

Given $\mathbf{Q}_{i,j}$, we obtain soft label $q^{(i)}$ for $\mathbf{Z}^{(i)}$ as the index of the maximum probability in $\mathbf{Q}_{i,:}$ and approximate the normalized target distribution $\mathcal{P}$ by:

$$\mathbf{P}_{i,j} = \frac{\mathbf{Q}_{i,j}^2 / \sum_i \mathbf{Q}_{i,j}}{\sum_j (\mathbf{Q}_{i,j}^2 / \sum_i \mathbf{Q}_{i,j})}. \tag{6.3}$$

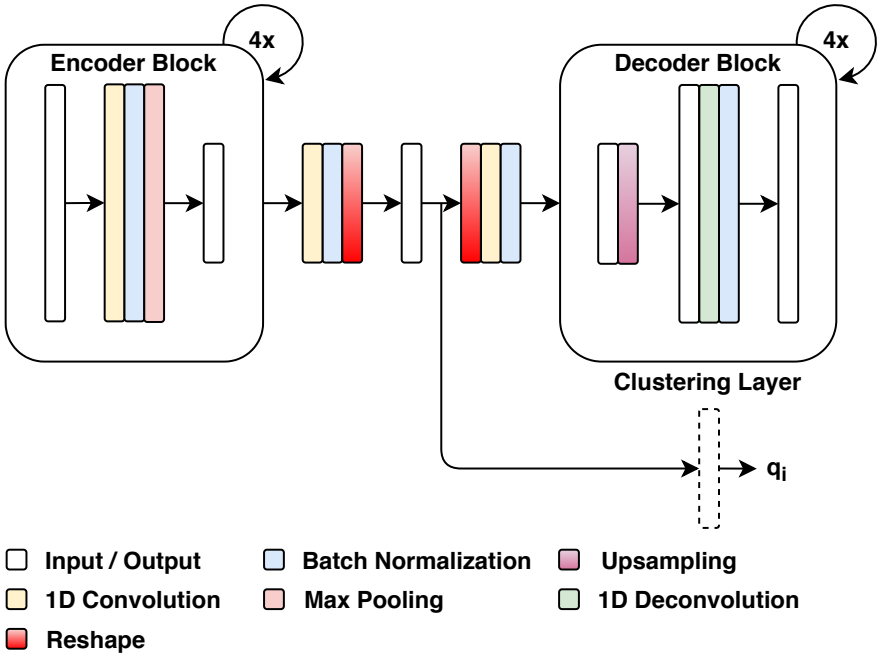

Figure 6.1 : Overview of the proposed deep clustering model. We connect the embedding layer of a CAE to a custom clustering layer that holds trainable weights corresponding to cluster centroids. These are learned jointly with the parameters of the CAE following pretraining.

During backpropagation, embedded samples $\mathbf{Z} \in \mathfrak{Z}$ are clustered through optimization of the Kullback-Leibler (KL) divergence between distributions $\mathcal{P}$ and $\mathcal{Q}$:

$$L_c = \mathrm{KL}(\mathcal{P} \parallel \mathcal{Q}) = \sum_i \sum_j \mathbf{P}_{i,j} \log \frac{\mathbf{P}_{i,j}}{\mathbf{Q}_{i,j}}. \tag{6.4}$$

6.2.3 Composite Clustering Model

Our clustering model, visualized in Figure 6.1, is initialized by pretraining a CAE and connecting its embedding layer to a specialized clustering layer according to Sections 6.2.1 and 6.2.2. Next, the clustering model is trained by optimizing the following:

$$L = L_r + \gamma L_c, \tag{6.5}$$

where γ is a hyperparameter that adjusts the influence of the clustering loss. Selecting $\gamma = 1$ and $L_r = 0$ reduces the above to the objective function employed by Xie *et al.* [48]. This essentially discards the decoder of the CAE and is likely to cause the optimization process to distort the learned representations. Instead, Guo *et al.* [83] use $\gamma = 0.1$ to regulate the effect of the clustering loss, which they show results in better-defined clusters.

A significant consideration is the frequency by which the target distribution $\mathcal{P}$ should be updated. While doing so too often may destabilize the clustering process, the opposite may also trap it in local minima. To adjust the frequency of updating the target distribution, we define hyperparameter ϕ as a fraction of a training epoch. For example, $\phi = 0.5$ calls for updating the target distribution twice in a single epoch, while $\phi = 2$ instead results in updating it once per two epochs.

It is important to note that, by predicting the transportation mode for every timestep in the input motion feature sequence, the segmentation model that we introduced in Section 4.2 can be used for both trajectory segmentation and mode identification. However, while it learns to distinguish transportation modes strictly based on labels, the clustering model is instead guided by the motion feature similarity between segments. As such, the latter could help experts gain useful insights into designing and optimizing ITSs by uncovering various mobility-related patterns that the supervised segmentation model could not. For instance, it could highlight classes with high sample variance; as shown in the leftmost subplot of Figure 6.2, cars were often mistaken for buses and trains due to exhibiting similar point-level motion patterns. Another use case could be to cluster segments for individual users, thereby helping understand their traffic behaviors in relation to other users. Finally, given the time-consuming, error-prone nature of manual trajectory annotation, it is likely that any such dataset

will contain erroneous labels. While training the supervised segmentation model on mislabeled samples could result in misclassified segments, the latter could still be safely used to train the clustering model since it does not learn from label information.

We also note that, in our experiments, we can determine the transportation modes associated with each cluster because the ground-truth labels are available (but not seen by the model during training). In real-world applications where ground-truth data are not available, determining the specific transportation mode associated with each cluster may require further analysis by transportation domain experts, since the proposed clustering method clusters trajectory segments based on their motion features.

6.2.4 Global Features

Along with the three *local* features extracted using eqs. (4.2 – 4.4) in Section 4.2.1 (velocity v, acceleration a, jerk k), we further integrate four features based on velocity for each trajectory segment. Our intuition is that these *global*, segment-level features may serve as meaningful clues at training time.

Specifically, we estimate for each truncated segment $\mathbf{X}$ of length L the (1) mean, (2) expectation of, and (3) max velocity. Drawing inspiration from reference [34], we further compute the (4) stop rate, which Zheng *et al.* define as the ratio of the number of timesteps where velocity falls below a set value to the overall distance of the associated segment. Instead, we modify the above definition to divide by the segment length L. Our definition of the stop rate can be formalized as $\|\mathbf{S}\|/L$, where $\mathbf{S} = \{\mathbf{p}^{(i)} | \mathbf{p}^{(i)} \in \mathbf{X}, v^{(i)} < 3\text{m/s}\}$. To fuse our *global* features with the three timestep-level (*local*) features, the former are repeated L times and concatenated with the latter, resulting in seven features in total.

6.3 Experiments

In this section, we first introduce the dataset used to evaluate the proposed transportation mode identification model and the hardware on which we performed our experiments. Then, we specify the model configuration, baselines, and evaluation metrics, before finally presenting our experimental results and ablation studies and hyperparameter sensitivity tests.

We conducted our experiments on the same computing server as in Section 5.3.1. We report the averaged results of five separate executions for all selected evaluation metrics.

6.3.1 Dataset and Simulation Setup

Dataset

We base our experiments on the openly available Geolife dataset [1, 34] which was described in Section 4.3.1. For model evaluation purposes, we only use labeled trajectories (belonging to 69 out of 182 participants) in our experiments. We note that it would be possible to also include unlabeled trajectories, e.g., by having the trained segmentation model in Section 4.2.1 perform inference on them and then incorporating the extracted segments in training of the clustering model. However, this would condition the evaluation of the clustering model on the aptitude of the segmentation model. Since the selected standard evaluation metrics require ground-truth labels, they are only applied to the labeled data.

We follow established transportation mode identification literature [1, 34, 3] in only retaining sufficiently-represented classes, namely *walk*, *bike*, *bus*, *car* (cars and taxis), and *train* (trains and subways). As per standard practice in unsupervised learning [48, 50], all labeled trajectories are used to both train and evaluate the clustering model; crucially, no ground-truth labels are involved when

training the latter. The originally labeled trajectories are finally preprocessed into sequences of local and global motion features, as described in Sections 4.2.1 and 6.2.4, respectively.

Model Configuration

The encoder network within the CAE is built by stacking five convolution layers with $\{32, 64, 128, 256, 1\}$ filters, respectively. The first three layers employ *same* padding and kernels of length 3, while the remaining ones use *valid* padding and a stride of 7 and 1, respectively. We use the ReLU function to activate every convolution and deconvolution layer, minus the final convolution on the encoder side and the final deconvolution of the decoder; these are not activated. We employ the Adam optimizer to pretrain the CAE for 600 epochs using an initial learning rate of 0.001 and default hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. We then attach the custom clustering layer and resume training the entire model with learning rate 0.0001. Except for our ablation studies, we set clustering hyperparameters $\phi = 2$, $\gamma = 20$ in our experiments. Training stops when, during an epoch, not more than 0.1% of segments are assigned to another cluster.

Baselines

To assess the hypothesized advantage of fusing *local* features with *global* ones, clustering performance is reported when using either *local*, *global*, or both sets of features. We evaluate the proposed framework against the following ten baselines, grouped into three categories:

1. **Traditional clustering on high-dimensional features.** We evaluate the k-Means (KM), Spectral Clustering (SC), and Hierarchical Agglomerative Clustering (HAC) algorithms on high-dimensional samples $\mathbf{X}$. We use their implementation as provided by the `scikit-learn` Python library. Because

these implementations are not designed for high-dimensional inputs, each feature is represented by its mean value for a given sample.

2. **Traditional clustering on embedded features.** In contrast to the above baseline group, where traditional clustering algorithms are applied to high-dimensional samples $\mathbf{X}$, here we apply them to K-dimensional samples $\mathbf{Z}$ as embedded by our pretrained CAE (without the clustering layer and associated loss). We disambiguate the resulting clustering cases as CAE-KM, CAE-SC, and CAE-HAC.

3. **Semi-supervised classification on high-dimensional features.** Given that, to the best of our knowledge, the literature has not yet explored unsupervised GPS-based transportation mode identification, we compare the performance of the proposed framework with the following four semi-supervised classification baselines: Semi-Two-Steps (STS), Semi-Pseudo-Label (SPL), SEmi-supervised Convolutional Autoencoder (SECA) [3], and Multi-View training with Proxy Labels (MVPL) [45]. STS first pretrains an autoencoder on labeled and unlabeled samples and then attaches a softmax layer, using the encoder-learned latent representations in supervised training. SPL generates pseudo-labels for the unlabeled data by selecting the most likely predicted class on every update of the neural network's weights, thereby allowing for joint supervised training on labeled and unlabeled instances. For a summary of SECA and MVPL, please refer to Section 2.4. The reported results for STS, SPL, and SECA are obtained from reference [3], where they are trained using 10% of the labeled data. The reported results for MVPL, which is trained on just 1% of labeled data, are similarly obtained from reference [45].

Evaluation Metrics

Given predicted cluster assignments $\hat{\mathbf{y}} \in \{0, \ldots, K-1\}^L$ and ground-truth labels $\mathbf{y} \in \{0, \ldots, K-1\}^L$, we quantify clustering performance using the following evaluation metrics:

- **Accuracy (ACC).** According to reference [99], clustering accuracy can be defined as:

$$\mathrm{ACC}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{\|\mathbf{y}\|} \sum_{i=0}^{\|\mathbf{y}\|-1} \delta(\mathbf{y}^{(i)}, \mathrm{Map}(\hat{\mathbf{y}}^{(i)})), \tag{6.6}$$

  where $\delta(\mathbf{y}, \hat{\mathbf{y}}) = 1$ if $\mathbf{y} = \hat{\mathbf{y}}$, otherwise $\delta(\mathbf{y}, \hat{\mathbf{y}}) = 0$. Function $\mathrm{Map}(\hat{\mathbf{y}}^{(i)})$ returns a mapping of predicted label $\hat{\mathbf{y}}^{(i)}$ to its associated ground-truth. The assignment $\mathrm{Map}(\hat{\mathbf{y}}^{(i)})$ is formulated as a linear program and determined based on the Hungarian method [97]. Please note that, unlike its prior definition in Chapter 5, accuracy here is measured at instance- rather than timestep level.

- **Normalized Mutual Information (NMI).** We define NMI as:

$$\mathrm{NMI}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\mathrm{MI}(\mathbf{y}, \hat{\mathbf{y}})}{\max(H(\mathbf{y}), H(\hat{\mathbf{y}}))}, \tag{6.7a}$$

$$\mathrm{MI}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=0}^{\|\mathbf{y}\|-1} \sum_{j=0}^{\|\hat{\mathbf{y}}\|-1} \frac{|\mathbf{y}^{(i)} \cap \hat{\mathbf{y}}^{(j)}|}{m} \log \frac{m|\mathbf{y}^{(i)} \cap \hat{\mathbf{y}}^{(j)}|}{|\mathbf{y}^{(i)}||\hat{\mathbf{y}}^{(j)}|}, \tag{6.7b}$$

  where $H(\mathbf{y})$ and $H(\hat{\mathbf{y}})$ are the entropies of vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$, and $\mathrm{MI}(\mathbf{y}, \hat{\mathbf{y}})$ denotes their mutual information. Here, NMI quantifies how much our knowledge about the ground-truth classes would increase given the predicted clusters.

Note that, although there exist evaluation metrics that do not rely on ground-truth labels, such as the Silhouette Coefficient or the Calinski-Harabasz Index, we select ACC and NMI following standard practice in the deep clustering literature.

Table 6.1 : Clustering evaluation results in terms of ACC and NMI (higher values are better).

Features	Local		Global		All	
Method	ACC	NMI	ACC	NMI	ACC	NMI
KM	.544	.387	.731	**.589**	.618	.507
SC	.405	.304	.699	.585	.591	.512
HAC	.521	.361	.662	.573	.595	.505
CAE-KM	.642	.419	.659	.544	.774	.610
CAE-SC	.479	.368	.657	.540	.752	.603
CAE-HAC	.649	.425	.618	.534	.757	.593
Proposed	**.701**	**.479**	**.738**	.586	**.805**	**.644**

Importantly, this also enables direct comparison with published results from the semi-supervised baselines.

6.3.2 Results

Our experimental results are reported in in Tables 6.1 and 6.2. The proposed clustering framework considerably outperformed the traditional clustering algorithms when using either *local* or all features. Compared with the semi-supervised baselines, it considerably outperformed STS, SPL, and SECA, with the latter reaching 62.9% accuracy using not only trajectory segments of approximately twice the length (248 GPS points, whereas we merely used 128), but also 10% ground-truth label information (versus our 0%). Although MVPL attained a higher accuracy than our framework, we note that it still leverages label information and benefits from 128 additional features extracted via Fourier and

Table 6.2 : Comparison of the proposed unsupervised approach (using *all* features) with competitive semi-supervised baselines.

Method	Labeled Data	ACC
STS [3]	10%	.544
SPL [3]	10%	.589
SECA [3]	10%	.629
MVPL [45]	1%	.848
Proposed	0%	.805

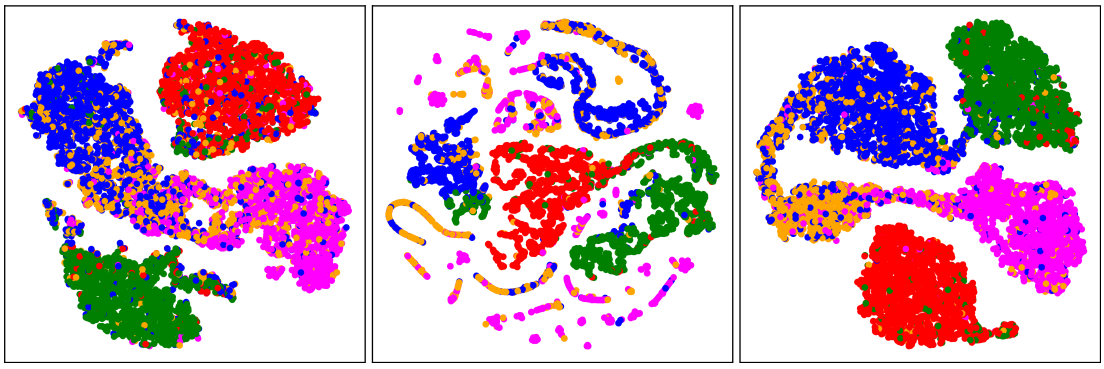

Figure 6.2 : Illustration of the clusters identified by the proposed clustering model when using timestep-level (*local*), segment-level (*global*), and *all* motion features. Samples plotted in green, red, orange, magenta, and blue correspond to *bike*, *walk*, *car*, *train* and *bus* classes.

wavelet transforms. We leave the exploration of these features towards improving the performance of our clustering model for future work. Overall, it appears that the combination of point-level or *local* features with segment-level or *global* ones produced significant ACC and NMI improvements for all baselines.

Figure 6.2 provides a visualization of the proposed framework's generated clusters in the two-dimensional space. When using *local* features (left subplot), we observe good overall cluster separation with the exception of the one associated with the *car* class. Even though just using *global* features attained higher accuracy, we note in the middle subplot that these features produced ill-formed clusters, thereby emphasizing the expected advantage of fusing *local* features with *global* ones. Indeed, this fusion resulted in not only the highest clustering accuracy but also the best formed clusters, as shown in the right subplot.

Regarding the traditional clustering methods, their subpar performance with either *local* or the combined features of the original high-dimensional data is not surprising. This partly because the features for each sample were averaged over 128 timesteps to obtain the appropriate dimensionality. Indeed, we observe that the same methods scored far better results with the lower-dimensional CAE-learned representations of the aforementioned features. They also fared significantly better on the *global* features, where k-Means almost matched the accuracy of the proposed framework; an intuitive explanation for this is that low dimensionality of the *global* features should better fit these clustering methods.

Hyperparameter Sensitivity

Since unsupervised learning is not guided by ground-truth annotations, model hyperparameter tuning becomes a non-trivial task. As such, it is desirable that the performance of unsupervised models not degrade significantly for sensible hyperparameter variations.

Table 6.3 : Sensitivity of accuracy to clustering loss strength γ.

Features	γ									
	.1	.2	.5	1	2	5	10	20	50	100
Local	.657	.668	.676	.681	.684	.690	.689	.701	.698	.699
Global	.708	.714	.693	.718	.692	.700	.721	.738	.696	.680
All	.796	.794	.786	.786	.792	.785	.791	**.805**	.790	.783

Table 6.4 : Accuracy versus target distribution update frequency ϕ.

Features	ϕ									
	.1	.2	.5	1	2	5	10	20	50	100
Local	.515	.535	.626	.693	.701	.694	.699	.700	.700	.699
Global	.648	.650	.665	.685	.738	.719	.706	.721	.726	.723
All	.769	.756	.799	.788	**.805**	.783	.802	**.805**	.804	.803

To examine the sensitivity of the proposed framework to hyperparameters γ and ϕ, we begin by setting $\phi = 2$ (i.e., a value that performs well in practice) and evaluate $\gamma \in \{0.1, 0.2, 0.5, 1, 2, \ldots, 100\}$. The results in terms of clustering accuracy are shown in Table 6.3. We observe that our model is rather sensitive to $\gamma < 1$ when using *local* features, as well as $\gamma > 20$ in the case of *global* features. On the contrary, when trained on both *local* and *global* features, it remains relatively stable throughout the selected range of γ values.

We then select $\gamma = 20$ and assess the proposed clustering framework for $\phi \in \{0.1, 0.2, 0.5, 1, 2, \ldots, 100\}$. According to Table 6.4, it appears that considerable instability occurs when using (i) *local* features and $\phi < 1$, or (ii) *global* features

and $\phi < 2$. Moreover, $\phi < 0.5$ also degrades clustering accuracy when using both *local* and *global* features. Overall, the proposed clustering framework is more robust to different values of ϕ when employing all features.

6.4 Summary

In this chapter, we introduced a framework based on unsupervised deep learning to cluster segments of GPS trajectories according to their associated transportation mode. After pretraining a convolutional autoencoder on fixed-length trajectory segments converted to motion feature sequences, we connected its embedding layer to a specialized clustering layer holding cluster centroids as learnable parameters. Next, we retrained the entire clustering model by jointly minimizing the weighted sum of the autoencoder's reconstruction loss and the custom layer's clustering loss. The proposed clustering framework attained 70.1% clustering accuracy on Geolife using point-level or *local* features, considerably outperforming not only traditional clustering methods but also the state-of-the-art in semi-supervised travel mode identification. Importantly, the fusion of *local* features with *global* ones computed for every trajectory segment boosted clustering accuracy to an even higher 80.5%. We conducted extensive case studies to evaluate the proposed framework's hyperparameter sensitivity and found that it is robust to the weight of the clustering loss as well as to the frequency by which we update the target distribution.

Chapter 7

Conclusion and Future Work

Transportation mode identification lies at the core of future intelligent transportation systems. The ability to deduce users' transportation modes from their GPS traces can benefit a variety of ITS operations, ranging from personalized advertisements and route recommendations to dynamic public transportation scheduling and traffic light optimization. Nonetheless, there remain significant challenges to its real-world applicability.

Although signal lapses in urban areas often render users' trajectories incomplete, most transportation mode identification approaches have not attempted to recover missing GPS points prior to preprocessing and classification. In Chapter 3, we introduced an uncertainty-aware imputation GAN to reconstruct incomplete sequences of magnitude and angle of displacement, before estimating unobserved GPS points using the recovered motion features. The proposed alternative imputation target allows UI-GAN to generalize to any geographic area without underlying map information.

With GPS sensors not readily capturing transportation mode information, the resulting trajectories are not automatically segmented by transporta-

tion mode. Inspired by recent developments in semantic image segmentation, in Chapter 4 we reframed trajectory segmentation as timestep-level transportation mode identification. Different from existing approaches focusing on identifying transportation mode change points using heuristics or optimization-based methods, the proposed deep learning framework is the first to learn the segmentation task from transportation mode labels.

Combined with privacy concerns, the labor-intensive nature of manual annotation mean that users' trajectories are typically not labeled by transportation mode. To this end, in Chapter 5 we proposed an unsupervised channel-calibrated Bayesian temporal convolutional network that learns to segment trajectories by maximizing the mutual information between temporally neighboring feature map patches, without using any transportation mode labels. BTCN leverages channel attention to capture feature map interdependencies and approximates variational inference via Monte Carlo dropout, using the mean and variance of the predicted distributions for timestep-level classification and predictive uncertainty estimation. To the best of our knowledge, BTCN is the first trajectory segmentation framework to leverage unsupervised deep learning and support uncertainty-filtered segmentation.

For the same reason, we introduced the first unsupervised deep learning framework for transportation mode identification in Chapter 6. After reducing the trajectory segments' dimensionality via unsupervised pretraining of an undercomplete convolutional autoencoder, we equip its embedding layer with a clustering module whose weights correspond to cluster centroids and resume training. We strike a balance between its reconstruction and clustering losses to encourage clustering-friendly latent representations and further incorporate motion-related features computed over each segment to improve accuracy.

We demonstrated the effectiveness of the proposed approaches via extensive experiments on established real-world datasets. Our experiments confirmed their superiority over existing approaches as well as the necessity of their components.

Future trajectory imputation and segmentation studies may consider using graph neural networks to capture not only temporal but also spatial information from non-Euclidean road network representations. Another promising research direction is how to effectively penalize over-segmentation in the absence of labels. This may be achieved via the design of novel segmentation objective functions, possibly in combination with appropriate post-processing schemes.

Most GPS-based transportation mode identification studies typically preprocess trajectories into sequences of motion features such as velocity or acceleration. These representations have been shown to be more suitable than raw GPS points for training machine or deep learning models, offering better generalization to geographic areas insufficiently covered in the training set. However, incomplete or irregularly-sampled trajectories result in noisy motion features and thus uncertain samples at training and test time. As such, it may be worthwhile to explore data-driven feature extraction in place of manual motion feature preprocessing. Finally, future work may investigate early classification approaches to satisfy the requirements of time-critical transportation applications.

Bibliography

[1] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw GPS data for geographic applications on the web," in *Proceedings of the 17th International Conference on World Wide Web*, Beijing, China, 2008, pp. 247–256.

[2] M. Perslev, M. Jensen, S. Darkner, P. J. Jennum, and C. Igel, "U-time: A fully convolutional network for time series segmentation applied to sleep staging," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 4417–4428.

[3] S. Dabiri, C.-T. Lu, K. Heaslip, and C. K. Reddy, "Semi-supervised deep learning approach for transportation mode identification using GPS trajectory data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, 2020.

[4] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European Conference on Computer Vision*, Munich, Germany, 2018, pp. 132–149.

[5] A. C. Prelipcean, G. Gidofalvi, and Y. O. Susilo, "Transportation mode detection – an in-depth review of applicability and reliability," *Transport Reviews*, vol. 37, no. 4, pp. 442–464, 2017.

[6] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, "Enhancing transportation systems via deep learning: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 144–163, 2019.

[7] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *IEEE 28th International Conference on Data Engineering*. Arlington, VA, USA: IEEE, 2012, pp. 1144–1155.

[8] P. Banerjee, S. Ranu, and S. Raghavan, "Inferring uncertain trajectories from partial observations," in *2014 IEEE International Conference on Data Mining*. Shenzhen, China: IEEE, 2014, pp. 30–39.

[9] Y. Li, Y. Li, D. Gunopulos, and L. Guibas, "Knowledge-based trajectory completion from sparse GPS samples," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Francisco, CA, USA, 2016, pp. 1–10.

[10] Z. Wang, S. Zhang, and J. J. Yu, "Reconstruction of missing trajectory data: A deep learning approach," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece: IEEE, 2020, pp. 1–6.

[11] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for dna microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, 2001.

[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[13] I. R. White, P. Royston, and A. M. Wood, "Multiple imputation using chained equations: issues and guidance for practice," *Statistics in Medicine*, vol. 30, no. 4, pp. 377–399, 2011.

[14] D. J. Stekhoven and P. Bühlmann, "MissForest—non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.

[15] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, no. 1, pp. 1–12, 2018.

[16] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," in *Advances in Neural Information Processing Systems*, 2018, pp. 6775–6785.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[18] J. Yoon, J. Jordon, and M. Van Der Schaar, "GAIN: Missing data imputation using generative adversarial nets," in *International Conference on Machine Learning*, 2018.

[19] S. C.-X. Li, B. Jiang, and B. Marlin, "MisGAN: Learning from incomplete data with generative adversarial networks," in *International Conference on Learning Representations*, 2019.

[20] Y. Luo, X. Cai, Y. Zhang, J. Xu *et al.*, "Multivariate time series imputation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1596–1607.

[21] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E2GAN: End-to-end generative adversarial network for multivariate time series imputation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3094–3100.

[22] Y. Liu, R. Yu, S. Zheng, E. Zhan, and Y. Yue, "NAOMI: Non-autoregressive multiresolution sequence imputation," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 238–11 248.

[23] S. Dabiri and K. Heaslip, "Inferring transportation modes from GPS trajectories using a convolutional neural network," *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 360–371, 2018.

[24] N. Schüssler and K. W. Axhausen, "Processing raw data from global positioning systems without additional information," *Transportation Research Record*, vol. 2105, no. 1, pp. 28–36, 2009.

[25] F. Biljecki, H. Ledoux, and P. Van Oosterom, "Transportation mode-based segmentation and classification of movement trajectories," *International Journal of Geographical Information Science*, vol. 27, no. 2, pp. 385–407, 2013.

[26] G. Xiao, Z. Juan, and C. Zhang, "Travel mode detection based on GPS track data and Bayesian networks," *Computers, Environment and Urban Systems*, vol. 54, pp. 14–22, 2015.

[27] Q. Zhu, M. Zhu, M. Li, M. Fu, Z. Huang, Q. Gan, and Z. Zhou, "Identifying transportation modes from raw gps data," in *International Conference of Pioneering Computer Scientists, Engineers and Educators*. Springer, 2016, pp. 395–409.

[28] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *arXiv preprint arXiv:1412.7062*, 2014.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition, Boston, MA, USA, 2015, pp. 3431–3440.

[30] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[31] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[32] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision*, Munich, Germany, 2018, pp. 801–818.

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, Munich, Germany, 2015, pp. 234–241.

[34] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding mobility based on GPS data," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, Seoul, Korea, 2008, pp. 312–321.

[35] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, "Transportation mode detection using mobile phones and GIS information," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, IL, USA, 2011, pp. 54–63.

[36] Z. Sun and X. J. Ban, "Vehicle classification using GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 37, pp. 102–117, 2013.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[39] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le, "Self-training with noisy student improves ImageNet classification," *arXiv preprint arXiv:1911.04252*, 2019.

[40] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Twenty-ninth AAAI Conference on Artificial Intelligence*, Austin, TX, USA, 2015.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[42] H. Wang, G. Liu, J. Duan, and L. Zhang, "Detecting transportation modes using deep neural network," *IEICE Transactions on Information and Systems*, vol. 100, no. 5, pp. 1132–1135, 2017.

[43] M. Simoncini, L. Taccari, F. Sambo, L. Bravi, S. Salti, and A. Lori, "Vehicle classification from low-frequency GPS data with recurrent neural networks," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 176–191, 2018.

[44] J. J. Q. Yu, "Travel mode identification with GPS trajectories using wavelet transform and deep learning," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[45] J. J. Q. Yu , "Semi-supervised deep ensemble learning for travel mode identification," *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 120–135, 2020.

[46] R. Zhang, P. Xie, C. Wang, G. Liu, and S. Wan, "Classifying transportation mode and speed from trajectory data via deep multi-scale learning," *Computer Networks*, vol. 162, p. 106861, 2019.

[47] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 5147–5156.

[48] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33rd International Conference on Machine Learning*, New York, NY, USA, 2016, pp. 478–487.

[49] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognition*, vol. 83, pp. 161–173, 2018.

[50] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 5736–5745.

[51] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017.

[52] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "ClusterGAN: Latent space clustering in generative adversarial networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, Honolulu, HI, USA, 2019, pp. 4610–4617.

[53] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1558–1567.

[54] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.

[55] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9865–9874.

[56] Y. Endo, H. Toda, K. Nishida, and A. Kawanobe, "Deep feature extraction from trajectories for transportation mode estimation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2016, pp. 54–66.

[57] L. Li, J. Zhu, H. Zhang, H. Tan, B. Du, and B. Ran, "Coupled application of generative adversarial networks and conventional neural networks for travel mode detection using GPS data," *Transportation Research Part A: Policy and Practice*, vol. 136, pp. 282–292, 2020.

[58] M. Rezaie, Z. Patterson, J. Y. Yu, and A. Yazdizadeh, "Semi-supervised travel mode detection from smartphone data," in *IEEE International Smart*

Cities Conference. Wuxi, China: IEEE, 2017, pp. 1–8.

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

[60] A. Bolbol, T. Cheng, I. Tsapakis, and J. Haworth, "Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification," *Computers, Environment and Urban Systems*, vol. 36, no. 6, pp. 526–537, 2012.

[61] Y. Duan, Y. Lv, Y.-L. Liu, and F.-Y. Wang, "An efficient realization of deep learning for traffic data imputation," *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 168–181, 2016.

[62] Y. Chen, Y. Lv, and F.-Y. Wang, "Traffic flow imputation using parallel data and generative adversarial networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1624–1630, 2019.

[63] L. Li, B. Du, Y. Wang, L. Qin, and H. Tan, "Estimation of missing values in heterogeneous traffic data: Application of multimodal deep learning model," *Knowledge-Based Systems*, 2020.

[64] E. F. d. S. Soares, K. Revoredo, F. Baião, C. A. de MS Quintella, and C. A. V. Campos, "A combined solution for real-time travel mode detection and trip purpose prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4655–4664, 2019.

[65] H. Gong, C. Chen, E. Bialostozky, and C. T. Lawson, "A GPS/GIS method for travel mode detection in New York city," *Computers, Environment and Urban Systems*, vol. 36, no. 2, pp. 131–139, 2012.

[66] X. Zhu, J. Li, Z. Liu, S. Wang, and F. Yang, "Learning transportation annotated mobility profiles from GPS data for context-aware mobile services," in *2016 IEEE International Conference on Services Computing (SCC)*. IEEE, 2016, pp. 475–482.

[67] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, HI, USA, 2017, pp. 11–19.

[68] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 1742–1750.

[69] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[70] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yu, "FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation," *arXiv preprint arXiv:1903.11816*, 2019.

[71] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 8856–8865.

[72] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, "Gated-SCNN: Gated shape CNNs for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, Seoul, Korea, 2019, pp. 5229–5238.

[73] Z. Xiao, Y. Wang, K. Fu, and F. Wu, "Identifying different transportation modes from trajectory data using tree-based ensemble classifiers," *ISPRS International Journal of Geo-Information*, vol. 6, no. 2, p. 57, 2017.

[74] M. Guo, S. Liang, L. Zhao, and P. Wang, "Transportation mode recognition with deep forest based on GPS data," *IEEE Access*, vol. 8, pp. 150 891–150 901, 2020.

[75] R. C. Shah, C.-y. Wan, H. Lu, and L. Nachman, "Classifying the mode of transportation on mobile phones using GIS information," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Seattle, WA, USA, 2014, pp. 225–229.

[76] M. Simoncini, F. Sambo, L. Taccari, L. Bravi, S. Salti, and A. Lori, "Vehicle classification from low frequency GPS data," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2016, pp. 1159–1166.

[77] P. A. Gonzalez, J. S. Weinstein, S. J. Barbeau, M. A. Labrador, P. L. Winters, N. L. Georggi, and R. Perez, "Automating mode detection for travel behaviour analysis by using global positioning systems-enabled mobile phones and neural networks," *IET Intelligent Transport Systems*, vol. 4, no. 1, pp. 37–49, 2010.

[78] H. Mäenpää, A. Lobov, and J. L. M. Lastra, "Travel mode estimation for multi-modal journey planner," *Transportation Research Part C: Emerging Technologies*, vol. 82, pp. 273–289, 2017.

[79] A. Yazdizadeh, Z. Patterson, and B. Farooq, "Ensemble convolutional neural networks for mode inference in smartphone travel survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2232–2239,

2019.

[80] X. Song, H. Kanasugi, and R. Shibasaki, "DeepTransport: Prediction and simulation of human mobility and transportation mode at a citywide level," in *International Joint Conference on Artificial Intelligence*, vol. 16, New York, NY, USA, 2016, pp. 2618–2624.

[81] D. J. Patterson, L. Liao, D. Fox, and H. Kautz, "Inferring high-level behavior from low-level sensors," in *International Conference on Ubiquitous Computing*. Seattle, WA, USA: Springer, 2003, pp. 73–89.

[82] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *International Joint Conferences on Artificial Intelligence*, 2017, pp. 1753–1759.

[83] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International Conference on Neural Information Processing*, Guangzhou, China, 2017, pp. 373–382.

[84] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 3861–3870.

[85] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, 2017, pp. 5574–5584.

[86] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4471–4480.

[87] R. Gu, G. Wang, and J.-N. Hwang, "Exploring severe occlusion: Multi-person 3D pose estimation with gated convolution," *arXiv preprint arXiv:2011.00184*, 2020.

[88] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[89] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 85–100.

[90] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations*, 2018.

[91] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[92] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey review*, vol. 23, no. 176, pp. 88–93, 1975.

[93] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 7132–7141.

[94] C. Markos, J. J. Q. Yu, and R. Y. D. Xu, "Capturing uncertainty in unsupervised GPS trajectory segmentation using Bayesian deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 390–398.

[95] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, 2020.

[96] F. D. S. Ribeiro, F. Calivá, M. Swainson, K. Gudmundsson, G. Leontidis, and S. Kollias, "Deep Bayesian self-training," *Neural Computing and Applications*, pp. 1–17, 2019.

[97] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[98] C. Markos and J. J. Q. Yu, "Unsupervised deep learning for GPS-based transportation mode identification," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[99] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Iberoamerican Congress on Pattern Recognition*, Havana, Cuba, 2013, pp. 117–124.