Unsupervised Deep Learning for GPS-Based Transportation Mode Identification

Christos Markos[1], and James J.Q. Yu[2], *Member, IEEE*

***Abstract*— Intelligent transportation management requires not only statistical information on users' mobility patterns, but also knowledge of their selected transportation modes. The latter can be inferred from users' GPS records, as captured by smartphone or vehicle sensors. The recently demonstrated prevalence of deep neural networks in learning from data makes them a promising candidate for transportation mode identification. However, the massive geospatial data produced by GPS sensors are typically unlabeled. To address this problem, we propose an unsupervised learning approach for transportation mode identification. Specifically, we first pretrain a deep Convolutional AutoEncoder (CAE) using unlabeled fixed-size trajectory segments. Then, we attach a clustering layer to the CAE's embedding layer, the former maintaining cluster centroids as trainable weights. Finally, we retrain the composite clustering model, encouraging the encoder's learned representation of the input data to be clustering-friendly by striking a balance between the model's reconstruction and clustering losses. By further incorporating features computed over each segment, we achieve a clustering accuracy of 80.5% on the Geolife dataset without using any labels. To the best of our knowledge, this is the first work to leverage unsupervised deep learning for clustering of GPS trajectory data by transportation mode.**

I. Introduction

Transportation mode identification is the task of inferring travelers' modes of transportation from their mobility data. Such knowledge can help location-based services provide users with accurate, personalized information based on their real-time location [1]. Examples of such services include posting targeted advertisements on electronic billboards, or notifying a user when to start their trip in order to reach their destination on time [1]. On a larger scale, city-wide knowledge of users' mobility and transportation mode patterns can be leveraged by intelligent transportation systems to improve traffic management through travel demand analysis, route recommendations, and transportation planning [2].

The first step towards transportation mode identification is acquiring users' mobility data. Global Positioning System (GPS) sensors are mobile sensors that capture ordered sequences of timestamped latitude-longitude pairs. Their presence in most modern smartphones and vehicles offers the benefit of user proximity, in turn allowing for much broader path coverage than traditional fixed-point sensors. Nonetheless, inferring users' transportation modes based solely on GPS trajectory data is a challenging problem, as GPS sensors can only record spatiotemporal characteristics of user movement without any explicit information on the utilized travel modes. Moreover, users may be reluctant to annotate their trips with such information due to privacy concerns or lack of motivation.

To build a model that can extract useful knowledge from the available GPS data, earlier work relied extensively on intuition to transform raw trajectory data into suitable representations for training machine learning models. Several studies have used Bayesian networks [3], decision trees [4], [5], random forests [6], and support vector machines [7], to name a few. Such approaches are limited not only by the significant domain expertise needed, but also the fact that fundamental features such as relative distance and velocity are prone to GPS measurement errors and traffic or environmental conditions [5]. The recent state-of-the-art performance of deep learning models in challenging fields such as computer vision [8] and natural language processing [9] has consequently attracted significant interest in the field of transportation mode identification. Researchers have successfully used multilayer perceptrons [10], recurrent neural networks [11], and convolutional neural networks [12], [13], among others. Still, the literature has not addressed the problem of inferring transportation modes from strictly unlabeled GPS data.

To address this problem, and given that traditional clustering algorithms such as K-means do not perform well with high-dimensional data [14], unsupervised deep learning is considered as a promising solution. This involves training deep neural networks without the guidance of any labeled samples. For instance, autoencoders [15] are typically trained to encode their input in such a way that it can then be reconstructed at their output layer with low error. By definition, however, autoencoders merely attempt to minimize the reconstruction error between the input and output data. This does not necessarily mean that the trained encoder will always produce similar embeddings for similar inputs, i.e., clustering-friendly representations [16], [17].

As such, a recent body of literature has proposed several deep clustering methods using deep neural networks. Some studies pretrain an autoencoder on the input data and then cluster the lower-dimensional learned embeddings by only optimizing the clustering loss. In [18], agglomerative clustering is applied to autoencoder-learned features. Another seminal work [19] first pretrains a denoising autoencoder

This work is supported in part by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032 and in part by Guangdong Provincial Key Laboratory No. 2020B121201001. *(Corresponding author: James J.Q. Yu.)*

The authors are with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Christos Markos is also with the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia.

and then replaces the decoder with a custom clustering layer. Instead, [14] replaces it with a soft K-means layer and retrains the model using samples of increasing clustering difficulty. More recent approaches have reported better results by jointly optimizing the clustering and autoencoder reconstruction losses. In contrast to [19], the decoder is preserved in [20] to reduce distortion of the learned feature space, while [17] instead computes the clustering loss using the mean squared error and performs separate updates to the network parameters, clusters, and centroids. DeepCluster [21] iteratively groups the neural network's output using K-means and sets the resulting cluster assignments as training labels to update its parameters. While these approaches have reported promising results on image and text data, as far as we are concerned, deep clustering approaches have not yet been applied to GPS trajectory data for transportation mode identification.

In this work, we propose unsupervised deep learning for GPS-based transportation mode identification. To this end, we first use fixed-size trajectory segments to pretrain a deep Convolutional AutoEncoder (CAE). We then attach a clustering layer to the CAE's embedding layer, the former maintaining cluster centroids as trainable weights, as proposed by [20]. Finally, we retrain the composite clustering model, encouraging the learned representation of the input data to be clustering-friendly by striking a balance between the model's reconstruction and clustering losses. To provide the model with additional information, we also incorporate features corresponding to mobility-related statistics computed over each segment. We report a clustering accuracy of 80.5% on the Geolife dataset [4], [5] without using any labels. To the best of our knowledge, this is among the pioneer efforts to cluster GPS trajectory data by transportation mode using unsupervised deep learning.

The rest of this paper is structured as follows. Section II contains the problem formulation and introduces the Geolife dataset. Section III describes the proposed methodology, including the proposed data preprocessing techniques, the *global* features which we extracted to improve clustering performance, and the clustering model components. Section IV presents the simulation setup and experimental results on two case studies, and finally, Section V concludes this paper.

II. Problem Formulation

Before delving into the specifics of the proposed methodology, we first establish the necessary definitions and formulate the problem of clustering GPS trajectory segments by transportation mode. We then introduce Geolife, the dataset that we used to validate our approach.

A. Clustering GPS Trajectory Segments

We consider a set of GPS trajectories T, where each $T_i \in T$ is a sequence $T_i = \{p_j\}_1^{L_i}$ of length L_i and p_j is a GPS point, defined as a tuple of timestamp, latitude, and longitude. A trajectory can be split into multiple trips, during which a traveler may use several modes of transportation. In order to cluster by transportation mode, we must further divide trips into segments such that each segment involves only a single transportation mode. For each T_i, we therefore extract the set of nonoverlapping same-mode segments $S_i = \{s_k\}_1^N$ composed of N subsequences $s_i = \{p_k\}^{L_{s_i}} \in T_i$ with $L_{s_i} \leq L_i$. We note that, in the absence of labels, trip segmentation is a nontrivial task. Such algorithms have been proposed in the literature; for instance, [12] used a discrete optimization algorithm to detect transportation mode changes along GPS trajectories with high fidelity. Nonetheless, as unsupervised trip segmentation is not the focus of our work, we leverage the available labels to divide trips into segments of the same transportation mode as in [4].

Using latitude and longitude pairs as features to train a clustering model could be problematic. Intuitively, the clustering model would have to be retrained whenever a user moved to a location that the model had not seen before. To this end, we instead compute *local*, point-level motion features including velocity, acceleration, and jerk for each GPS point p_j in a segment. We also capture *global*, segment-level features such as average velocity and stop rate for each segment s_i, as will be described in Section III. We thereby process T into a dataset $S = \{s_i\}_1^n$ of n same-mode segments, with each point in a segment being described by a vector of motion features.

Given S, we view the problem of clustering each segment into $K < n$ clusters through the scope of optimizing a clustering objective function. Specifically, the objective is to partition S into K disjoint clusters $C = \{c_i\}_1^K$ such that the probability distance between a "true" distribution of encoded segments (over C) and an estimated auxiliary distribution are minimized. We refer the reader to Section III for a detailed explanation of the proposed approach.

B. Dataset

This study is evaluated on the Geolife dataset by Microsoft Asia Research [4], [5], which has been widely used for GPS-based trajectory research. It contains 17,621 GPS trajectories by 182 users obtained over five years, the majority of which were collected in Beijing, China at a sampling rate of $1-5$ seconds. Out of those users, only 69 have labeled parts of their trajectories by transportation mode.

The main transportation modes include walking, bike, bus, car, taxi, train, subway and airplane. Following the dataset authors' recommendations, we treat cars and taxis as a single class, "driving", and trains and subways as "train". As typically done in the literature [4], [5], [12], we only retain the classes for which there are sufficient samples, namely "walk", "bike", "bus", "driving", and "train". Note that labels are only used for trip segmentation and clustering evaluation.

III. Proposed Methodology

In this section, we first detail our data preprocessing techniques, including feature extraction, outlier removal, feature normalization, and segmentation into fixed-size chunks suitable for training the proposed CAE. We then introduce *global*, segment-level features, and describe the CAE used to learn a lower-dimensional representation of the input data. Finally, we specify the clustering layer as well as the loss functions that integrate it with the CAE to produce the composite clustering model.

A. Data Preprocessing and Local Features

The Geolife dataset consists of timestamped GPS trajectories, i.e., tuples of dates and times, latitudes, and longitudes. These trajectories contain various trips and transportation modes. As such, we first perform trip segmentation, extracting a new segment whenever the transportation mode changes or the time elapsed between two consecutive points within a trip exceeds 20 minutes. This particular time threshold was first proposed in the seminal paper of [4], and has since been used by numerous subsequent works; see [5], [12] for some examples.

Following trip segmentation, we extract motion features using the raw GPS data at hand. We first compute the geodesic distance and elapsed time between each pair of GPS points. These two features enable the computation of higher-order distance derivatives, such as velocity, acceleration, and jerk. Given a GPS point p_i and its successor p_{i+1}, we estimate the former's relative distance Δx_{p_i}, elapsed time Δt_{p_i}, velocity V_{p_i}, acceleration A_{p_i}, and jerk J_{p_i} as follows:

$$\Delta x_{p_i} = \text{Geodesic}(p_i[lat, lon], p_{i+1}[lat, lon]), \tag{1}$$

$$\Delta t_{p_i} = p_{i+1}[t] - p_i[t], \tag{2}$$

$$V_{p_i} = \frac{\Delta x_{p_i}}{\Delta t_{p_i}}, \tag{3}$$

$$A_{p_i} = \frac{V_{p_{i+1}} - V_{p_i}}{\Delta t_{p_i}}, \tag{4}$$

$$J_{p_i} = \frac{A_{p_{i+1}} - A_{p_i}}{\Delta t_{p_i}}. \tag{5}$$

In this study, we empirically select velocity, acceleration, and jerk as the time series features to train our clustering model, following previous transportation mode identification literature [12], [13].

Next, we discard unrealistic GPS points according to certain criteria, some of which were inspired by [12]. Specifically, we remove a GPS point if any of the following are true:

- Its latitude or longitude does not fall within valid ranges, i.e., [-90, 90] and [-180, 180] degrees, respectively;
- Its timestamp is greater than that of the next point;
- Its velocity or acceleration exceeds reasonable thresholds [12] given its corresponding transportation mode.

We then remove any points whose velocity exceeds the 99th percentile, or whose acceleration or jerk do not fall between the 1st and 99th percentiles. Since the distribution of velocities is highly skewed, we transform it using the cubic root. Next, we standardize all three features to zero mean and unit variance.

At this point, the resulting trip segments vary significantly in number of points and length, with many segments spanning several kilometers. However, our CAE requires fixed-size inputs of shape $(1, n_{\text{P}}, n_{\text{F}})$, where n_{P} is the number of points per segment and n_{F} is the number of features per point. In this study, we have empirically selected n_{P} to be 128. Therefore, we divide all segments into nonoverlapping chunks of length 128, discarding any segment whose total number of GPS points is less than that. As opposed to [12], we make no further assumptions such as minimum total duration or distance.

B. Global Features

In addition to the *local* features described in the previous subsection, we also compute four velocity- and acceleration-based features over the entirety of each segment. Intuitively, these *global* features might provide our CAE with useful hints during training.

For each segment s_i, we compute the average velocity AV_{s_i}, expectation of velocity EV_{s_i}, and maximum velocity MV_{s_i}. Inspired by [5], we also select the stop rate SR_{s_i}, which they define as the number of GPS points whose velocity is less than a threshold divided by the total segment distance. However, we instead compute it as:

$$\text{SR}_{s_i} = |P_s| / n_{\text{P}}, \tag{6}$$

where $P_s = \{p_i | p_i \in s_i, V_{p_i} < 3\text{m/s}\}$ and n_{P} is the number of GPS points per segment. Finally, to use these *global* features with our time series features, we repeat the former 128 times and append them to the latter for a total of seven features.

C. Convolutional Autoencoder

An autoencoder is a neural network that aims to reconstruct its input under some constraint that promotes the extraction of useful latent representations, or embeddings. Without loss of generality, it consists of an encoder that maps the input x_i to its latent representation h_i, and a decoder that outputs $\hat{x}_i$ by attempting to reconstruct x_i from h_i. CAEs are a class of autoencoders that process the data using convolution layers. If the embedding layer's dimensionality is lower than that of the input layer, the autoencoder is termed "undercomplete".

In this study, we first pretrain a fully convolutional, undercomplete autoencoder to learn an initial lower-dimensional representation of the input data. The term "fully convolutional" refers to not using a densely-connected embedding layer, which has been adopted by other CAE-based works [17], [19], [20]; instead, we make appropriate use of 1×1 convolutions and reshape layers. The encoder and decoder are symmetrical: the former consists of consecutive pairs of convolution and max pooling layers, while the latter uses deconvolution and upsampling layers, respectively. The only exception is the encoder's final convolution layer, i.e., the embedding layer, which is not followed by a max pooling layer. We use convolutions and deconvolutions configured such that their outputs have the same length as their inputs. Except for the final layer, each convolution and deconvolution layer is followed by a batch normalization layer to facilitate model training.

During pretraining, we attempt to minimize the CAE's reconstruction loss L_r, quantified as the mean squared error between the original input s_i and the reconstructed input $\hat{s}_i$:

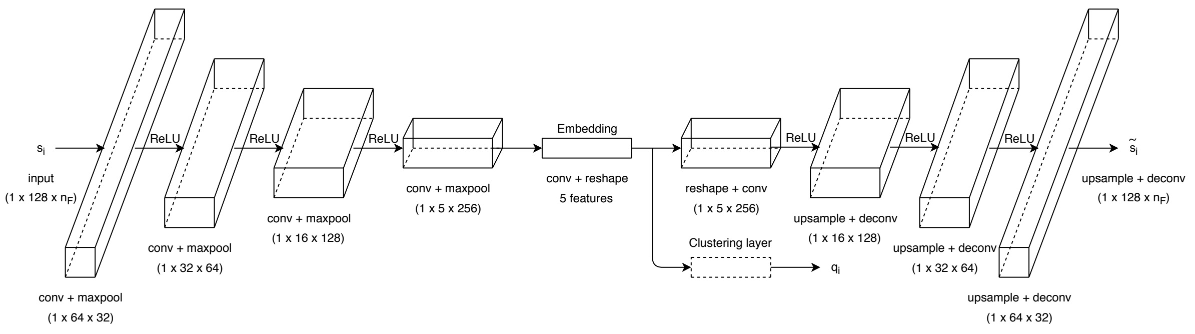

Fig. 1. The composite clustering model.

$$L_r = \frac{1}{n}\sum_{1}^{n}(s_i - \hat{s_i})^2. \tag{7}$$

D. Clustering Layer

After pretraining the CAE, we attach a clustering layer to its embedding layer as in [20]. The clustering layer contains trainable weights $\{\mu_j\}_1^K$, where K is equal to the number of desired clusters and μ_j corresponds to the coordinates of the j-th centroid. In this study, K is set to 5.

On the forward pass, the clustering layer uses Student's t-distribution to assign cluster membership probabilities q_{ij} to all embedded samples z_i in the current batch as follows:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}}. \tag{8}$$

Having computed q_{ij}, we obtain soft labels q_i as the indices of the maximum probabilities, and estimate the normalized target distribution P as:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})}. \tag{9}$$

On the backward pass, we cluster the embedded samples z_i by minimizing the Kullback-Leibler (KL) divergence between P and Q:

$$L_c = KL(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{10}$$

E. Composite Clustering Model

We initialize the composite clustering model, illustrated in Fig. 1, with a pretrained CAE whose embedding layer is connected to a clustering layer, as described in Sections III-C and III-D. We then train the composite clustering model by minimizing the following objective function:

$$L = L_r + \gamma L_c, \tag{11}$$

where hyperparameter γ controls the influence of the clustering loss on the total loss. Setting $L_r = 0$, $\gamma = 1$ reduces the objective function to the one used in [19]. This effectively discards the CAE's decoder, potentially causing the clustering process to adversely perturb the learned embedding. Instead, [20] used $\gamma = 0.1$ to limit the strength of the clustering process, producing better clusters on the same datasets.

Another issue is how often to update the target distribution P; updating it too often could cause instability of the clustering process, while the opposite might entrap it in bad local minima. We use hyperparameter F, defined as a fraction of a single training epoch, to control the frequency of target distribution updates. For instance, $F = 0.5$ means that the target distribution is updated twice on every epoch, while $F = 2$ calls for an update once every two epochs.

IV. Performance Evaluation

This section first describes our simulation setup, including the configuration of model parameters and clustering hyperparameters, our choice of evaluation metrics, as well as the hardware on which the simulations were conducted. It then presents the results from two case studies; the first compares our clustering performance against established baselines, while the second tests the sensitivity of our proposed methodology to hyperparameter variations.

A. Simulation Setup

The CAE's encoder consists of five convolution layers with 32, 64, 128, 256, and 1 filter. The first three layers use a kernel size of 1×3 with "same" padding, while the fourth and fifth ones use a stride of 1×7 and 1×1 respectively, both with "valid" padding. All convolution and deconvolution layers are activated using the Rectified Linear Unit (ReLU) function, except for the last layers of the encoder and decoder which are linearly activated. The CAE is pretrained for 600 epochs using the Adam optimizer with an initial learning rate of 10^{-3} and default hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. After attaching the clustering layer, we retrain the composite model using a learning rate of 10^{-4}, instead. Unless otherwise stated, we use $\gamma = 20$, $F = 2$ when clustering on *local*, *global*, or all features. Training stops when fewer than 0.1% of samples are reassigned to a different cluster during an iteration.

Given the predicted labels $\hat{Y}$ and the ground-truth labels Y, we use the following metrics to evaluate clustering performance:

1) **Accuracy (ACC).** Following the definition in [16], clustering accuracy is defined as:

$$\mathrm{ACC}(Y, \hat{Y}) = \frac{1}{m}\sum_{1}^{m}\delta(y_i, \mathrm{Map}(\hat{y}_i)), \tag{12}$$

where m equals the number of samples in the dataset and $\delta(y, \hat{y})$ equals one if $y = \hat{y}$ or zero otherwise. $\mathrm{Map}(\hat{y}_i)$ is a function that maps a predicted label $\hat{y}_i$ to the equivalent label in the dataset. Using a linear assignment problem formulation, $\mathrm{Map}(\hat{y}_i)$ is then computed using the Hungarian algorithm [22].

TABLE I
CLUSTERING EVALUATION RESULTS

	Local features		Global features		All features	
	ACC	NMI	ACC	NMI	ACC	NMI
KM	54.4%	38.7%	73.1%	**58.9%**	61.8%	50.7%
SC	40.5%	30.4%	69.9%	58.5%	59.1%	51.2%
HAC	52.1%	36.1%	66.2%	57.3%	59.5%	50.5%
CAE + KM	64.2%	41.9%	65.9%	54.4%	77.4%	61.0%
CAE + SC	47.9%	36.8%	65.7%	54.0%	75.2%	60.3%
CAE + HAC	64.9%	42.5%	61.8%	53.4%	75.7%	59.3%
SECA [12]	62.9%	-	-	-	-	-
Proposed	**70.1%**	**47.9%**	**73.8%**	58.6%	**80.5%**	**64.4%**

2) **Normalized Mutual Information (NMI).** NMI is defined as:

$$\mathrm{NMI}(Y,\hat{Y}) = \frac{\mathrm{MI}(Y,\hat{Y})}{\max(H(Y),H(\hat{Y}))}, \qquad (13)$$

where $\mathrm{MI}(Y,\hat{Y})$ is the mutual information of Y and $\hat{Y}$, and $H(Y)$, $H(\hat{Y})$ correspond to the entropies of Y and $\hat{Y}$, respectively.

All experiments were conducted on a server with an NVIDIA GeForce RTX 2080Ti GPU and an Intel Xeon Silver 4210 CPU clocked at 2.20GHz. The reported values for ACC or NMI refer to the averaged results obtained over five executions.

B. Clustering Evaluation

To demonstrate the effectiveness of integrating *local* and *global* features, we evaluate clustering performance in three scenarios, i.e., using (i) *local* features, (ii) *global* features, and (iii) the combination thereof. We compare our results with those of seven baselines, which are categorized as follows:

- **Traditional clustering on original features.** We select the widely used (1) K-Means (KM), (2) Spectral Clustering (SC), and (3) Hierarchical Agglomerative Clustering (HAC) algorithms, as implemented in Python's "scikit-learn" machine learning library. Since these implementations are not designed for data of more than two dimensions, we take the mean feature values per data sample when using either the *local* features or their combination with the *global* ones.
- **Traditional clustering on embedded features.** While the algorithms in the previous category cluster the original samples s_i, here we instead use the CAE-embedded samples z_i for K-means, spectral, and hierarchical agglomerative clustering. For clarity, we refer to these cases as (4) "CAE + KM", (5) "CAE + SC", and (6) "CAE + HAC".
- **Semi-supervised classification.** Since no other work in the literature has performed unsupervised transportation mode identification, we evaluate our proposed framework against the (7) SEmi-supervised Convolutional Autoencoder (SECA) [12], which used as little as 10% of the ground-truth labels for semi-supervised classification instead.

We report our experimental results in Table I. The proposed framework achieved significantly better results than the traditional clustering algorithms when applied to *local* or all features. In addition, it considerably outperformed SECA, which has an accuracy of 62.9% using not only 10% of the available ground-truth labels but also trajectory segments of nearly twice the size (248) as ours (128). It is evident that combining *local*, point-level features with *global*, segment-level features resulted in substantial improvements for all evaluated methods.

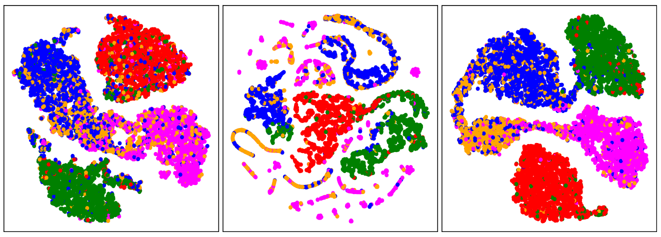
Fig. 2. The five clusters produced by the proposed framework for local, global, and all features, respectively. We use red, green, blue, orange, and magenta to denote "walk", "bike", "bus", "driving", and "train" classes.

Fig. 2 plots a two-dimensional visualization of the clusters generated by the proposed framework. We note that with *local* features, all clusters except for the one corresponding to "driving" were generally well separated. Crucially, although using only *global* features resulted in better accuracy than using only *local* ones, the clusters in the former case were highly irregular. This highlights the proposed benefit of combining *local* and *global* features, which produced both the highest accuracy and the best defined clusters as shown in the third subplot of Fig. 2.

Unsurprisingly, the traditional clustering algorithms performed worst when applied to the original data with *local* or all features. This can be attributed in part to the features being averaged over all 128 timesteps for each sample in order to have the required dimensionality. On the other hand, the same algorithms achieved better results when applied to the CAE-learned embeddings of *local* or all features. They also did much better on the original data when using *global* features, with KM nearly matching the accuracy of our framework; intuitively, the *global* features' low dimensionality should be a better fit for the traditional clustering algorithms.

C. Hyperparameter Sensitivity

As mentioned in Section III, the lack of ground-truth labels in unsupervised learning makes it difficult to tune the hyperparameters of any clustering model. Hence, such a model should ideally be robust to a reasonable range of hyperparameter variations.

To assess the proposed model's sensitivity to hyperparameters γ and F, we first empirically select a value that works well enough for F, i.e., $F = 2$, then test values for γ ranging from 0.1 to 100. Table II shows the corresponding clustering accuracy results. We note that the proposed clustering framework appears to be somewhat sensitive to values of $\gamma < 1$ when trained on *local* features and $\gamma > 20$ on *global* ones. In contrast, it appears to be more robust to variations of γ when trained on all features.

Next, we set $\gamma = 20$ and evaluate the clustering model as F iterates through the previous range of values. According to

TABLE II

SENSITIVITY OF ACCURACY TO STRENGTH OF CLUSTERING LOSS

Features	γ									
	0.1	0.2	0.5	1	2	5	10	20	50	100
Local	65.7%	66.8%	67.6%	68.1%	68.4%	69.0%	68.9%	70.1%	69.8%	69.9%
Global	70.8%	71.4%	69.3%	71.8%	69.2%	70.0%	72.1%	73.8%	69.6%	68.0%
All	79.6%	79.4%	78.6%	78.6%	79.2%	78.5%	79.1%	**80.5%**	79.0%	78.3%

TABLE III

SENSITIVITY OF ACCURACY TO FREQUENCY OF TARGET DISTRIBUTION UPDATES

Features	F									
	0.1	0.2	0.5	1	2	5	10	20	50	100
Local	51.5%	53.5%	62.6%	69.3%	70.1%	69.4%	69.9%	70.0%	70.0%	69.9%
Global	64.8%	65.0%	66.5%	68.5%	73.8%	71.9%	70.6%	72.1%	72.6%	72.3%
All	76.9%	75.6%	79.9%	78.8%	**80.5%**	78.3%	80.2%	**80.5%**	80.4%	80.3%

the results shown in Table III, it seems that values of $F < 1$ and $F < 2$ result in significant instability when training on *local* or *global* features, respectively. Albeit on a smaller scale, $F < 0.5$ also impacts performance when using all features. Nonetheless, we note that our clustering model is most robust to variations of F when trained on all features.

V. CONCLUSION

In this paper, we proposed an unsupervised deep learning approach to clustering GPS trajectories by transportation mode. Using fixed-size trajectory segments, we first pre-trained a deep CAE, and then attached a clustering layer to its embedding layer, the former having cluster centroid coordinates as trainable weights. We finally retrained the composite model by jointly optimizing the reconstruction and clustering losses. The proposed framework achieved a clustering accuracy of 70.1% on Geolife using only *local*, time series features, significantly outperforming both traditional clustering algorithms and the semi-supervised state-of-the-art in transportation mode identification. When combining the *local* features with *global* features averaged over each trajectory segment, the proposed framework achieved an even higher accuracy of 80.5%. Our hyperparameter sensitivity tests also showed that the proposed framework is not overly sensitive to the strength of the clustering loss over the reconstruction loss or the frequency of target distribution updates. In future work, we will investigate methods for unsupervised trip segmentation and real-time, progressive transportation mode identification.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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, 2011, pp. 54–63.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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*, in press.

[13] 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.

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

[15] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[16] 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.

[17] 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.

[18] 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, 2016, pp. 5147–5156.

[19] 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, 2016, pp. 478–487.

[20] 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.

[21] 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.

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