A Deep Learning Approach for Identifying User Communities Based on Geographical Preferences and Its Applications to Urban and Environmental Planning

DANIELLE L. FERREIRA*, University of California Santa Cruz (UCSC), USA
BRUNO A. A. NUNES, University of California San Francisco (UCSF), USA
CARLOS ALBERTO V. CAMPOS, Federal University of State of Rio de Janeiro (UNIRIO), Brazil
KATIA OBRACZKA, University of California Santa Cruz (UCSC), USA

Understanding human mobility plays a vital role in urban and environmental planning as cities continue to grow. Ubiquitous geo-location, localization technology and availability of bigdata-ready computing infrastructure have enabled the development of more sophisticated models to characterize human mobility in urban areas. In this work, our main goal is to extract spatio-temporal features that characterize user mobility and, based on the similarity of these features, identify user *communities*. To this end, we propose a novel approach that leverages image processing techniques to represent user geographical preferences as images and then apply deep convolutional autoencoders to extract latent spatio-temporal mobility features from these images. These features are then fed to a clustering algorithm, which identifies the underlying community structures. We use a diverse urban mobility datasets to validate the proposed framework. Our results show that the proposed framework is able to significantly increase the similarity between intra-community nodes (by up to 107%) as well as dissimilarity between inter-community nodes (up to 54%) when compared against no pre-processing of the datasets, i.e without pre-processing the datasets through any feature fusion method. Moreover, it was also able to reach up to 100% improvement when compared against community identification using Principal Component Analysis (PCA). Our results also show that the proposed approach yields significant increase in contact time amongst users belonging to the same community, by up to 80% when compared to the average contact time when not considering community structures, and by up to 150% when compared to the baseline. To the best of our knowledge, our proposal is the first to consider deep convolutional autoencoding to perform automatic extraction of non-linear spatio-temporal mobility features characterizing individual users from raw mobility datasets with the goal of identifying user communities.

CCS Concepts: • **Computing methodologies** → **Unsupervised learning**; • **Information systems** → **Mobile information processing systems**; *Spatial-temporal systems.*

ACM Reference Format:
Danielle L. Ferreira, Bruno A. A. Nunes, Carlos Alberto V. Campos, and Katia Obraczka. 2020. A Deep Learning Approach for Identifying User Communities Based on Geographical Preferences and Its Applications to Urban and Environmental Planning. *ACM Trans. Spatial Algorithms Syst.* 1, 1 (August 2020), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Current affiliation: Departament of Cardiology, University of California San Francisco (UCSF).

Authors' addresses: Danielle L. Ferreira, danielle.ferreira@uniriotec.br, University of California Santa Cruz (UCSC), Santa Cruz, USA; Bruno A. A. Nunes, bruno.astutoarouchenunes@ucsf.edu.br, University of California San Francisco (UCSF), San Francisco, USA; Carlos Alberto V. Campos, beto@uniriotec.br, Federal University of State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil; Katia Obraczka, katia@soe.ucsc.edu, University of California Santa Cruz (UCSC), Santa Cruz, USA.

2374-0353/2020/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Currently, about fifty percent of the world's population lives in urban areas and the forecast is that by 2050 this percentage will grow to approximately seventy percent [10]. As such, the greatest wave of city migration is yet to come and together with it a wide range of challenges raised by the need to improve the style and quality of life of a growing urban population. According to [10], a better understanding of city dynamics would allow for improved services as well as minimized environmental impact resulting from urban expansion.

Urban mobility, defined as the displacement of people across an urban region over time [9], is critical to understand the dynamics of an urban center. As cities grow, the complexity of urban transportation and transit systems and the time people spend in transit will greatly increase. As a result, expanded- and new transportation services will be required demanding deeper investigation into urban mobility [3, 43]. Additionally, understanding human mobility in urban areas is crucial to other city management and planning applications such as public health, emergency response, education, entertainment, shopping, etc [29].

As computational resources become more widely available through cloud- and edge computing services, machine learning techniques, such as neural networks (NNs), which not too long ago were considered totally prohibitive in terms of their computational demands, have now become mainstream tools to handle the enormous amounts of data being generated by sensing devices embedded mostly everywhere. A special category of NNs named *deep convolutional autoencoders* has been used as an efficient and effective image processing technique and have, more recently, been applied in a variety of other domains, ranging from data augmentation, de-noising, activity and speed recognition, computer vision, to name a few [40].

In this work, we propose a framework to identify user communities based on user's spatio-temporal geographical preferences. The proposed framework employs a deep convolutional autoencoder (CAE) architecture to learn latent spatio-temporal mobility features from an image-based representation of user mobility traces recorded in a variety of urban scenarios. From these mobility datasets, the proposed framework is able to identify geographical preference similarities among users and in turn group users into *communities* based on such preferences.

To validate and evaluate our CAE framework, we use a diverse set of publicly available mobility traces and show that the proposed framework is able to significantly increase similarity between intra-community nodes (by up to 107% for the datasets used) as well as dissimilarity between inter-community nodes (up to 54% for our datasets) when compared with our baseline, i.e, without pre-processing the datasets through any feature fusion method. Moreover, it was also able to reach up to 100% improvement when compared with an alternate community extraction approach that uses Principal Component Analysis (PCA). Besides the degree of similarity amongst users in the same communities and dissimilarities amongst users in different communities, we also evaluate contact times between users, which are an important metric for opportunistic networks. Our experiments show that our CAE approach yields a significant increase in contact times between users belonging to the same community (for the datasets considered, by up to 80% when compared to the average contact time when not considering community structures and by up to 150% when compared to user communities extracted from the baseline). Additionally, our approach also increases contact time between members of the same community (from 10% up to 125% for our datasets) when compared to PCA.

While autoencoders have been widely used to predict future mobility trajectories and traffic conditions, to the best of our knowledge, our work is the first to consider deep autoencoder NNs to perform automatic extraction of non-linear spatio-temporal mobility features from real mobility

datasets ultimately achieving improved user community identification. As such, we can summarize the contributions of this paper as follows:

- We propose an image-based representation of spatio-temporal user mobility features extracted from mobility traces. Such mobility records contain only a time series of user locations (either GPS coordinates or Access Point associations/disassociations) that capture user mobility in a variety of scenarios, including different urban settings (such as downtown areas, University campuses, etc) that incorporate a variety of modes of transportation, including private vehicles, buses, taxis, pedestrians, and bikes.
- We develop a deep convolution autoencoder (CAE) based approach to perform automatic spatio-temporal mobility features extraction contained in the images representing user mobility features.
- We demonstrate that our approach to automatically extract user mobility features by employing the proposed image-based method and deep convolutional learning architecture can be used as input to clustering algorithms for identifying communities that group users according to similar spatial and temporal mobility patterns.
- Through extensive experimentation using mobility records from a variety of scenarios, we show quantitative evidence that by using autoencoders' nonlinear feature fusion capability ahead of clustering can significantly increase the quality of the community structures identified when compared to linear feature fusion approaches such as PCA.
- Finally, we evaluate different autoencoder architectures, namely fully-connected, variational, and convolutional and discuss their impact on the performance of our user mobility based community identification framework.

The remainder of this paper is organized as follows. An overview of background and related work is discussed in Section 2. Our proposed approach is described in detail in Section 3. In Section 4, we present our experimental methodology, including the mobility datasets studied, as well as define the performance metrics used for evaluating and validating our proposal. Section 5 presents our experimental results and Section 6 discusses some applications of the proposed framework in the context of urban and environmental planning and management. Finally, Section 7 concludes the paper with some directions for future work.

2 BACKGROUND AND RELATED WORK

This section reviews related work on mobility characterization and presents a brief overview of autoencoders and our rationale for using them to extract features from raw user mobility datasets.

2.1 Mobility Characterization

In recent years, the wide availability of localization devices and techniques, such as the Global Positioning System (GPS), cellular base-station and Wi-Fi positioning systems have enabled human mobility data to be captured and recorded in a seamless and ubiquitous fashion. The availability of such positioning data not only enabled a variety of services and applications including road navigation, intelligent transportation systems, ride-sharing, etc, but also, motivated a large body of research on user mobility characterization and modeling. Below, we briefly describe some examples.

User mobility was found to be highly predictable and largely independent of the distance users cover on a regular basis [54], where most users visit the same places and share the same probability density function for places visited [25]. Additionally, the probability of a user to visit new locations or returning to previously visited locations follows a scaling law pattern, i.e., the probability of users visiting a new place decreases over time, while the chances of returning to places they frequently visit increases. Also, it is well known that members of the same social group also present similar

mobility behavior [22]. Moreover, human mobility exhibits strong non-linear dynamics and hence cannot be described by linear stochastic models [20].

The study of movement patterns has applications in a wide range of fields, such as urban planning [6, 48, 70], identifying similarities among individuals [53, 72], evaluating and proposing protocols for mobile networks [24, 30], public health management [11, 45], among others.

Machine learning has been used to identify patterns within large-scale, high-dimensional mobility data [56, 70]. Most of the work to-date uses supervised learning to map data instances to labels and predict new, unlabelled data. However, much of the data available from positioning technologies are not labeled (e.g., GPS records).

Most efforts based on unsupervised learning, on the other hand, use clustering algorithms to group instances that have similar behavior [33, 56, 69, 74]. Clustering has been extensively explored in machine learning and data analytics. It tries to organize data observations into groups with similar features, and its performance highly depends on the quality of the input data [4, 33, 47]. However, most of these efforts do not consider non-linear feature extraction, nor pre-processing or data transformations ahead of clustering the data. Unfortunately, the expressive power of linear features is very limited: they cannot be stacked to form deeper, more abstract representations since the composition of linear operations yields another linear operation [7].

Our work addresses this gap by using autoencoders to automatically learn from non-linear data representations, which can be stacked into deeper networks to better map input data to a feature space with improved data representation for clustering. As it will become clear from our experimental results, the ability to adequately represent non-linear features, as well as the pre-processing transformations on the raw positioning data are crucial steps towards extracting valuable and meaningful mobility features from available human mobility datasets. More recently, non-linear techniques to extract features from mobility datasets for trajectory prediction have been proposed [41]. For example, the problem of feature extraction for estimating users' transportation modes from their movement trajectories is addressed in [17], [23], [58], among others. In [73], and [44], mobility patterns are mined for urban traffic prediction and depressive states prediction [46].

The work described in [73] mines mobility patterns for trajectory-user linking (TUL). It takes into account the sparsity and high dimensionality of human trajectories and proposes a semi-supervised learning framework, which learns the human mobility in a neural generative architecture with stochastic latent variables that span hidden states in recurrent neural networks. TUL learns mobility patterns and classifies trajectories by users, i.e., it correlates unlabeled trajectories to the corresponding users in geotagged social media (GTSM) data, such as data generated by Instagram, Foursquare, and Twitter. However, social networks provide information about user location or interests with low granularity, since information is only recorded when users actually use the social network. For example, uploading images on Instagram, or check-in using Foursquare. Moreover, our study seeks to extract features of user mobility relying solely on traces containing time series of raw GPS coordinates or WiFi association/disassociation data, thus avoiding the need for information from social networks, cellular providers, etc.

Our work differs from efforts such as the ones outlined above as follows: (i) Our focus is on user community identification; and (ii) we rely solely on raw mobility traces containing time-series of user location.

2.2 Autoencoders

An autoencoder, or AE, is a neural network architecture designed to learn data encodings in an unsupervised fashion. As mentioned before, it is typically used for dimensionality reduction, where the complexity and variability of the data are reduced into an encoded, more compact

representation. Along with data reduction, there is also a reconstruction step that tries to reconstruct a representation as close as possible to the original input. In other words, the AE takes a set of unlabeled data $x \in R^n$ and tries to learn an approximation to the identity function to force the output to be as similar as possible to the input.

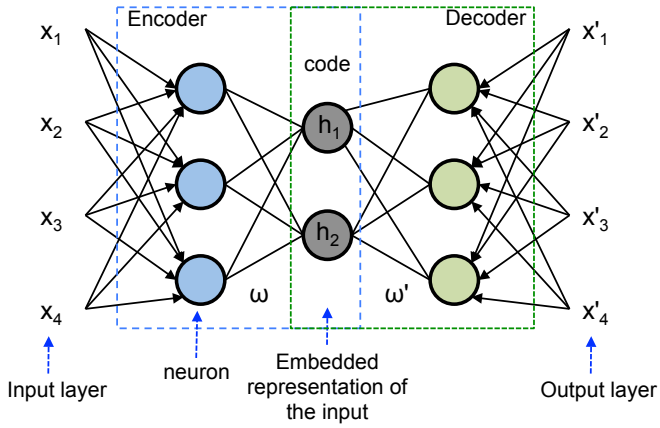


Fig. 1. Example of an autoencoder: it learns a representation h (*code*) of the Input x using a weight matrix ω, which is used to reconstruct x to obtain x'. The nodes in the network are also known as *neurons*.

The AE shown in Figure 1 consists of three basic components: (1) the encoder, which is the portion before the most compressed layer (or code) of the architecture. It compresses the input vector x into a latent representation h using a weight matrix ω; (2) the code, h, or latent space representation, is a lower-dimensionality representation of the input. This reduced representation allows us to discover interesting structures about the data; and (3) the decoder, the portion after the code, maps h back to the input, reconstructing it to obtain x' with another weight matrix ω'. Parameter optimizations are used to minimize the average reconstruction error between x and x'. Usually, the input and output layers have the same dimensionality. Note that the AE illustrated in Figure 1 has only a single encoder layer and a single decoder layer. However, using multiple encoding/decoding layers offers many advantages compared to shallow AEs. For instance, so-called deep AEs can decrease the amount of training data needed to learn some functions and yield better compression [26].

Training the network means learning the weight matrix ω' associated with all the neurons in the network, where a neuron, also knowns as node or unit, is the basic unit of computation in a neural network. During the training, each unit located in any layer in between the input and output layers, also called hidden layers, receives several inputs from the preceding layer. The unit computes the weighted sum of these inputs and eventually applies an activation function to produce the output. The most popular activation functions are *Linear, Logistic, ReLU, SELU,* and *Tanh*. The non-linear behavior of neural networks comes from the choice of these activation functions. After these steps, the output x' is compared to the input x, and the error will be propagated to every individual unit using the back-propagation algorithm [37]. Finally, each weight's contribution to the error is calculated and a *loss function* is used to adjust the parameters at each layer (i.e., update the weights). A typical loss function is the mean squared error or cross-entropy when input values are binary or modeled as bits.

Different types of AEs have been proposed. The one represented in Figure 1 is called *fully connected* since its neurons form a fully-connected network. *Variational autoencoders*, or VAEs, are another type of AE. The main difference from traditional AEs is that VAEs learn the parameters of a probability distribution to model the data, while traditional AEs learn an arbitrary function to encode and decode the input. In VAEs, the constraint of the activations in the hidden units over the whole data should be drawn from the standard Gaussian distribution, i.e., zero mean and unit variance in each direction. After learning the probability distribution, the parameters are sampled from it and then the encoder network generates samples closely resembling the training data. Two loss functions are simultaneously optimized to train the model weights, namely a *reconstruction loss function* and the Kullback-Leibler divergence between the learned latent distribution and a prior unit Gaussian.

Another type of AE, which has been widely used for image processing tasks is called *convolutional autoencoders*, or CAEs. CAEs are designed to process data inputs in the form of multidimensional arrays, e.g. images composed of 2D arrays containing pixel intensities represented in different color channels. Some examples of data inputs are 1D signals for time series, 2D for images, and 3D for video or 3D images. CAEs use the same principles as traditional autoencoders discussed above, but instead of fully-connected layers, it contains convolutional layers in the encoder part and deconvolution layers in the decoder part as discussed in more detail below.

CAE architectures are structured in several stages of convolutional and pooling layers. The units in a CAE are organized in layers called *feature maps*, also known as *convolutional filters or kernels* that are connected through a set of weights. Again a non-linear activation function, such as ReLU, is passed through the weights. In this way, CAEs are able to detect local groups of values in an array of images that are often highly correlated, and also detect spatial invariance patterns. In other words, if a pattern is identified in a part of an image, it could appear also in other parts. Hence, the convolutional layer is responsible for detecting patterns from the previous layer, and the pooling layer is responsible for merging semantically similar features into one. In CAEs, the pooling layer is responsible for reducing the dimensionality of the representation and creating an invariance to handle small shifts and distortions on the images. Usually, CAEs contain two or three stages of convolutional layers, non-linearity and pooling stacked, which may be followed by additional convolutional and/or fully-connected layers. The back-propagation algorithm is also used to training CAEs [37].

To date, the vast majority of applications of CAEs focus on image data. As discussed in Section 2.3 below, most work that use non-linear approaches for mobility characterization are based on either fully-connected or variational AEs. Our proposed approach is motivated by CAEs' well-known favorable cost-performance trade-offs. As such, an important contribution of this paper is to demonstrate the use of CAEs as an effective tool for extracting salient features from raw mobility records which are then used to automatically identify user communities that share common mobility characteristics.

2.3 Deep autoencoders for community identification

Deep learning (DL) models have been widely employed in recent years by researchers and practitioners to solve a plethora of different problems in many areas [7, 37, 40]. In the literature, it is possible to find DL architectures that fit specific purposes. For example, the use of U-Nets in medical imaging segmentation problems in [51], the use of Convolutional Neural Networks (CNNs) for semantic segmentation and image classification and recognition problems [35, 42, 59, 67], application of Fully Connected (FC) neural networks for regression and classification problems [50], Generative models for style transfer and data augmentation, and Recurrent Neural Networks (RNNs) applied to sequential and temporal data analysis [13, 14, 27, 66, 75], to name a few.

Most examples mentioned above are applied to supervised learning problems, where there are known labels or ground truth (GT) values that can be used to train neural networks (NNs). Once efficiently trained, these NNs should be able to identify such labels automatically in new, unseen samples in a generalizable fashion. While autoencoders (AE) are also an example of NN architectures, they can be applied to problems where a supervised approach is not possible, e.g., where there are no labels or GTs.

Identifying user community structures from raw mobility data requires unsupervised learning approaches since, most of the time, there is no previous knowledge from these raw records about the nature of the relationship between users (i.e., whether they belong to a certain community). Principal Component Analysis (PCA) [8] is a class of algorithms that has been widely used for unsupervised learning problems, especially for dimensionality reduction. PCA applies linear transformations to the data, rotating axis on the directions of where the data presents the higher variability. This way, it is possible to describe most of the variability in the data with only a few variables, instead of using the raw higher-dimensional data. Autoencoders (AEs) are another well-known class of algorithms that can be applied to unsupervised representation learning and has been used in a number of applications, such as pattern identification and dimensionality reduction [12]. One of our main reasons for applying AEs, instead of PCA, is the non-linear nature of the activations on the output of the AE layers [61]. In other words, AEs are able to capture non-linearities intrinsic to the mobility data that PCA cannot represent due to its linear nature.

Recent efforts propose the application of non-linear methods to extract community structures. Some examples include [15, 19, 28, 55, 65]. In [65] and [15], the problem of community detection is stated as modeling the probability that two vertices in an unweighted and undirected graph are connected. They also use deep autoencoder architectures, namely fully connected and variational autoencoders, respectively to learn a latent representation of such probability distributions for their datasets. Both works apply clustering algorithms on the latent variables to identify the communities. Even though these approaches can be applied to a number of different datasets, they are not suitable to be used on a raw mobility trajectory directly. In contrast to other types of data, extracting features from trajectory data is a non-trivial task [62]. Trajectory records can be highly unstructured and heterogeneous. They may have different sampling rates, number of users, lengths, sparsity/density. They may include different types of mobility patterns such as vehicular, bike, and pedestrian trajectories. Moreover, the types of features present and that can be extracted from this kind of datasets can also vary, e.g., initial- and final positions, speed, duration, type of user/device, social relationships. These features provide unique insights for exploring trajectories.

Adjacency matrices, like the ones used in [65] and [15], can be used to identify user communities based on the amount of time users spend together, or *contact time*. In our work, we group users solely based on individual user mobility patterns and geographical preferences, which is motivated by the fact that such mobility datasets can be more easily generated, and a large variety of them are publicly available and accessible. We should point out that, in our work, we compute contact times as a way to validate and compare user community identification approaches, instead of the very criterion to identify user communities. In other words, we use contact times between users in the same community as well as contact times between users in different communities to show how well the proposed approach is able to generate user community structures when compare to alternate techniques.

3 DEEP LEARNING ASSISTED USER COMMUNITY IDENTIFICATION

This section presents our proposed approach for extracting mobility patterns from raw GPS- and WiFi network datasets using convolutional autoencoding and clustering algorithms. Figure 2

summarizes the steps involved in the proposed framework and illustrates its key components, which will be described in detail below.

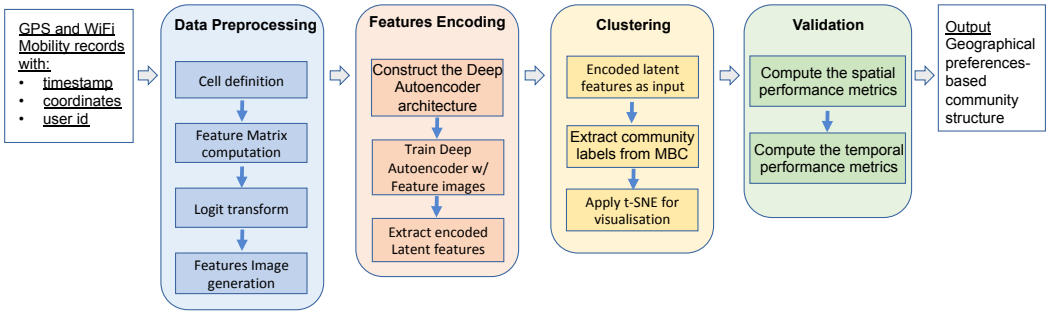


Fig. 2. Proposed user community structure identification framework illustrating all its components and data-flow.

3.1 Data pre-processing - converting mobility records into images

The first step in processing GPS- or WiFi mobility traces is to generate a matrix summarizing spatio-temporal mobility features extracted from the traces. In the space dimension, the region covered by the trace is divided into a grid of equally-sized *cells*. User trajectories are then considered as a sequence of cell positions.

In the time dimension, the temporal resolution depends on the sampling resolution of GPS devices or WiFi devices. We should point out that mobility data can be aggregated and/or their temporal and/or spatial resolution adjusted to consider how mobility patterns change at different times of the day, day of the week, specific areas, etc. Also, Figure 7 illustrates a use case for the GeoLife trace. Similar aggregation can be used for the space dimension, for example, aggregating a group of cells into regions within the region being considered (e.g., different sections in a city, different suburbs in a suburban area).

Then, a matrix containing spatio-temporal features can be constructed using time- and space information from the traces as described above. The resulting spatio-temporal mobility matrix is defined as:

$$FM = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1C} \\ t_{21} & t_{22} & \dots & t_{2C} \\ \dots & \dots & \dots & \dots. \\ t_{N1} & t_{N2} & \dots & t_{NC} \end{bmatrix} \quad (1)$$

where N is the number of users and C the number of cells present in the dataset. Each position in the feature matrix $FM(i, j)$ contains the average time user i spent in cell j normalized by the total time the user appears in the trace.

The next step is to apply a non-linear transformation, namely the Logarithmic Likelihood *L*ogit [31], to FM, generating FM'. This step maps the values $FM(i, j)$ (which are values in the range [0, 1]) to the full range of real numbers, $FM'(i, j)$. This step is important in order to highlight the similarities between users.

Finally, an image for each user is generated by representing each row of FM' as a 2D-image. To construct this image, we simply use each position (i, j) as a pixel and the value of each position $t'_{i,j}$ as the intensity of that pixel.

3.2 Feature encoding - CNN autoencoder for extracting geographical preferences

Figure 3 shows an instantiation of the proposed convolutional autoencoder (CAE) architecture when applied to one of the mobility traces used in our study (in this case, the GeoLife trace which is described in detail in Section 4.1). As illustrated in the figure, our CAE architecture consists of two main components, that is, the Encoder responsible for encoding the input image and extracting mobility features; and the decoder responsible for prediction and model output (i.e, image reconstruction). The model input is the image generated from mobility records representing the spatio-temporal characteristics captured from the mobility record as detailed in Section 3.1.

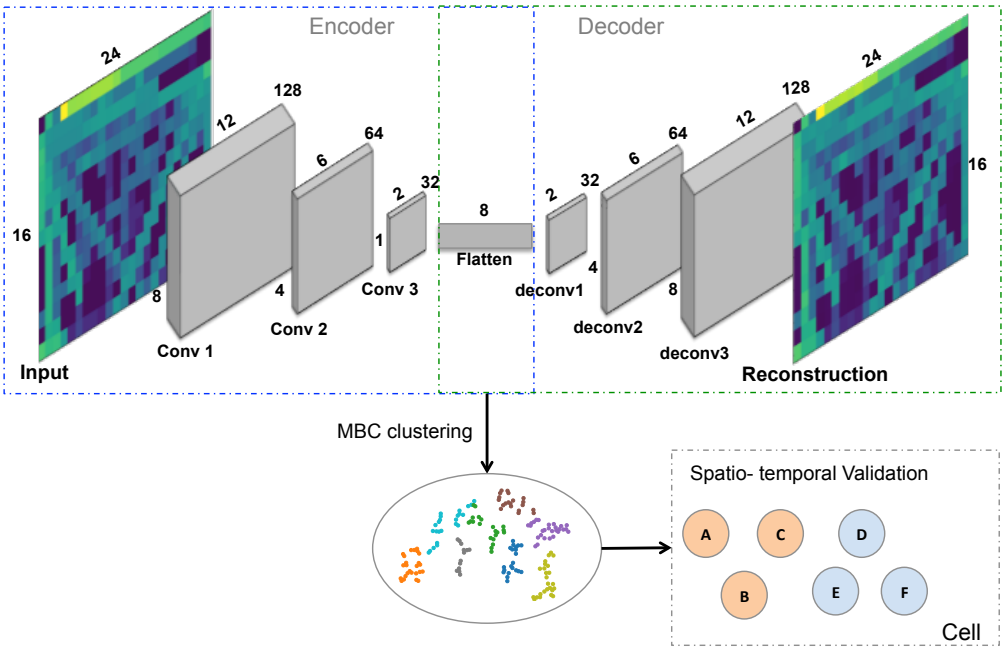


Fig. 3. The proposed deep autoencoder architecture for the GeoLife dataset. There are three convolutional layers followed by a fully connected layer which is composed by only 8 neurons. Then three deconvolutional layers reconstruct the Input on the decoding part. The 8 embedded features represent the encoding of the inputs and are used as input for the MBC clustering algorithm. To evaluate the performance of our approach against other AE architectures, the nodes' contact times are computed, considering the user communities identified.

Feature encoding is performed as follows:

(1) Building the CAE: A deep autoencoder architecture is built for each trace. It is not possible to train a single architecture for general use since the models depend on the size of the input and vary with the application scenario (i.e., the size of the area, number of cells and feature matrix changes from scenario to scenario). The parameters used by the CAE are described in detail in Section 4.1.

(2) Training the CAE: The CAE is trained by using the input image representation obtained from the mobility features described above. During training, the spatio-temporal behavior of the user and its geographical preference are learned, while the model tries to output images as close as possible to the input.

(3) Extracting encoded latent features: Once the network is trained, the reduced embedded features are extracted from the autoencoder latent representation space. These features can be used to make predictions and comparing the original input with the reconstructed image. The size of the reduced feature space depends on the autoencoder architecture and the dataset. Section 4.1 presents the latent space size for the datasets considered in our study. The learned features are finally concatenated into a dense vector (at the *Flatten* layer) which contains the reduced representation of the input model features.

Finally, at the decoder stage, the inverse process is carried out, i.e., the latent vector is reshaped into a matrix, and transformed by a number of deconvolutional layers until the original image is reconstructed. Note that the encoded representation of the input (from the Flatten layer) is used as input to the clustering step.

3.3 Clustering - user community identification

The goal of the user community identification step is to group users with similar spatial and temporal mobility features. To this end, the latent encoded representation of the mobility patterns extracted from the mobility traces by the autoencoder is used as input to the clustering algorithm which identifies user communitiies based on the users' mobility features. As a result of the clustering step, users are labeled according to the community to which they belong.

Two of the most popular feature-based clustering techniques are K-means and Model-Based Clustering (MBC). The K-means algorithm [64] updates cluster centroids by minimizing the within-cluster sum of squared errors. It generates K clusters represented by their centroids.

In this work, we apply MBC to identify user communities from encoded spatio-temporal mobility features generated by the proposed CAE. MBC is a representative of a probabilistic model approach for data clustering that models the density function by a probabilistic mixture model. This method assumes that the data is generated by a mixture distribution and the clusters are defined by one or more mixture components [18]. Each cluster can be modeled by a Gaussian distribution that has three parameters: mean vector, covariance matrix and an associated probability in the mixture, where each point has a probability of belonging to each cluster. The Gaussian Mixture Model algorithm, which assumes that the original data consists of several Gaussian distributions, is a well-known MBC approach [32]. Data that follows the same independent Gaussian distribution is considered to belong to the same cluster. The Expectation-Maximization (EM) algorithm, initialized by hierarchical model-based clustering, is often used for estimating the parameters of the model, where clusters are centered at the mean value, and the geometric features (shape, volume, and orientation) are given by the covariance matrix.

Model-Based Clustering (MBC) has linear complexity and attempts to handle more arbitrarily shaped clusters. Due to the standard deviation parameter, the clusters can take on any elliptical shape, rather than being restricted to circles. This solves problems found in hierarchical and k-means algorithms, which tend to produce spherical and same size groups.

To facilitate visualization of the resulting clusters, the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique was applied in order to reduce the extracted features to a 2D plain. t-SNE is a commonly used technique for the visualization of high-dimensional data in scatter-plots [57]. The technique aids in visualizing high-dimensional data by giving each data-point a location in a two or three low-dimensional data representation in such a way, that nearby points correspond to similar objects and that distant points correspond to dissimilar objects. It aims to preserve as much as possible the significant structure of the high-dimensional data in its low-dimensionality representation.

Other techniques, such as Principal Components Analysis (PCA), also aim at preserving such structure. They mainly differ in the type of structure they preserve. PCA is a linear technique, which keeps the low-dimensionality representations of dissimilar data points far apart. On the other hand, t-SNE is a non-linear technique that keeps the low-dimensionality representations of similar data points close together. t-SNE is also capable of revealing global structure such as the presence of clusters in the data.

t-SNE computes an NxN similarity matrix in both the original data high dimension and in the low-dimensional latent space. The similarity matrix contains the probabilities given by a Student-t distribution between two data-points, where high probability means a pair of similar objects and low probability a pair of dissimilar ones. The low-dimensional embedding is learned by minimizing the Kullback-Leibler divergence between the probability distributions, in the original high dimensional and the low-dimensional data space, with respect to the locations of the points in the latent.

3.4 Validation - computing spatio-temporal performance metrics

As illustrated in Figures 2 and 3, the final step of our user community identification framework is its validation using a number of spatio-temporal metrics. Validation of the user community structure is based on a number of spatio-temporal metrics. For the spatial evaluation, the following similarity metrics are used to compute the quality of the resulting user community structure: the mean square error (MSE), the Adjusted Rand Index (ARI) and the Structural SIMilarity (SSIM) index, which are described in the next paragraph. The average of the similarity metrics for each pair of input images is compared for all node pairs belonging to the same community and belonging to different communities. As such, we expect to see higher average similarity metrics (e.g., SSIM and ARI) between users belonging to the same community and higher average dissimilarity metrics (e.g., MSE) for users belonging to different communities.

The simplest metric is the *mean square error* (MSE), calculated by averaging the squared differences of pixel intensity of two images. However, the MSE is not the most effective way to detect image similarities and dissimilarities. The *Structural SIMilarity* (SSIM) [60] Index identifies the information structures found in images and therefore is considered as a better alternative to measure image similarity. A third metric to evaluate clustering results is the *Adjusted Rand Index* (ARI) [52]. ARI is the corrected-for-chance version of the *Rand Index*, which measures the percentage of decisions (cluster assignments of all pairs of users) that are made correctly.

For time-domain validation and evaluation of the resulting user community structure, we use *contact time* between users belonging to the same- and different communities. The average total time spent together in the same cell for pairs of users belonging to the same community and different communities is computed. Thus, we expect to see higher contact time values between users from the same communities and lower values for users belonging to different communities.

4 EXPERIMENTAL METHODOLOGY

We performed experiments on three mobility datasets to evaluate the performance of the proposed autoencoder architecture and other variants of autoencoders as well as PCA. The datasets we used in our experiments are described in Section 4.1. The autoencoder architectures and parameters are presented in Section 4.2, and the performance comparison in Section 4.3.

4.1 Experimental datasets

The experiments were performed on three datasets selected to represent different mobility scenarios: (1) GeoLife, (2) San Francisco cabs, and (3) Dartmouth. We briefly describe each of the datasets, as well as the pre-processing, including the autoencoder architectures, applied on them.

- **Geolife dataset:** The GeoLife trace records mobility in various scenarios in the city of Beijing, including different modes of transportation (e.g. walking, cycling and driving) [71]. The trace contains GPS trajectories of 182 users, collected over three years and sampled every 5 seconds. The user trajectory is represented in the dataset by a sequence of latitude and longitude set of coordinates over time, containing 17,621 trajectories, over 50,000 hours. The user trajectories are recorded in latitude-longitude geographical coordinates. We converted the geolocation coordinates to two-dimensional UTM Cartesian coordinates [36].
  A 2D-image was generated for each node, where each position of the matrix is a pixel, as detailed in Section 3.1. The 182 images in this dataset have 15 x 23 = 345 pixels, which corresponds to the number of cells for this trace. The values of the feature matrices were individually normalized by their maximum values, so that the values of the pixels for every user would remain between 0 and 1. These images with normalized pixel values were used as input for training the architectures.
- **San Francisco dataset:** The San Francisco trace is a vehicular trace and consists of GPS trajectories of 483 cabs in the City of San Francisco, USA [49]. It was collected during 24 days with intervals between sample positional records ranging from 1 to 3 minutes. Similar to GeoLife trace, we generated an image for each node. The 425 images in this dataset have 12 x 58 = 696 normalized pixels.
- **Dartmouth dataset:** This dataset is a Wi-Fi association trace from the Dartmouth College campus' WLAN [34]. The trace logs user access to Dartmouth College's campus WLAN using Access Point (AP) association and disassociation events. It has 6,524 users over 60 days. In this dataset, the location of a user was set to the location of the access point to which the user was associated at the time. We worked with a subset of this dataset, with only the busy days where the larger number of APs were active. After filtering, this dataset allowed 2004 images (one for each active user) with 26 x 18 = 468 normalized pixels.

4.2 Autoencoder architectures and parameters

In our comparative study, we use four different autoencoder architectures, namely:

- **AE:** fully connected network with five dense layers on the encoder part and five symmetric dense layers to reconstruct the input. This network has the following structure: the five encoding layers contain 512, 256, 64, 32 and 16 units, respectively. The latent mid-layer is composed of an 8-unit feature vector that is used as input for the MBC clustering algorithm.
- **VAE:** fully connected network with one fully connected layer on the encoder part and one symmetric fully connected layer to reconstruct the input. This network has the following structure: For Geolife dataset the two encoding layers contains 64 and 8 units, and for San Francisco and Dartmouth datasets the layers contains 128 and 8 units, respectively. The latent mid-layer is composed of an 8-unit feature vector that is used as input for the MBC clustering algorithm.
- **CAE:** convolutional network with four 2D convolutional layers on the encoder and four 2D convolutional layers to reconstruct the input. The Geolife and San Francisco networks have the following structure: the four convolutional layers contain 128, 64, 32 and 16 filters of size (3x3). The latent layer contain 1 filter of size 3x3. We used the output of the activation in the middle layer, of shape [1, 2] as input features for the clustering algorithm. For Dartmouth dataset the values of the filters are 128, 64, 8 and 4 with size 2x2. The latent layer contains 1 filter of size 2x2, leading to a shape of [1, 2] after activations, to be used as input for the clustering algorithm.

- **Full CAE:** convolutional network with three 2D convolutional layers on the encoder, followed by a fully-connected layer in the latent space, and three symmetric 2D convolutional layers to reconstruct the input. The GeoLife and San Francisco networks have the following structure: the three convolutional layers contain 128, 64 and 32 filters of size (3x3), strides of size 2, respectively. For Dartmouth dataset the parameters for the three convolutional layers are 128, 256 and 16 filters of size (2x2) and strides equal to 2. The encoded latent vector for all three datasets used as input to the MBC is the output of the activations of the fully-connected layer with 8 units.

	Input layer size	layer type	depth	encoder structure	latent layer	filter	stride
GL-AE	345	fully	5	512-256-64-32-16	8 fully	-	-
GL-VAE	345	fully	2	64-8	8 fully	-	-
GL-CAE	15x23	conv	4	128-64-32-16	1x(3x3)	(3x3)	2
GL-Full CAE	15x23	conv/fully	3	128-64-32	8 fully	(3x3)	2
SF-AE	696	fully	5	512-256-64-32-16	8 fully	-	-
SF-VAE	696	fully	2	64-8	8 fully	-	-
SF-CAE	15x28	conv	4	128-64-32-16	1x(3x3)	(3x3)	2
SF-Full CAE	15x28	conv/fully	3	128-64-32	8 fully	(3x3)	2
DM-AE	468	fully	5	512-256-64-32-16	8 fully	-	-
DM-VAE	468	fully	2	64-8	8 fully	-	-
DM-CAE	18x26	conv	4	128-64-8-4	1x(2x2)	(2x2)	2
DM-Full CAE	18x26	conv/fully	3	128-256-16	8 fully	(2x2)	2

Table 1. Hyper-parameters and depths of the autoencoders. conv is the abbreviation of convolution layers and fully is the abbreviation of fully connected layers.

In order to extract features from the images generated for each mobility dataset, we needed to train the four different autoencoder architectures used in our study for each dataset. The most well-known technique for selecting the hyper-parameters of the AEs is grid search [1]. It consists in selecting a set of values for each hyper-parameter. Because this approach may be too expensive, We used a variation of this technique that randomly samples the hyper-parameters uniformly within the grid range. Following this approach, We tested a range of hyper-parameters and neural network depths and use the ones that achieved the lowest loss on the training. The values chosen for the hyper-parameters, as well as the depths of the autoencoder architectures are summarized in Table 1.

For all experiments, the activation function used for all input layers is the *Rectified Linear Unit* (ReLU). The activation function on the output layer is linear, except for the VAE architecture, where a sigmoid function was used. The AE, CAE, and Full CAE networks for all traces were trained to minimize mean square error. The VAE used Kullback Leibler Divergence as its loss function. The optimizer used was *Adam* for all autoencoders. The batch size was set to match the number of image samples, i.e., 182 for GeoLife, 425 for San Francisco, and 2004 for Dartmouth. All architectures were trained for 1000 epochs on the same dataset. The training error and number of parameters of the resulting network and the time elapsed during the training are shown in Table 2. The computation times presented in this table were measured on a laptop computer with a 2.0 GHz quad-core Intel Core i7 processor and 8 GB of memory.

4.3 Performance comparison

In our performance comparison study, besides the AE architectures described in Section 4.2, we also include two "baseline" approaches. The first one which we refer to as "modularity" does not group users in communities and thus all performance metrics (i.e., SSIM, MSE, ARI and contact time,

	Time elapsed (sec.)	Loss	Total Parameters
GL - AE	9.58	0.0046	483,481
GL - VAE	9.49	-31.12	91,237
GL - CAE	2184.21	0.0503	331,890
GL - Full CAE	268.36	3.1765e-04	495,017
SF - AE	49.39	0.0245	1,015,600
SF - VAE	24.05	-217.47	182,216
SF - CAE	2500.02	6.9078e-05	388,017
SF - Full CAE	989.44	8.8735e-05	200,802
DM - AE	78.34	0.0056	443,374
DM - VAE	32.15	-27.89	19,224
DM - CAE	1047.79	0.0072	72,122
DM - Full CAE	537.47	0.0044	266,073

Table 2. Time elapsed, Training Loss and total number of parameters for the three autoencoder architectures analyzed.

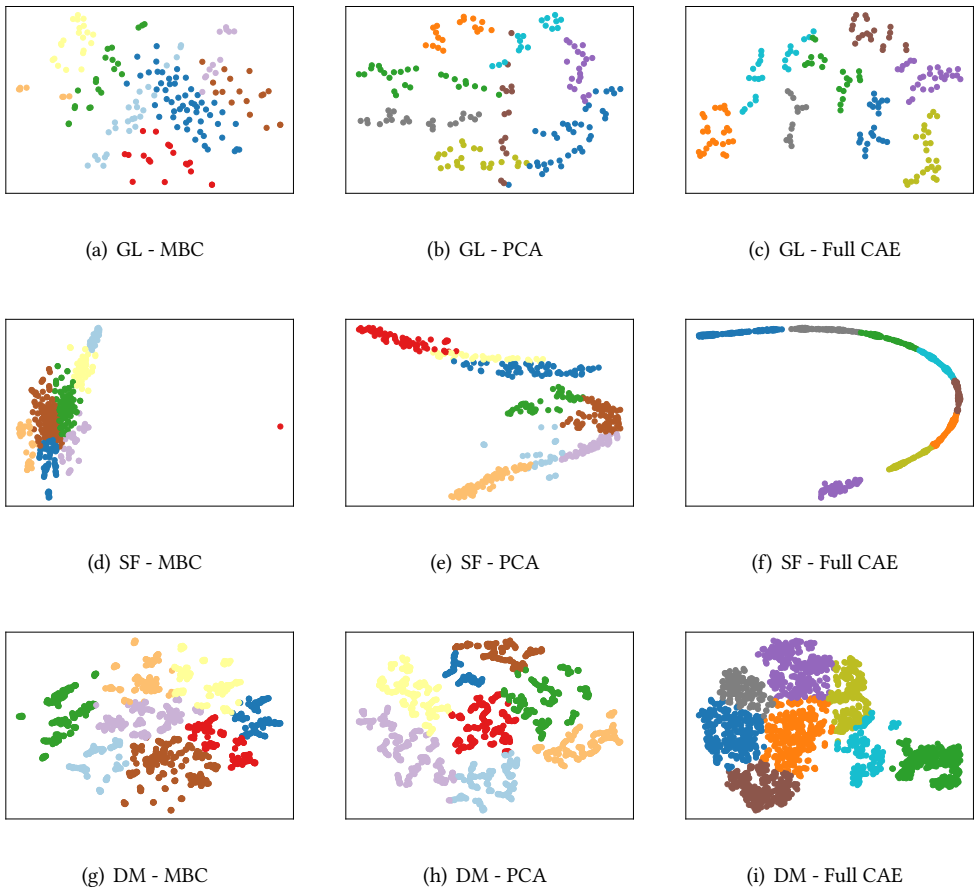
which are described in Section 3.4) are calculated by averaging the metrics for all pairs of users in the dataset. The other baseline approach called "MBC" applies clustering to the datasets without any dimensionality reduction. Additionally, we use liner reduction using Principal Component Analysis (PCA). It is important to mention that PCA and autoencoders share common characteristics. PCA uses linear combinations of the original variables that maximize the variance. Reconstructing the input data from its principal components minimizes the mean squared reconstruction error. Autoencoders with linear activations, which minimize the mean quadratic error also learn the principal components of the data [5]. However, differently from PCA, AEs can learn nonlinear combinations of the features and even more complete representations of data. It is important to mention that human mobility exhibits non-linear behavior and hence cannot be described by linear stochastic models [20]. Moreover, it is also possible to provide higher quality in the AE reconstructions by adjusting the hyper-parameters and applying improvements over the standard AE. As such, we also include different AE models in our comparative study.

In all experiments, the t-SNE technique was used for visualizing the clusters. Its perplexity parameter [57] used to compute the input similarities varied from 5 to 25. Typical values for this parameter are reported in [57] to be between 5 and 50. For the Dartmouth dataset, it was set to be 5, for GeoLife and San Francisco, it was set to 25.

5 RESULTS

Figure 4 shows the output of the clustering step after applying t-SNE for visualization. It plots clusters for the three mobility datasets using three pre-processing/feature encoding methods, namely: MBC (i.e., no dimensionality reduction applied before clustering), PCA, and autoencoder. The results presented for the autoencoder approach were obtained using Full CAE architecture. No significant visible differences were observed in the clustering visualization amongst the different autoencoder architectures.

As discussed before, one of the motivations behind applying autoencoders before clustering was the non-linear nature of the activations. The hypothesis was that the autoencoders would be able to represent non-linearities intrinsic to the data that PCA could not represent, given its linear nature. It is clear that the clusters become increasingly distinct as the linear and non-linear pattern identification techniques are performed and the best visual results are achieved when using autoencoders.



(a) GL - MBC (b) GL - PCA (c) GL - Full CAE

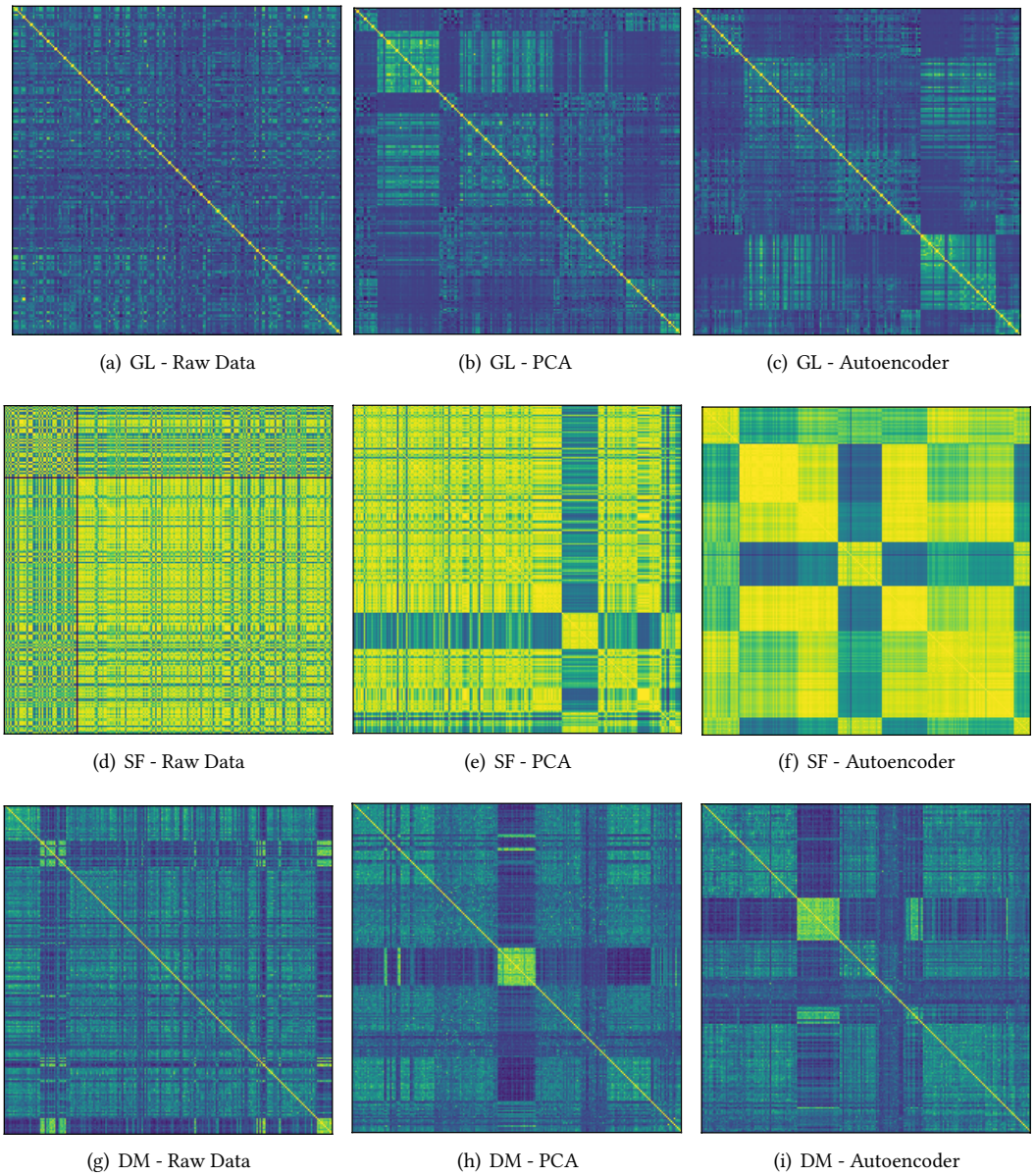
(d) SF - MBC (e) SF - PCA (f) SF - Full CAE

(g) DM - MBC (h) DM - PCA (i) DM - Full CAE

Fig. 4. Clustering visualization after applying t-SNE for MBC clustering, reduced data with PCA, and latent representation by Full AE autoencoder architecture using GeoLife (GL), San Francisco (SF), and Dartmouth (DM) mobility datasets.

Figure 5 visualizes the matrices of pairwise SSIM values $ssim_{ij}$ between users i and j, computed after considering the labels attributed to each user by our proposed methodology when using raw data, PCA, and autoencoder. The values of the 3 matrices computed for each dataset (i.e., each line of Figure 5) are in fact the same for all 3 images. What changes from image to image, for a specific dataset, is the reordering of the matrix rows and columns. Each matrix for each dataset has its rows and columns reordered and sorted according to their group labels (i.e., labels defined by the clustering algorithms) as they were rearranged in a way that users belonging to the same community are shown together in the matrices. In the figure, the brighter the color, the higher the similarity (i.e., SSIM value close to 1), whereas darker colors indicate lower similarity.

We expect to see brighter colors along the main diagonal, as users belonging to the same community tend to have similar geographical preferences and as a result, a higher SSIM similarity metric among them. We find, however, bright regions showing high similarity between different communities. This only means that sometimes, users of different communities can have also



(a) GL - Raw Data (b) GL - PCA (c) GL - Autoencoder

(d) SF - Raw Data (e) SF - PCA (f) SF - Autoencoder

(g) DM - Raw Data (h) DM - PCA (i) DM - Autoencoder

Fig. 5. Similarity for each pair of users sorted by label groups, computed using the SSIM metric for GeoLife (GL), San Francisco (SF), and Dartmouth (DM) datasets. Yellow color shows high similarity and dark green low ones.

similar preferences. These interesting patterns revealed by the SSIM matrices can be interpreted as relationships between the different communities identified.

The noisier and less structured the patterns in the images, the less efficient in separating similar spatial preferences the method is. We can observe that images (a), (d) and (g), generated by applying

clustering over the raw data, are the noisiest of them all, which means they are the less structured ones. Overall, cluster separation is more distinct when using the autoencoder and raw data yields less distinct clusters. The goodness of the method becomes visually more evident especially when looking at SF-Autoencoder results in Figure 5(f), where the different regions with different colors are much more evident and well defined when compared to the images resulting from the PCA and Raw Data. For these last two, we can still see noisier and not well defined structured regions of SSIM values for different communities. PCA also represents better structures in these matrices when compared to Raw Data, however quantitative metric results will show that autoencoder is able to differentiate better the communities, not only in terms of spatial behavior but temporal as well. We discuss this further below.

	SSIM (same)	SSIM (diff)	MSE (same)	MSE (diff)	ARI (same)	ARI (diff)
GL-Modularity	0.1264	0.1264	0.1074	0.1074	0.1140	0.1140
GL-MBC	0.1342 (-)	0.1861 (-)	0.1041 (-)	0.0600 (-)	0.1515 (-)	0.1429 (-)
GL-PCA	0.2512 (87%)	**0.0862** (54%)	**0.0683 (34%)**	**0.1512 (152%)**	0.1942 (28%)	**0.0662 (54%)**
GL-AE	0.2266 (69%)	0.1199 (46%)	0.0776 (25%)	0.1159 (93%)	0.1866 (23%)	0.0820 (43%)
GL-VAE	0.1973 (47%)	0.1144 (38%)	0.0873 (16%)	0.1052 (75%)	0.1541 (2%)	0.1012 (29%)
GL-CAE	0.2372 (77%)	0.0959 (49%)	**0.0688 (34%)**	0.1349 (124%)	0.1882 (24%)	0.0971 (32%)
GL-Full CAE	**0.2784 (107%)**	**0.0860 (54%)**	**0.0687 (34%)**	0.1297 (116%)	**0.2120 (40%)**	0.0816 (43%)
SF-Modularity	0.7632	0.7632	0.0347	0.0347	0.1501	0.1501
SF-MBC	0.7826 (-)	**0.7074** (-)	0.0342 (-)	0.0392 (-)	**0.1674 (-)**	0.1579 (-)
SF-PCA	0.8747 (12%)	0.7147 (-1%)	0.0159 (54%)	**0.0473 (21%)**	0.1640 (-2%)	0.1498 (5%)
SF-AE	0.9528 (22%)	0.7860 (-11%)	0.0010 (97%)	0.0301 (-23%)	0.1618 (-3%)	0.1521 (4%)
SF-VAE	0.9341 (19%)	0.7172 (-1%)	0.0025 (93%)	0.0409 (4%)	0.1666 (-1%)	**0.1393 (12%)**
SF-CAE	0.9517 (22%)	0.7370 (-4%)	**0.0008 (98%)**	0.0416 (6%)	0.1637 (-2%)	0.1455 (8%)
SF-Full CAE	**0.9545 (22%)**	0.7254 (-3%)	**0.0008 (98%)**	0.0410 (5%)	0.1649 (-1%)	0.1444 (9%)
DM-Modularity	0.2018	0.2018	0.0198	0.0198	0.1804	0.1804
DM-MBC	0.2898 (-)	0.1813 (-)	0.0180 (-)	0.0198 (-)	0.2116 (-)	0.1708 (-)
DM-PCA	0.3242 (12%)	0.1652 (9%)	0.0161 (11%)	0.0202 (2%)	0.2435 (15%)	0.1569 (8%)
DM-AE	0.3120 (8%)	0.1931 (-6%)	0.0166 (11%)	**0.0207 (5%)**	0.2444 (15%)	0.1734 (-2%)
DM-VAE	0.3089 (6.6%)	0.1774 (2%)	0.0162 (11%)	**0.0208 (5%)**	0.2329 (12%)	0.1596 (6%)
DM-CAE	0.3419 (18%)	0.1829 (-1%)	0.0158 (12%)	0.0204 (3%)	0.2484 (17%)	0.1709 (0%)
DM-Full CAE	**0.3580 (24%)**	**0.1625 (10%)**	**0.0157 (13%)**	0.0202 (2%)	**0.2634 (24%)**	**0.1505 (12%)**

Table 3. Performance comparison using three different metrics, SSIM, MSE, and ARI for the four AE architectures considered as well as PCA and baseline approaches for Geolife (GL), San Francisco (SF), and Dartmouth (DM) datasets.

Spatial performance is shown in Table 3 while performance according to a temporal metric is shown in Table 4 for different methods of mobility patterns extraction. Table 3 presents results for the three similarity metrics that are SSIM, MSE, and ARI for GeoLife, San Francisco and Dartmouth mobility datasets. The results are shown in absolute values of similarity and also as the percentage difference in relation to MBC clustering. MBC clustering is then used as the baseline for assessing the quality of the clustering methodology; note that it only slightly outperforms the "modularity" approach, which uses no community structure. Table 4 presents results for contact time, i.e., the average total time users spend together in a cell. CI_{min} and CI_{Max} represent lower and upper bounds, respectively, for a 95% confidence interval.

We observe from Table 3 that despite the fact that other methods, in most cases PCA, also yield good performance according to some metrics, the Full CA architecture outperforms it in most metrics for all datasets. It is important to mention that while MSE only considers the Euclidean distance between data points, SSIM can capture the variations in contrast and luminosity of the

	Comm	Mean	CI_{min}	CI_{Max}	# contacts
GeoLife	Modularity	4024842	3615976	4433707	32037
MBC	Same	2863731 (-)	2312482 (-)	3414981 (-)	17432
	Diff	5410701 (-)	4801901 (-)	6019501 (-)	14605
PCA	Same	4795244 (67%)	3672235 (59%)	5918253 (73%)	5470
	Diff	3863132 (-29%)	3428206 (-29%)	4298059 (-29%)	26567
AE	Same	6165288 (115%)	4856236 (110%)	7474340 (119%)	4962
	Diff	3632565 (-33%)	3212565 (-33%)	4052565 (-33%)	27075
VAE	Same	6063904 (112%)	4685518 (102%)	7442289 (118%)	4354
	Diff	3704137 (-31%)	3283634 (-31%)	4124639 (-32%)	27683
CAE	Same	5361019 (87%)	4298002 (86%)	6424036 (88%)	4470
	Diff	3872740 (-28%)	3429318 (-28%)	4316162 (-28%)	27567
Full CAE	Same	**7175820 (151%)**	**5644526 (144%)**	**8707115 (155%)**	4308
	Diff	3535303 (-35%)	3127447 (-35%)	3943159 (-35%)	27729
San Francisco	Modularity	236227	230051	242404	176820
MBC	Same	226120 (-)	217844 (-)	234398 (-)	90246
	Diff	246763 (-)	237561 (-)	255966 (-)	86574
PCA	Same	238084(5%)	223890 (2%)	252279 (7%)	35454
	Diff	235762 (-4%)	228905 (-4%)	242618 (-5%)	141366
AE	Same	244589 (8%)	229042 (5%)	260136 (10%)	28020
	Diff	234653 (-5%)	227923 (-4%)	241383 (-6%)	148800
VAE	Same	249878 (10%)	239179 (9.7%)	260578 (11%)	66288
	Diff	228041 (-7.6%)	220527 (-7.2%)	235554 (-8%)	110532
CAE	Same	251571 (11%)	240800 (11%)	262342 (12%)	65842
	Diff	227124 (-7%)	219641 (-7%)	234608 (-8%)	110978
Full CAE	Same	**259195 (14%)**	**243613 (12%)**	**274777 (17%)**	33792
	Diff	230801 (-6%)	224112 (-6%)	237490 (-7%)	143028
Dartmouth	Modularity	1006	985	1027	4018020
MBC	Same	1129 (-)	1075 (-)	1183 (-)	635442
	Diff	983 (-)	960 (-)	1006 (-)	3382578
PCA	Same	1399 (24%)	1335 (24%)	1463 (24%)	581186
	Diff	940 (-4%)	918 (-4%)	962 (-4%)	3436834
AE	Same	1493 (32%)	1432 (32%)	1554 (32%)	607324
	Diff	919 (-7%)	897 (-7%)	942 (-7%)	3410696
VAE	Same	1130 (0%)	1065 (-1%)	1193 (-1%)	1203022
	Diff	984 (0%)	962 (0%)	1006 (0%)	2812994
CAE	Same	1488 (32%)	1404 (31%)	1572 (33%)	531164
	Diff	933 (-5%)	912 (-5%)	954 (-5%)	3486856
Full CAE	Same	**1557 (38%)**	**1485 (38%)**	**1629 (38%)**	589824
	Diff	911 (-7%)	890 (-7%)	933 (-7%)	3428196

Table 4. Contact times for Geolife (GL), San Francisco (SF), and Dartmouth (DM) datasets.

images. Consequently, MSE is not able to differentiate similar images that only vary in contrast and luminance, providing larger than expected errors [21]. On the other hand, the SSIM metric is not significantly impacted by the changes in luminance and contrast, given that it is equipped to address such issues. Recall that the images representing mobility features extracted from mobility traces use pixel intensity to encode the time spent in a given region, the SSIM is the most adequate metric to measure image similarity. According to the SSIM metric, the Full CAE architecture was able to increase the similarity between nodes belonging to the same community by up to 107%

and the dissimilarity by up to 54% for the GeoLife scenario. The second best method was PCA which showed 87% improvement over the baseline. We argue that this may be due to the movement patterns captured by the Geolife trace, where linear variables were able to represent reasonably well most of the data. However, in the SF scenario, Full CAE achieved 83% better performance and 100% in the Dartmouth scenario compared to PCA.

Table 4 complements the spatial similarity results presented in Table 3 with an assessment of temporal similarity. We can observe from Table 4 that the Full CAE architecture exhibits results that are up to 151% better when compared to MBC, and 125% better compared with PCA. These results show that the community detection method when using Full CAE is able to extract community structures where the users belonging to the same community actually and consistently (i.e., for all datasets studied) spend more time together in the same location. On the other hand, users that do not belong to the same community tend not to meet, and the Full CAE architecture also managed to decrease contact time for members of different communities.

In summary, our results demonstrate that the Full CAE architecture outperforms the other approaches according to most metrics for all datasets. To evaluate the computational complexity of the techniques under study, we measure the time elapsed for training, training loss, and the total number of parameters for each autoencoder architecture and for each dataset analyzed. As we observe from Table 2, the Time elapsed and Training Loss metrics for the Full CAE architecture are smaller than the ones measured for the CAE architecture, even when the total number of network parameters was larger, as it was the case for the GL and DM datasets. Full CAE's reported performance according to the Time elapsed metric demonstrates the feasibility of our approach in identifying user communities which can find applications in intelligent transit and transportation systems, as well as in urban and environmental planning.

We should point out that, although we presented results that show that the proposed Full CAE architecture performs better than the other approaches studied, other AE architectures are still being developed and may prove to be good candidates for feature extraction.

6 DISCUSSION AND APPLICATION

An important characteristic of smart mobility applications such as Lime, Bird, Scoot, Lyft, and Uber-owned Jump is that users can either pick up or drop off equipment anywhere in the city. Attending to the needs of groups of users that share the same mobility pattern allows for (1) increasing the efficiency equipment usage (more shared bicycles/scooters available by certain route) and (2) reducing expenses with equipment relocation, by the equitable placement of shared vehicles.

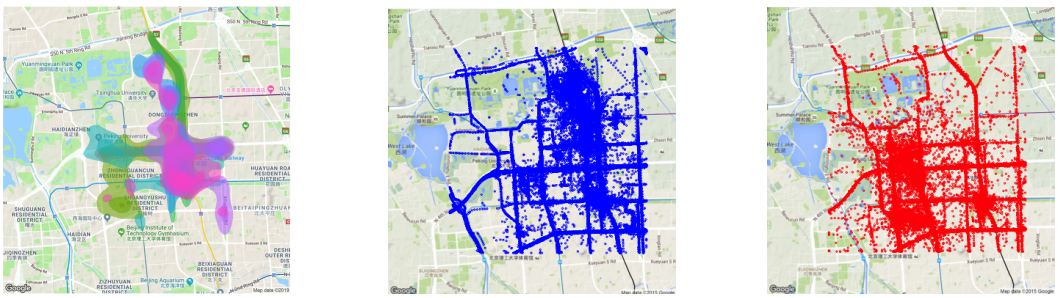
Recently, some cities experimented issues with such vehicles (bikes/scooters) blocking sidewalks and building entrances, causing accidents (e.g., people tripping on scooters) and making public spaces less accessible to children and people with disabilities. The mobile pattern extracted by the proposed method may be used in the decision-making of where to make shared mobility equipment available. Therefore the application can make more precise decisions regarding the distribution of equipment throughout the area visited by the same mobility group. Moreover, the job of collecting and reallocating assets still needs to be performed so that they are available to users in the appropriate parks and racks. This can also be optimized once user groups' movement patterns are known.

Carpool services, such as Waze, Scoop, and Lyft are other examples of mobile-based applications. A carpooling activity consists of a matching process that enables drivers and passengers to be matched, and a daily route commute process that chooses the order at which passengers will be picked up and dropped off. The complexity of the problem of finding the length of the carpooling route, and select the best route scales dramatically according to the increase in the number of candidates and the number of passengers in the carpool [63]. Such applications improve the

efficiency in the carpool sharing matching algorithm if they had knowledge about mobility patterns of groups/communities that share the same paths (fully or partially). A vehicle-to-passenger communication (V2P) approach to support communications between riders and drivers *that allows ride-sharing*, such as [39], could also benefit from our approach.

These applications normally rely on user id, home address and work address to perform matching. If the application knows a priory the information about the group structure, labeling users according to groups that reflect their geographical preferences, communication between drivers and riders and even the matching algorithms could be improved.

Message routing protocols on wireless networks can also take advantage of the mobility pattern identified by the proposed method since users belonging to the same group spend more time together and can be good message forwarders. This characteristic was studied in several previous work [2, 16, 38, 68] and it was found that the time users spend together, in one region and at the same time, has a great impact on the probability of delivering messages in some types of networks. In this work, we show in Table 4 that the proposed group identification strategy increases the average total time spent together for members of the same group by up to 80% if compared to the average time pairs of users spend together without considering community structures. We show that our best autoencoder model increases average total contact time by up to 150% when comparing with an approach that clusters the users using raw mobility data. Moreover, the same metric is improved by 80% when considering our best approach, for one of the datasets in our study, when compared to the non-deep learning approach. On the other hand, when using our best model, we also find that users belonging to different groups (i.e, users who do not have a strong relationship or common interests) have much smaller total contact then the same metric computed for groups extracted by other methods.



(a) All groups in the map (b) Blue group (c) Red group

Fig. 6. Groups extracted by the proposed methodology plotted in the map of the city of Beijing.

Figure 6(a) shows the probability distribution of finding a member of a given group in a given position on the map, for all groups extracted by our method and plotted over the Beijing city map. As expected, different groups have different preferences for different regions of the city. This is more evident when looking at Figure 6(b) and Figure 6(c). These figures show the trajectories of all users belonging to groups 1 (blue) and 2 (red), respectively. The blue group visits much of the city, but has a preference for the upper-right region of the map, while the red group preferences are more evident towards the lower-left region. Figure 7 also shows the probability distribution of finding a member of a group on the map but filtered by weekdays. We can observe from the figure that geographical preferences can change over time. Hence, the proposed autoencoder architecture should be retrained at appropriated time intervals, according to the temporal sensitivity of the application applying the models, such as carpool sharing and V2P communications.

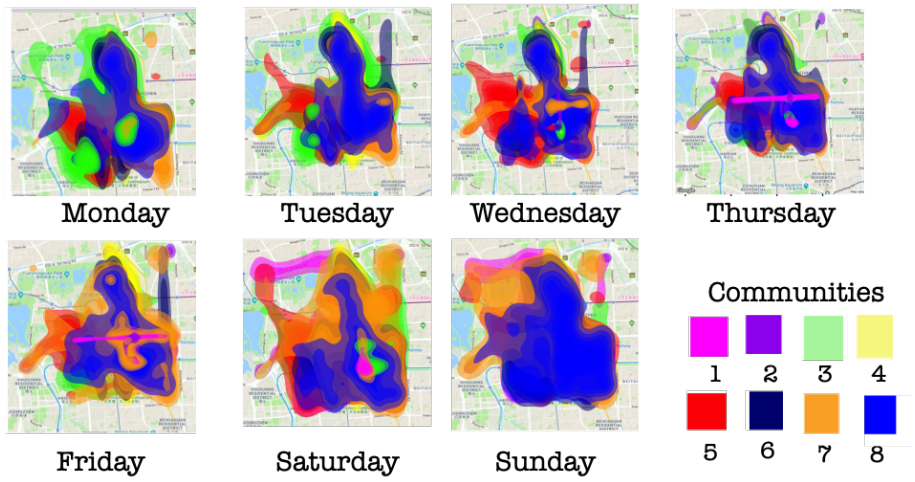


Fig. 7. All communities by week days in the city of Beijing.

Smart mobility applications can distribute their equipment, parking lots and racks according to the groups' regions of preference, also considering groups' routes. The sharing algorithms of carpooling applications could run their matching searches between source and destination more efficiently within a group, increasing the probability of overlapping routes and origin and destination pairs. Also, algorithms for message forwarding or advertisement dissemination applications could take advantage of the location and encounters among the same group of users to forward their messages in order to reach a specific audience, increasing message delivery rate.

7 CONCLUSION

In this paper, we proposed an approach to automatically identify user community structures from real mobility records (e.g., GPS fixes and Wi-Fi network logs). We show that the proposed methodology which uses deep autoencoder to pre-process raw mobility datasets is able to more accurately uncover community structures that identify groups of users sharing common geographical interests and temporal relationships. The proposed methodology was built based on 3 main pillars: (1) geographical preferences feature generation, pre-processing and mobility data transformations, (2) deep autoencoders for dimensionality reduction and extraction of latent non-linear representations of the mobility data, and (3) clustering the output of the autoencoders and visualizing clusters by applying the t-SNE visualization technique.

Through extensive experimentation using three real mobility records representing diverse urban mobility scenarios, we show the effectiveness of the proposed autoencoder-based methodology. Our results show that automatically extracted features lead to an improvement of the performance of spatial similarity metrics while increasing contact time for users in the same community from 30% up to 150%. Moreover, the proposed approach reduces the complexity of the features design task.

8 ACKNOWLEDGMENTS

This work was partially supported by grant CNS 1321151 from the US National Science Foundation and by the Coordenaçao de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] Charu C Aggarwal. 2018. Neural networks and deep learning. *Cham: Springer International Publishing* (2018).

[2] Majeed Alajeely, Robin Doss, and Asma'a Ahmad. 2017. Routing Protocols in Opportunistic Networks: A Survey. *IETE Technical Review* 0, 0 (2017), 1–19.

[3] Vito Albino, Umberto Berardi, and Rosa Dangelico. 2015. Smart Cities: Definitions, Dimensions, Performance, and Initiatives. *Journal of Urban Technology* 22, 01 (2015), 3–21.

[4] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. 2018. Clustering with Deep Learning: Taxonomy and New Methods. *CoRR* abs/1801.07648 (2018).

[5] Pierre Baldi and Kurt Hornik. 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks* 2, 1 (1989), 53 – 58. https://doi.org/10.1016/0893-6080(89)90014-2

[6] Favyen Bastani, Yan Huang, Xing Xie, and Jason W. Powell. 2011. A Greener Transportation Mode: Flexible Routes Discovery from GPS Trajectory Data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '11)*. ACM, New York, NY, USA, 405–408. https://doi.org/10.1145/2093973.2094034

[7] Yoshua Bengio, Aaron C Courville, and Pascal Vincent. 2012. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538* 1 (2012), 2012.

[8] C. M. Bishop and N. M. Nasrabadi. 2007. Pattern Recognition and Machine Learning. *Journal of Electronic Imaging* 16, 4 (2007).

[9] Geoff Boeing. 2017. The Structure and Dynamics of Cities: Urban Data Analysis and Theoretical Modeling, by Marc Barthelemy. *Journal of the American Planning Association* 83, 4 (2017), 418–418.

[10] Francesco Calabrese, Laura Ferrari, and Vincent D. Blondel. 2014. Urban Sensing Using Mobile Phone Network Data: A Survey of Research. *ACM Comput. Surv.* 47, 2 (Nov. 2014), 25:1–25:20.

[11] Luca Canzian and Mirco Musolesi. 2015. Trajectories of Depression: Unobtrusive Monitoring of Depressive States by Means of Smartphone Mobility Traces Analysis. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. 1293–1304.

[12] David Charte, Francisco Charte, Salvador GarcÃa, MarÃa J. del Jesus, and Francisco Herrera. 2018. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. *Information Fusion* 44 (2018), 78 – 96.

[13] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8, 1 (2018), 6085.

[14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[15] Jun Jin Choong, Xin Liu, and Tsuyoshi Murata. 2018. Learning community structure with variational autoencoder. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 69–78.

[16] M. Chuah and A. Coman. 2009. Identifying Connectors and Communities: Understanding Their Impacts on the Performance of a DTN Publish/Subscribe System. In *2009 International Conference on Computational Science and Engineering*, Vol. 4. 1093–1098.

[17] Sina Dabiri and Kevin Heaslip. 2018. Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transportation Research Part C: Emerging Technologies* 86 (2018), 360 – 371. https://doi.org/10.1016/j.trc.2017.11.021

[18] Abhijit Dasgupta and Adrian E. Raftery. 1995. Detecting Features in Spatial Point Processes with Clutter via Model-Based Clustering. *Journal of the Americ. Stat. Association* 93 (1995), 294–302.

[19] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.

[20] Manlio De Domenico, Antonio Lima, and Mirco Musolesi. 2013. Interdependence and predictability of human mobility and social interactions. *Pervasive and Mobile Computing* 9, 6 (2013), 798 – 807.

[21] Richard Dosselmann and Xue Dong Yang. 2011. A comprehensive assessment of the structural similarity index. *Signal, Image and Video Processing* 5, 1 (2011), 81–91.

[22] Nathan Eagle and Alex Sandy Pentland. 2006. Reality mining: sensing complex social systems. *Personal and ubiquitous computing* 10, 4 (2006), 255–268.

[23] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Akihisa Kawanobe. 2016. Deep Feature Extraction from Trajectories for a Transportation Mode Estimation. In *Advances in Knowledge Discovery and Data Mining*, James Bailey, Latifur Khan, Takashi Washio, Gill Dobbie, Joshua Zhexue Huang, and Ruili Wang (Eds.). Springer International Publishing, Cham, 54–66.

[24] D. L. Ferreira, B. A. A. Nunes, and K. Obraczka. 2018. Scale-Free Properties of Human Mobility and Applications to Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems* 19, 11 (2018), 3736–3748.

[25] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *nature* 453, 7196 (2008), 779.
[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
[27] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional Recurrent Neural Networks. In *Artificial Neural Networks – ICANN 2007*, Joaquim Marques de Sá, Luís A. Alexandre, Włodzisław Duch, and Danilo Mandic (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 549–558.
[28] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
[29] Andrea Hess, Karin Anna Hummel, Wilfried N. Gansterer, and Günter Haring. 2015. Data-driven Human Mobility Modeling: A Survey and Engineering Guidance for Mobile Networking. *Comput. Surveys* 48, 3 (Dec. 2015), 38:1–38:39.
[30] S. Hong, K. Lee, and I. Rhee. 2010. STEP: A spatio-temporal mobility model for humans walks. In *The 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2010)*. 630–635.
[31] T. Florian Jaeger. 2008. Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. *Journal of Memory and Language* 59, 4 (2008), 434 – 446. Special Issue: Emerging Data Analysis.
[32] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. Melbourne, Australia, 1965–1972.
[33] Faina Khoroshevsky and Boaz Lerner. 2016. Human mobility-pattern discovery and next-place prediction from GPS data. In *IAPR workshop on multimodal pattern recognition of social signals in human-computer interaction*. Springer, 24–35.
[34] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. 2009. CRAWDAD data set dartmouth/campus (v. 2009-09-09). Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus.
[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*. 1097–1105.
[36] Richard B Langley. 1998. The UTM grid system. *GPS world* 9, 2 (1998), 46–50.
[37] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521 (05 2015), 436 EP –.
[38] Feng Li and Jie Wu. 2009. LocalCom: A Community-based Epidemic Forwarding Scheme in Disruption-tolerant Networks. In *6th IEEE SECON'09*. 1–9.
[39] N. Liu, M. Liu, J. Cao, G. Chen, and W. Lou. 2010. When Transportation Meets Communication: V2P over VANETs. In *2010 IEEE 30th International Conference on Distributed Computing Systems*. 567–576. https://doi.org/10.1109/ICDCS.2010.83
[40] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. 2016. A survey of deep neural network architectures and their applications. *Neurocomputing* 234 (12 2016).
[41] Zhidan Liu, Zhenjiang Li, Kaishun Wu, and Mo Li. 2018. Urban traffic prediction from mobility data using deep learning. *IEEE Network* 32, 4 (2018), 40–46.
[42] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.
[43] Rémi Louf and Marc Barthelemy. 2014. How congestion shapes cities: from mobility patterns to scaling. *Scientific reports* 4, 5561 (2014). https://doi.org/10.1038/srep05561
[44] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. 2017. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* 17, 4 (2017), 818.
[45] Abhinav Mehrotra and Mirco Musolesi. 2018. Using autoencoders to automatically extract mobility features for predicting depressive states. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 3 (2018), 127.
[46] Abhinav Mehrotra and Mirco Musolesi. 2018. Using Autoencoders to Automatically Extract Mobility Features for Predicting Depressive States. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 127 (Sept. 2018), 20 pages. https://doi.org/10.1145/3264937
[47] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. 2018. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. *IEEE ACCESS* 6 (2018), 39501–39514.
[48] Rahul Nair, Elise Miller-Hooks, Robert C. Hampshire, and Ana Busic. 2013. Large-Scale Vehicle Sharing Systems: Analysis of VÃ©lib'. *International Journal of Sustainable Transportation* 7, 1 (2013), 85–106.
[49] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility.
[50] Miguel Rocha, Paulo Cortez, and JosÃ© Neves. 2007. Evolution of neural networks for classification and regression. *Neurocomputing* 70, 16 (2007), 2809 – 2816.
[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham, 234–241.

[52] Jorge M. Santos and Mark Embrechts. 2009. On the Use of the Adjusted Rand Index As a Metric for Evaluating Supervised Classification. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II (ICANN '09)*. Springer-Verlag, Berlin, Heidelberg, 175–184.

[53] Katarzyna Siła-Nowicka, Jan Vandrol, Taylor Oshan, Jed A Long, Urška Demšar, and A Stewart Fotheringham. 2016. Analysis of human mobility patterns from GPS trajectories and contextual information. *International Journal of Geographical Information Science* 30, 5 (2016), 881–906.

[54] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.

[55] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. 2014. Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[56] Eran Toch, Boaz Lerner, Eyal Ben-Zion, and Irad Ben-Gal. 2019. Analyzing large-scale human mobility data: a survey of machine learning methods and applications. *Knowledge and Information Systems* 58, 3 (2019), 501–523.

[57] Laurens Van Der Maaten. 2014. Accelerating t-SNE Using Tree-based Algorithms. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 3221–3245.

[58] Hao Wang, GaoJun Liu, Jianyong Duan, and Lei Zhang. 2017. Detecting transportation modes using deep neural network. *IEICE TRANSACTIONS on Information and Systems* 100, 5 (2017), 1132–1135.

[59] Kunfeng Wang, Chao Gou, Nanning Zheng, James M Rehg, and Fei-Yue Wang. 2017. Parallel vision for perception and understanding of complex scenes: methods, framework, and perspectives. *Artificial Intelligence Review* 48, 3 (2017), 299–329.

[60] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612.

[61] Sebastian J. Wetzel. 2017. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Phys. Rev. E* 96 (Aug 2017), 022140. Issue 2.

[62] R. Wu, G. Luo, J. Shao, L. Tian, and C. Peng. 2018. Location prediction on trajectory data: A review. *Big Data Mining and Analytics* 1, 2 (June 2018), 108–127. https://doi.org/10.26599/BDMA.2018.9020010

[63] Li W Zhao Y Xia J, Curtin KM. 2015. A New Model for a Carpool Matching Service. *PLoS ONE* 10, 6 (2015), 46–50.

[64] Dongkuan Xu and Yingjie Tian. 2015. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science* 2, 2 (01 Jun 2015), 165–193.

[65] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. 2016. Modularity Based Community Detection with Deep Learning. In *IJCAI*, Vol. 16. 2252–2258.

[66] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine* 13, 3 (2018), 55–75.

[67] Shiqi Yu, Sen Jia, and Chunyan Xu. 2017. Convolutional neural networks for hyperspectral image classification. *Neurocomputing* 219 (2017), 88–98.

[68] Peiyan Yuan, Lilin Fan, Ping Liu, and Shaojie Tang. 2016. Recent progress in routing protocols of mobile opportunistic networks: A clear taxonomy, analysis and evaluation. *Journal of Network and Computer Applications* 62 (2016), 163 – 170.

[69] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 29.

[70] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 38.

[71] Yu Zheng, Xing Xie, and Wei-Ying Ma. 2010. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. *IEEE Data(base) Engineering Bulletin* (June 2010). http://research.microsoft.com/apps/pubs/default.aspx?id=131038

[72] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web*. ACM, 791–800.

[73] Fan Zhou, Qiang Gao, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. 2018. Trajectory-User Linking via Variational AutoEncoder.. In *IJCAI*. 3212–3218.

[74] W. Zhu, W. Peng, C. Hung, P. Lei, and L. Chen. 2014. Exploring Sequential Probability Tree for Movement-Based Community Discovery. *IEEE Transactions on Knowledge and Data Engineering* 26, 11 (Nov 2014), 2717–2730.

[75] A. Ziat, E. Delasalles, L. Denoyer, and P. Gallinari. 2017. Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery. In *2017 IEEE International Conference on Data Mining (ICDM)*. 705–714.