A deep learning framework to generate synthetic mobility data

Eren Arkangil*,Mehmet Yildirimoglu, Jiwon Kim, Carlo Prato
*The University of Queensland
School of Civil Engineering
{m.yildirimoglu,jiwon.kim,h.arkangil}@uq.edu.au

†University of Leeds
School of Civil Engineering

***Abstract*—Synthetic datasets are useful when real-world data is limited or unavailable. They can be used in transport simulation models to predict travel behavior or estimate demand for transportation services. However, building these models requires large amounts of data. We propose a novel framework to generate a synthetic population with trip chains using a combination of generative adversarial network (GAN) with recurrent neural network (RNN). Our model is compared with other recent methods, such as Composite Travel Generative Adversarial Networks for Tabular and Sequential Population Synthesis (CTGAN) and shows improved results in predicting trip distributions. The model is evaluated using multiple assessment metrics to gauge its performance and accuracy.**

***Index Terms*—Data Augmentation, Synthetic Mobility Data, Generative Adversarial Networks (GAN), Recurrent Neural Networks (RNN)**

I. Introduction

Census and household travel surveys are conducted and published every 5 or 10 years in most countries. They provide rich content about the demographic information of households and geospatial information on the daily trip routines these households have. The Census dataset includes socioeconomic characteristics of the population such as age, sex, and industry, while household surveys include variables related to the personal trip behaviors of the participants in addition to socioeconomic variables. Although Census data provides various demographic features of the households, the information about travel behavior is mostly not given or very limited. On the other hand, household travel surveys are more travel oriented and provides more details in relation to the travel features of the households.

Census and household survey datasets are commonly used to understand and model travel habits and preferences of the households [1]–[3] These datasets are useful for developing traditional travel demand models as well as agent-based simulation models (e.g., MATSim - Multi-Agent Transport Simulation Toolkit). Particularly, the agent-based simulation models require a large amount of micro-level population data to create the agents in the simulation environment. However, the publicly available samples are limited to get comprehensive outcomes because they only represent a small proportion of the population. Moreover, published datasets are given aggregated and anonymized due to privacy reasons. For example, the conditional distribution of variables (e.g., age distribution given sex) for the entire population of the census are deemed confidential and usually not published. One of the solutions to overcome these challenges is supporting original samples with synthetic data [3]–[6].

The process of a synthetic population for transportation systems can be formulated as a two-step process. First, the household population and its socioeconomic features should be created, and afterward, their trip activities need to be combined [3]. Population data synthesis differs from trajectory data synthesis due to the nature of the datasets. Population features are nominal and kept in a tabular format, whereas trajectories are sequential and not necessarily tabular. Although Census data is generally used to generate population data, it is not useful for trajectory prediction and generation tasks. In addition to the socioeconomic features that Census dataset comprises, household travel surveys include activity locations of individuals. Due to their nature, the activity locations are given as a sequence of spatial zones and will be referred to as trip sequences or trajectories.

Our methodology offers a novel way to generate a synthetic population along with corresponding trip sequences. Our framework is capable of generating a relational database including both socioeconomic features and sequential activity locations. Current state-of-the-art models [3]–[8] mostly focus on only socioeconomic features, while we offer a broader scope by incorporating sequential trip data into the synthetic population generation problem.

In summary our contributions can be listed as:

- Using a Conditional GAN to learn the joint distribution of the socioeconomic variables in the tabular population data
- Using recurrent neural networks to create synthetic sequences of locations
- Proposing a novel merging method to integrate population and trip data

github/erenarkangil/population-mobility-synthesis

- Offering new evaluation methods for aggregated and disaggregated results

II. Literature Review

The proposed algorithms for synthetic data generation problem depends on the type of the data and the task. For example, generative adversarial network models are generally preferred for image generation and recurrent network models for text generation [9]. Our problem includes two types of data: tabular type of population and sequential type of trajectories. The generation of these datasets require different methodologies. There is a wide range of methods that has been proposed to produce synthetic population data; Iterative Proportional Fitting [3], Markov Chain Monte Carlo [5], Bayesian Network [7], and deep learning models [4], [6].

Iterative proportional fitting is based on replication of the sample data. It calculates marginal weights for specified columns of the dataset and generates a synthetic population using these weights. IPF based methods are prone to overfitting, because the interactions between columns are not taken into account. Sometimes these weights are available [3], but it is rarely the case where the weights of all column combinations in this form are available for a given dataset and if the dataset does not contain such information IPF becomes computationally expensive.

Monte Carlo Markov Chain (MCMC) method has been used when direct sampling from joint distributions $P(x, y)$ are not available. It learns from marginal distributions to generate data, and captures characteristics between variables. The principle of the MCMC approach is to use the Gibbs Sampler and combinations of marginal distributions for up-sampling the empirical data [5]. The Gibbs sampler's input is created using marginals and partial conditionals from the dataset.

Bayesian network (BN) method uses Directed Acyclic Graph (DAG) and structure learning methods to find dependencies between variables. BN does not perform well with complex and large-scale datasets since it is time-consuming and computationally expensive. Additionally, prior knowledge is needed to describe the relation between variables and to build DAGs.

Generative adversarial networks (GAN) are the most up-to-date approach to generate population data. GAN architectures are commonly designed to handle uniform datasets such as generation of images [10] rather than tabular datasets. Vanilla and most of the other GAN models suffer from mode collapse problems while generating tabular data [4], [8]. Nevertheless, some of the recent GAN models differ from others considering their capability of handling mixed variables (i.e., numerical and categorical) and learning the joint distribution between variables. Of particular interest to our study, conditional GAN [6], [8] models seem to address the limitations that existing algorithms face when generating tabular population data.

While GAN models are a promising direction to produce synthetic population data, they are not designed to be used in directly generating sequences of discrete tokens, such as trajectories. Sequence Generative Adversarial Nets with Policy Gradient (SEQGAN) [11] adapts the reinforcement learning approach to overcome this issue where generator network is treated as a reinforcement learning agent. This allows the algorithm to generate sequential dataset by using generative networks. Trajectory generation problem is very similar to text generation problem, where predicting the next element in a sequence is the main task, therefore text generation algorithms can be implemented for trajectory generation purposes. For example, recurrent neural network (RNN) and GAN based text generation algorithms are used to create synthetic trajectory data [4], [12]. The model developed by Badu-Marfo and Farooq [4] named Composite Travel GAN, where the tabular population data is generated by Vanilla GAN and the trip data is generated by SEQGAN. To the best of our knowledge, this is the most recent that combines the generation of synthetic population and trajectory problems by using deep learning frameworks. The other composite models [3] morely based on less complex statistical models to generate population with activity chains rather than complex machine learning methods. These articles are not focused on understanding underlying spatial distributions and generally showing less successful results in spatial metrics such as trip distributions [4]. Nonetheless, Badu-Marfo and Farooq [4] does not provide a distinct matching algorithm that enables the merge of the two synthetic datasets, which is one of the contributions that our study makes.

III. Methodology

As previously mentioned, most studies focus on either population or location or activity chain generation. However, these two pieces of information are often needed together (e.g., an agent with demographic features and spatial activity locations in an agent-based simulation model). This paper proposes a comprehensive generation framework that handles both tabular population and non-tabular location data. Our framework consists of three main components: (i) Conditional GAN model to generate synthetic population data, (ii) RNN-based generation algorithm for trip sequence synthesis, (iii) combinatorial optimization algorithm to merge the output of these two algorithms effectively. Please see Fig. 1 for an overview of the proposed framework. We have two sets of features to be trained. The first set represents population features, and the second represents trip features. First set consists of elements of the population training data $P_i = \{P_{i1}, P_{i2}, P_{i3}, \dots \ P_{im}\}$ where each P_i represents a unique individual and m is the total number of features. The second set is defined as $L_i = \{L_{i1}, L_{i2}, L_{i3}, \dots \ L_{in}\}$ where i represents an individual and n represents the maximum number of activity locations. Note that the number of activity locations might differ form one individual to another; we consider the individuals that have visited at most n locations. Feature set L is used to train the location generation component of our algorithm.

Although the feature set P is used to generate population samples, it includes origin destination values in addition to the socioeconomic features. These two features can in fact

TABLE I: Population Features

Features	Samples
Age	25
Sex	Male
Industry	Construction
Origin	(-27.468, 152.021)
Destination	(-27.270, 153.493)

TABLE II: Sequential Features

Features	Samples
Location 1	Kenmore
Location 2	St. Lucia
Location 3	South Brisbane
Location 4	Kenmore

be matched with the activity locations in the feature set L, see Table 2. These common features will be used later within the bipartite assignment to match the observations in the population and location data, see Figure 1. Nevertheless, origin and destination values in the population feature set P are given in coordinates, while the activity locations in the feature set L are represented as zone names or zone ids. This is because it is computationally easier to train continuous variables in the CTGAN model; the excessive number of discrete zone variables in the network increases the computational complexity and causes memory issues within the CTGAN framework. Therefore, we use centroid coordinates instead of zone ids to train the CTGAN model. We keep the locations as categorical for the feature set L, because sequential models such as RNN and SEQGAN need discrete variables for training process.

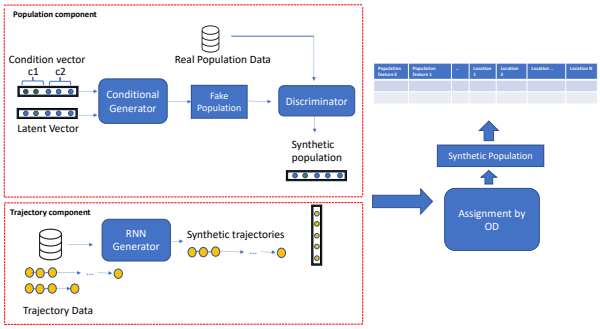

Fig. 1: Schematic representation of the framework

A. Tabular data generation

1) Generative Adversarial Networks: Generative Adversarial Networks (GANs) have two neural network components as generator and discriminator, where they play a minimax game against each other. The generator tries to minimize the loss. The discriminator takes input from real and generated data for classification and returns values between 0 and 1 representing the probability of the given input is fake. Therefore, the discriminator wants to maximize the loss.

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))] \quad (1)$$

The generator produces desired output by using empty latent space and the discriminator behaves like a detective and tries to understand if the output from the generator comes from the real sample or not. Latent space is nothing but a feature vector drawn from a Gaussian distribution with a mean of zero and a standard deviation of one. Through training, the generator updates the weights of this vector and learns the real distributions of the sample dataset. Vanilla GAN architecture commonly fails to handle imbalanced and mixed dataset, however recent studies propose significant changes in the GAN architecture to handle tabular data and learn joint distributions [6], [8], [13].

2) Conditional GAN: Conditional Tabular GAN (CTGAN) architecture differs from other GAN models considering its ability to generate tabular data and learn joint distributions. We make use of CTGAN model to synthesize the population because it allows us to generate discrete values conditionally. Besides, it ensures that correleations are kept. CTGAN explores all possible combinations between categorical columns during training. The model consists of fully connected networks to capture all possible relations between columns [8] and learns probability distributions over all categories. It uses mode-specific normalization and Gaussian mixture to shift the continuous values to an arbitrary range. One hot encoding and Gumble softmax methods are used for discrete values. By using Gumble Softmax trick, the model is able to sample from a categorical variables. The generator creates samples using batch normalization and ReLU (rectified linear unit) activation functions after two fully connected networks. Finally, the Gumble Softmax for discrete values and $tanh$ function is applied to generate synthetic population rows.

Besides, it includes PacGAN framework [14] to prevent mode collapse, which is the common failure for vanilla GANs [4]. CTGAN performs well in synthetic population generation, however, it has difficulties generating realistic trip distributions because the model does not support hierarchical table format and sequential features. Moreover, one hot encoding method can cause memory issues due to high dimensionality of locations. The composite models are proposed to address this issue [4], [15].

B. Location generation

Synthetic data generation for music, text, and location can be formulated similarly considering the sequential nature of the underlying data. To create input sequences, raw input data is converted into vectors of real numbers. Each discrete value is represented with an unique number in these vectors. This process is called tokenization in natural language processing. The number of tokens is determined by the number of the unique sequence values, which is named vocabulary size in text generation algorithms. RNNs show good performance in predicting and generating conditional elements in a sequence [12], [16]. In this study, we use *textgenrnn* library for producing location sequences. The model is inspired by char-rnn [17] and takes location tokens as the input layer. Afterwards, it generates equal size of sequence lists by padding. The sequences are given to the embedding layer as input and create weighted vectors for each token. The embedding layer is followed by two hidden LSTM layers. The attention layer is skip connected to the LSTM and embedding layers. These layers are followed lastly by a dense and output layer.

C. Bipartite assignment

We sample synthetic population and location datasets separately after training CTGAN and RNN models. Because our purpose is to generate individuals with location sequences, we produce the same number of samples for population and location components. We then use the Hungarian algorithm to merge the generated samples.

The Hungarian method takes a non-negative n by n matrix as input, where n represents the number of sampled synthetic individuals. All row and column values are assigned a cost value in the matrix. This cost is defined as the distance between the origin coordinates by CTGAN and RNN components. Since the generated location sequences are discrete, we use the centroid coordinates of the given locations. The cost can be formulated as $d = \sqrt{(x_{ctgan} - x_{rnn})^2 + (y_{ctgan} - y_{rnn})^2}$, where x_{ctgan} and y_{ctgan} are the coordinates of the origin in the synthetic population data, and x_{rnn} and y_{rnn} are the coordinates of the origin zone centroid in the synthetic location data. With the distance values, we build a complete bipartite graph $G = (P, T)$ with n population vertices (P) and n location vertices (T) where the edges are distances between the vertices. Hungarian algorithm finds the optimal matching by minimizing total distance. We use linear assignment without replacement to keep generated trajectories as they are and ensure the trip distributions do not change in the merging process. The algorithm minimizes the total cost (i.e., total distance between the pairs) via the following three steps. Firstly it subtracts the smallest element in each row from all the other elements. Then the smallest entry is equal to 0. Secondly, it subtracts the smallest element in each column. This makes the smallest entry 0 in the column. The steps continue iteratively until the optimal number of zeroes is reached. The optimal number of zeroes is determined by the minimum number of lines required to draw through the row and columns that have the 0 entries.

IV. Experimental Results

We test our algorithm using the Queensland Household Travel Survey data that includes travel behavior data from randomly selected households. The population and trip activity routines of the households are published as separate tables. Each observation is assigned to a unique person identifier code. We therefore connect population and trip activity tables by using these identifiers. The population features contain socioeconomic characteristics such as age, sex, and industry (see Table 1), while the location data includes the sequences of activity locations. The trip locations are categorical and defined as Statistical Area Level 1 (SA1) built from whole Mesh Blocks by Australian Bureau of Statistics. We eliminated observations which contains missing column values and limited the data considering a 40 km radius around Brisbane CBD. We also limited the number of location features to 4 or fewer (84% of total observations). The maximum length of location sequence in dataset is 17, however using location features in this format decrease the model performance due to missing columns. We have 5356 unique individuals with their activity chain locations in our dataset. . In order to build a consistent setup, we train the population and trajectory components with the same dataset.

A. Parameter Configuration

Our CTGAN and RNN components include layer, batch, and epoch size parameters, which we trained with varying numbers. Due to the small size of our dataset, we preferred smaller layer size than base models. Additionally, we opted for a small batch size to mitigate the impact of the limited data size. We applied the early stopping method for epoch size to prevent overfitting. Our RNN model also incorporates a temperature parameter, which determines the creativity of the model through the softmax function in sequential generator models. Values closer to 1 produce more diverse results and efficient sampling, particularly sampling from small datasets.

B. Results

We will examine the results in three sections. First, we focus on our results for population generation from the CTGAN algorithm. Second, we discuss the trip sequence distributions resulting from the RNN algorithm. Last, we focus on the final dataset that results from the merging of the synthetic population and location datasets and analyze the conditional distributions between population and trip specific features. We use a wide range of test metrics considering the most prevalent statistical tests in the transportation literature [4], [6].

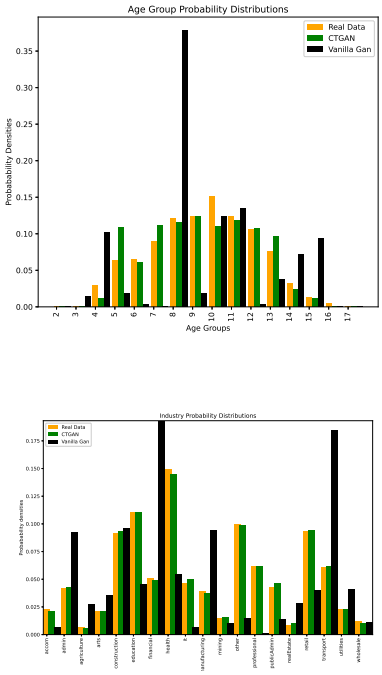

Fig. 2: Comparison of marginals for sex group (top) and industry attributes (bottom)

1) Synthetic Population Data: As discussed in section 2, GAN models are the most promising algorithms to generate

synthetic population. In order to evaluate the performance of the Vanilla GAN model on these features, we compared it with industry and age group features that have more than 20 classes. These feature are sparse in nature, meaning that some of the classes have very few data points, therefore hardest to get correct distributions, especially for Vanilla GAN models. As observed in Figure 3, CTGAN outperforms Vanilla GAN in parallel with literature results. Vanilla GAN model cannot learn imbalanced distributions in the tabular data and struggles to reproduce the distribution of industry and age group features. It exhibits overfitting behavior towards certain labels and generates an excessive amount of them.

2) Synthetic Trip Sequences: Comparing trip sequences one at a time only provides little insight into the trip sequence distributions and cannot be used to determine whether the realistic route patterns are formed. Therefore, we consider the trip length to measure the similarity between real and synthetic trip sequences [4], [12]. We calculate the total distance between all activity locations for each location in the real and synthetic dataset and compare the trip distance distributions that result from this analysis. We use OSRM (Open Source Routing Machine) API to calculate the shortest distance between origin and destination points on a map. We also compare the resulting distributions with the synthetic location dataset from SEQGAN that been applied earlier by [4] for comparison purposes.

Fig 3 clearly shows that RNN closely replicates the trip length distribution in the real dataset, while SEQGAN largely fails to capture the trend in the trip distance distribution. SEQGAN under predicts the distances less than 5km and over predicts the trip lengths greater than 10km, which has been observed in other studies [4] as well. This comparison clearly shows that RNN outperforms SEQGAN and produces realistic trip distance distributions.

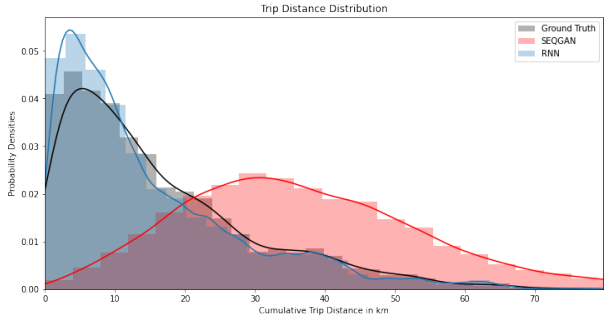

Fig. 3: Comparison of trip length distributions

3) Synthetic Composite Data: The marginal distribution of trip distances are solely not enough to show if the correlations are kept between population and trajectories. One needs to consider joint or conditional distributions between population and trip specific features to further examine this aspect. Figure 4 shows conditional trip distances on a given population feature, industry in this case. In particular, Figure 4a presents the distribution of trip distance for those individuals who are employed in the education sector. Note that this conditional distribution can be observed only after the two datasets are merged using the Hungarian algorithm considering the distance between the pairs. Therefore, these results are labeled as 'CTGAN+RNN' in Figure 4a. The comparison between real dataset and the synthetic dataset that results from our algorithm reveals again a very strong similarity. The framework that we propose captures the joint distribution between variables after matching the two datasets.

Figure 4b depicts the distribution of trip distances for all sectors on the coordinate plane to see if the results in 4a are biased only for certain industries. We have 19 industry variables and use 30 bins to obtain trip length intervals. Thus, figure 4b has 570 points in total. Each point on the plane refers to conditional trip distance percentages of the synthetic and real samples. The percentage values on the X and Y axes represent the frequency of the observations in a specific bin interval. Therefore, for each industry, the sum of 19 separate points is equal to 1. Fig. 4b also includes a proportionality line and a regression line to illustrate the overall agreement between synthetic and ground trip distributions.

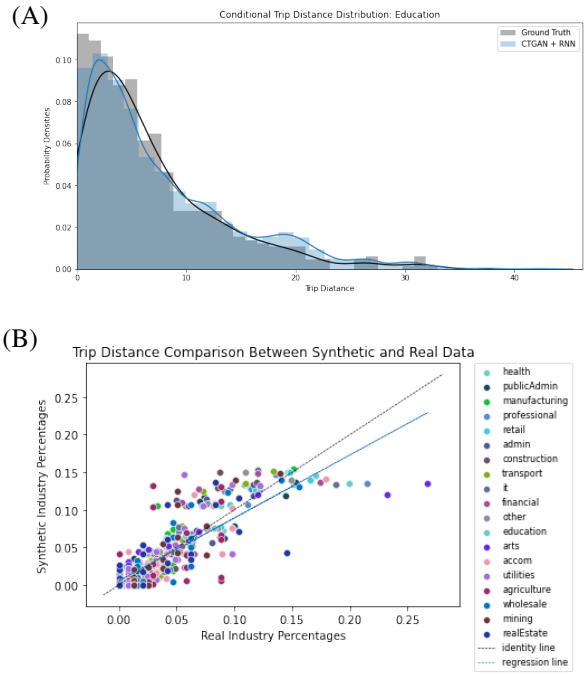

Fig. 4: Comparison of trip length distributions for (a) Education (b) All industries

The marginal and conditional distributions we get from figure 3 and 4 are also tested with statistical metrics; Pearson Correlation, Mean-standardized root mean square error $(SRMSE)$ and $R2$ Score (Coefficient of Determination). We created two frequency lists for the trip length distributions of Conditional GAN, Sequential GAN and RNN models. In Table 3, CTGAN simply indicates the results based on origin and destination values (in coordinates) that are generated as part of the population features. SEQGAN and RNN can be used solely to generate marginal trip distance distributions,

while CTGAN+SEQGAN and CTGAN+RNN represents the results after merging of the corresponding population and location datasets. We set the constant term as 0 when we fit a linear regression equation and calculate the $R2$ score. Overall, SEQGAN exhibits a poor performance on imitating real distribution because the generated conditional distributions follow Gaussian distribution rather than tracking the real distribution. Table III gives a summary of the synthetic trip sequence generation performance of three algorithms.

TABLE III: Benchmarking algorithms on statistical tests

Trip Distance	Algorithm	Pearson	R2 Score	SRMSE
Unconditional	CTGAN	0.495	0.231	0.0572
	SEQGAN	−0.256	−0.558	0.116
	RNN	0.967	0.927	0.0053
Conditional	CTGAN	0.302	0.018	0.0479
	CTGAN + SEQGAN	−0.161	−0.375	0.0789
	CTGAN + RNN	0.867	0.724	0.0134

Fig 5 exhibits the spatial distribution of activity locations in both real and synthetic datasets. The most popular activity locations such as airport and city center are coloured with yellow, as most intense areas according to the real samples. As seen in Figure 5, the synthetic location distribution is similar to the ground truth. Even though some of the areas may exhibit a mismatch in terms of trip frequency, the percentage error does not deviate from the real amount significantly, Therefore we can conclude that our method is able to learn geospatial distributions contextually.

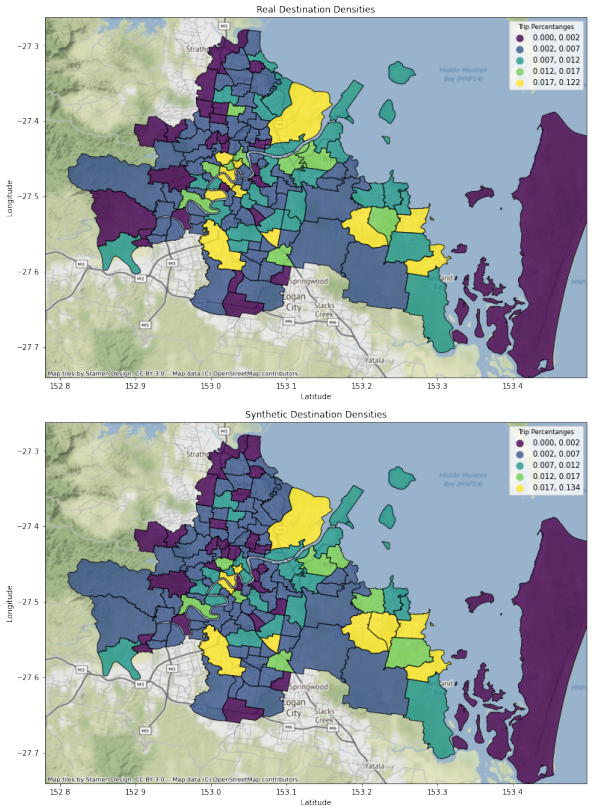

Fig. 5: Spatial distribution of real and synthetic datasets

V. Conclusion and Future Work

Synthesis of a mobility database is challenging due to the complex nature of the data. We propose a hybrid framework for generating such complex dataset. We divided the synthesis problem into two sub-parts by generating population and sequential trajectories separately. We use up-to-date models such as CTGAN for tabular data generation and RNN for location generation. We have also applied a combinatorial optimization algorithm to merge these two synthetic datasets. One of the limitations of our model is that it relies solely on the integration of population and location datasets based on spatial origin-destination features. In future work, we plan to address this limitation by developing a conditional generator model for our location synthesis component. This model will enable us to incorporate sociodemographic features into the merging process, thereby producing more robust results regarding the correlations between population and locations.

References

[1] Vincent Bindschaedler, Reza Shokri, and Carl A Gunter. Plausible deniability for privacy-preserving data synthesis. *arXiv preprint arXiv:1708.07975*, 2017.

[2] Peter R Stopher. *Methods for household travel surveys*, volume 236. Transportation Research Board, 1996.

[3] Sebastian Hörl and Milos Balac. Synthetic population and travel demand for paris and île-de-france based on open and publicly available data. *Transportation Research Part C: Emerging Technologies*, 130:103291, 2021.

[4] Godwin Badu-Marfo, Bilal Farooq, and Zachary Patterson. Composite travel generative adversarial networks for tabular and sequential population synthesis. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[5] Bilal Farooq, Michel Bierlaire, Ricardo Hurtubia, and Gunnar Flötteröd. Simulation based population synthesis. *Transportation Research Part B: Methodological*, 58:243–263, 2013.

[6] Gael Lederrey, Tim Hillel, and Michel Bierlaire. Datgan: Integrating expert knowledge into deep learning for synthetic tabular data. *arXiv preprint arXiv:2203.03489*, 2022.

[7] Lijun Sun and Alexander Erath. A bayesian network approach for population synthesis. *Transportation Research Part C: Emerging Technologies*, 61:49–62, 2015.

[8] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. Text data augmentation for deep learning. *Journal of big Data*, 8(1):1–34, 2021.

[10] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[11] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[12] Alex Berke, Ronan Doorley, Kent Larson, and Esteban Moro. Generating synthetic mobility data for a realistic population with rnns to improve utility and privacy. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 964–967, 2022.

[13] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*, pages 97–112. PMLR, 2021.

[14] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *Advances in neural information processing systems*, 31, 2018.

[15] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29, 2016.

[16] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. *arXiv preprint arXiv:1811.02549*, 2018.

[17] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, 21:23, 2015.