Mobility21

A USDOT NATIONAL UNIVERSITY TRANSPORTATION CENTER

Carnegie Mellon University

Penn
University of Pennsylvania

The Ohio State University

CCAC

Real Time Traffic Congestion Prediction and Mitigation at the City Scale

John Shen
Carnegie Mellon University
https://orcid.org/0000-0002-7225-0629

ABHINAV JAUHRI, Carnegie Mellon, United States, https://orcid.org/0000-0002-9695-9261
BRAD STOCKS, Carnegie Mellon, United States
JIAN HUI LI, Intel Corp., United States
KOICHI YAMADA, Intel Corp., United States
JOHN PAUL SHEN, Carnegie Mellon, United States

FINAL RESEARCH REPORT

Contract # 69A3551747111

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

Generating Realistic Ride-Hailing Data Sets Using GANs

ABHINAV JAUHRI, Carnegie Mellon, United States
BRAD STOCKS, Carnegie Mellon, United States
JIAN HUI LI, Intel Corp., United States
KOICHI YAMADA, Intel Corp., United States
JOHN PAUL SHEN, Carnegie Mellon, United States

This paper focuses on the synthetic generation of human mobility data in urban areas. We present a novel application of Generative Adversarial Networks (GANs) for modeling and generating human mobility data. We leverage actual ride requests from ride sharing/hailing services from four major cities to train our GANs model. Our model captures the spatial and temporal variability of the ride-request patterns observed for all four cities over a typical week. Previous works have characterized the spatial and temporal properties of human mobility data sets using the *fractal dimensionality* and the *densification power law*, respectively, which we utilize to validate our GANs-generated synthetic data sets. We also validate the synthetic data sets using a dynamic vehicle placement application. Such synthetic data sets can avoid privacy concerns and be extremely useful for researchers and policy makers on urban mobility.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**; *Model verification and validation*; Parallel computing methodologies; • **Security and privacy** → *Human and societal aspects of security and privacy*.

Additional Key Words and Phrases: ride-sharing, human mobility, datasets, real-time learning, spatial computing

1 INTRODUCTION

Ride sharing or hailing services have disrupted urban transportation in hundreds of cities around the globe [4, 34]. In United States, it has been estimated that between 24% to 43% of the population have used ride-sharing services in 2018 [27]. Uber alone operates in more than 600 cities around the globe [28]. Ride sharing services have turned urban transportation into a convenient utility (available any place at any time), and become an important part of the economy in large urban areas [11].

Ride request data from ride sharing services can potentially be of great value. Data gathered from ride sharing services could be used to provide insights about traffic and human mobility patterns which are essential for intelligent transportation systems [32]. Ride requests in major cities with high penetration by such services exhibit spatial and temporal variability. Modeling of such variability is a challenging problem for researchers although there is existing work to predict human mobility patterns at coarser-grained geographical locations [15]. Our work aims to model both spatially and temporally granular human mobility patterns. Such granularity of data can help with unresolved challenges related to intelligent transportation, such as: optimal algorithms for dynamic pooling of ride requests [3], real-time pre-placement of vehicles [17, 26], and city scale traffic congestion prediction [22] and avoidance [2, 33]. Access to large amount of actual ride request data is essential to understanding and addressing these challenges.

Data from ride sharing services have been used for real-time sensing and analytics to yield insights on human mobility patterns [16, 31]. Each city exhibits a different pattern of urban mobility – there

Authors' addresses: Abhinav Jauhri, Carnegie Mellon, NASA Research Park, Moffett Field, United States, ajauhri@cmu.edu; Brad Stocks, Carnegie Mellon, NASA Research Park, Moffett Field, United States, brad.stocks@sv.cmu.edu; Jian Hui Li, Intel Corp. 2200 Mission College Blvd. Santa Clara, United States, jian.hui.li@intel.com; Koichi Yamada, Intel Corp. 2200 Mission College Blvd. Santa Clara, United States, koichi.yamada@intel.com; John Paul Shen, Carnegie Mellon, NASA Research Park, Mofett Field, United States, jpshen@cmu.edu.

could be cultural or economical factors governing these patterns. If ride sharing services constitute a significant percentage of the riders in a city, can we build models from ride request data to model urban mobility for the whole city and provide societal benefit without compromising personal privacy? This question motivates us to explore the potential of using Generative Adversarial Networks (GANs) to generate synthetic ride request data sets that exhibit very similar attributes as the actual ride request data sets.

Synthetic data can be very useful in multiple applications to train, test, or inform algorithms. It is widely used to train anomaly detection systems, but also in microsimulation models[1] to inform policy intervention, planning, sensor network deployment, and much more. Some general purpose synthetic data generation methods also utilize ways of describing the data to be generated [12], while others are designed to generate very specific types of data in a rather unstructured way. In this paper, we focus on synthetic data sampling methods that preserve privacy and the correlation between geographical locations in urban cities while incorporating constraints on time for which the data is generated.

This work proposes a novel approach of generating synthetic ride request data sets using GANs. This approach involves viewing ride requests as a (temporal) sequence of (spatial) images of ride request locations. The approach uses GANs to match the properties of the synthetic data sets with that of real ride request data sets. Many recent works using neural networks have looked at demand prediction [37, 38] and traffic prediction at intersections [36]. In our work, we are looking at generating actual ride requests for both spatially and temporally granular intervals. Also, we compare and validate the spatial and temporal variations of the synthetic data sets with the real data sets. For evaluation, we also perform experiments for the vehicle placement problem [17], and compare the results from real and synthetic data sets. In dealing with large amount of data for many cities and long training times for GANs, we develop effective ways to parallelize and scale our GANs training runs using large CPU clusters on AWS. We present our GANs scaling approach and experimental results, and show that significant reduction in training times can be achieved.

To the best of our knowledge, no prior work has looked at generating complete ride requests i.e. source and destination points at fine granularity of time and geographical space for multiple cities. Moreover, we focus on a modeling approach which is computationally efficient for easy adoption in industry and academia.

2 DATA SETS FROM RIDE SHARING SERVICES

In this section, we introduce the actual (real) ride request data sets used for our GANs training and evaluation. We use the real data sets to compare with and validate the GANs generated synthetic data sets.

Our real ride request data sets consist of all the ride requests for an entire week for the four cities. There is a strong repeating pattern from week to week as shown in Figure 2 capturing workdays and weekends. Hence the week-long data should be quite representative. For all four cities, the ride sharing services have significant penetration. Hence we believe the ride request data sets also reflect the overall urban mobility patterns for these cities.

2.1 Ride Requests

Our data sets are real ride requests for four cities over one week period from ride sharing services operating in the United States. Each ride request in the data set includes: request time and pickup location (latitude & longitude), and drop-off time and location (latitude & longitude). For this work we focus on ride request time and pickup location for generating pickup locations; and ride request

[1] https://github.com/citybound/citybound

time and drop-off location to generate drop-off locations. After training independent GANs models for pickup and drop-off locations, we generate synthetic locations using GANs and leverage graph generator approach [6, 16] to pair all pickup and drop-off locations to obtain synthetic ride requests. The trajectory or optimal route for a ride is not within the scope of this work.

For the rest of the paper, we will use the term *ride-locations* to refer to both pickup and drop-off locations wherever they can be used interchangeably.

We do temporal and spatial quantization of the raw ride request data. We partition the entire week into 2016 time intervals of 5 minutes each, and lump together all the ride requests within each interval. We partition spatially the area of the entire city into small squares with side length, ϵ, of 50 meters, and lump together all the ride-locations occurring within the same square area. Lower values of ϵ can be considered with added computational costs of model training; higher values of ϵ ($>$ 100 meters) may hide some of variational properties of ride requests in densely populated cities. Each square area is then represented by a single pixel in a 2-D image with the gray scale intensity of the pixel reflecting the number of ride-locations in that square area (in a time interval). Occurrence of no ride-locations in an area is denoted by zero pixel intensity; positive integers $(1, 2, 3, \ldots)$ as pixel intensity denote the number of ride-locations in the square area.

Combining the temporal and spatial quantizations, the real ride request data set for each city becomes a time sequence of images with each image spatially capturing all the ride requests occurring in a particular 5-min interval.

2.2 Spatial and Temporal Patterns

The actual ride requests in every city exhibit distinct patterns of variability in both the spatial dimension (over geographical area of the city) and the temporal dimension (over each day and over each week). In Figure 1, this variability is illustrated. The ride request density is at its highest at 6pm, and continually decreases over time till 3am. Spatially there are dense patches of ride requests and these dense patches can shift with time, reflecting shifting concentrations of commuters in different areas at different times of day. We observe similar repeating patterns of temporal and spatial variability for all four cities.

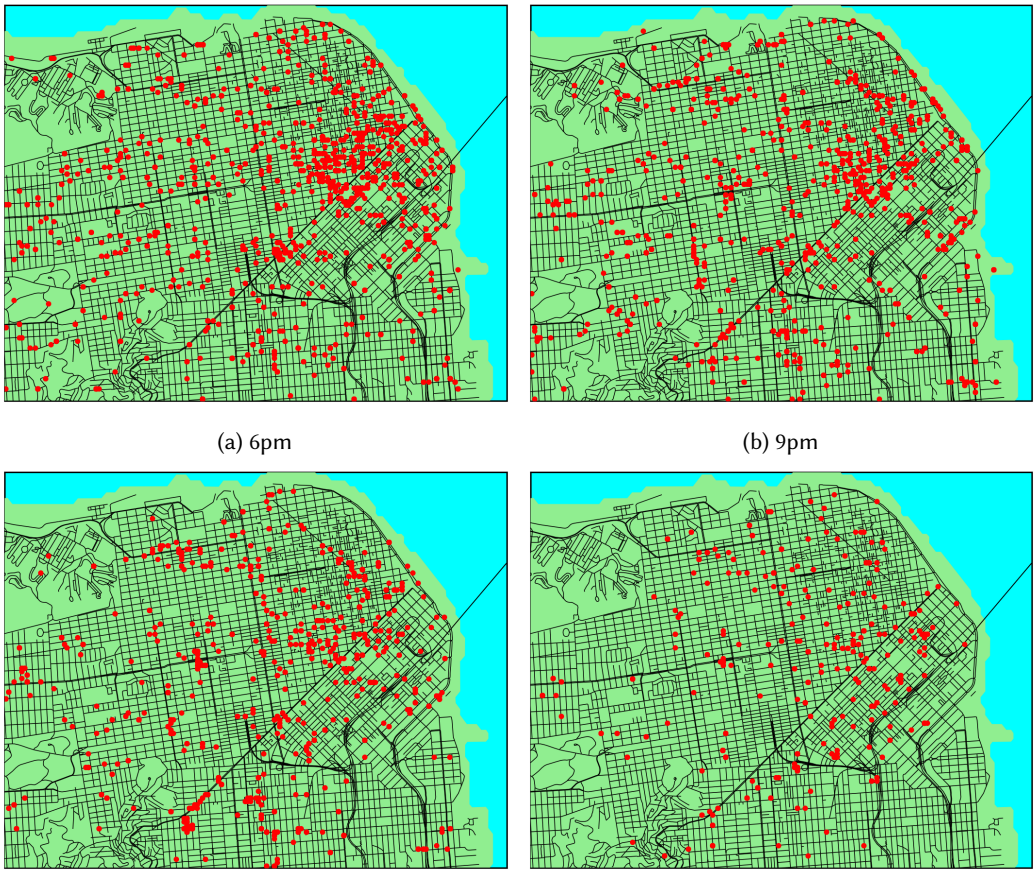
Densification Power Law: A graph can be used to model the ride requests within a 5-min interval, with nodes [2] representing both pickup and drop off locations; a directed edge connecting the pickup node and the drop-off node. It was shown in [16] that the size and density of this Ride Request Graph (RRG) evolves in time in response to the fluctuation of ride requests during each day and through out each week.

It was observed that these ride request graphs exhibit and obey the *Densification Power Law* (DPL) property, similar to other graphs modeling human behaviors such as social networking graphs and publication citation graphs [21]. It was further observed that the ride request graphs for each city exhibit a distinct degree or *exponent* of the DPL, and that this *DPL Exponent* (α) can be viewed as a very succinct quantitative characterization of the temporal variability of the ride request patterns for that city. For any time snapshot t:

$$e(t) \propto n(t)^{\alpha} \tag{1}$$

where $e(t)$ and $n(t)$ are the number of edges and number of nodes respectively, formed by all ride requests occurring in the time interval t. Edge weight denote the number of requests from the same source (pickup) to destination (drop-off) nodes in time snapshot t. The number of edges grows according to a specific exponential power (α) of the number of nodes.

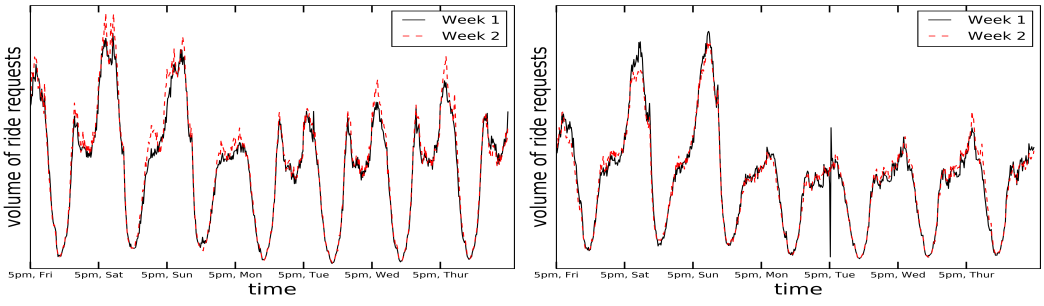
[2]Each node covers a geographical region of a square of length ϵ. Nodes within which no ride requests occur are not considered in the graph.



(a) 6pm (b) 9pm

(c) 12am (d) 3am

Fig. 1. Ride requests for a small region of downtown San Francisco for a typical week day. Each figure shows the aggregated ride-locations (red dots) over a period of an hour. Each red dot may represent one or more ride-locations. Ride density varies spatially and temporally.



(a) San Francisco (b) Los Angeles

Fig. 2. Pattern depicting similarity in ridership for two consecutive weeks.

Correlation Fractal Dimension: There is also a comparable quantitative characterization of the spatial variability of the ride request patterns for each city. The actual geographical locations of the nodes of the ride request graphs is not explicitly represented and therefore another characterization is needed. *Correlation Fractal Dimension* [1, 30] provides a succinct description of a k-dimensional point-set to provide statistics about the distribution of points; it provides a quantitative measure of self-similarity. Fractal dimension has been used to understand and authenticate major works of art [7]. The spatial distribution of ride requests in each time interval can be viewed as a point-set image. We can measure the Correlation Fractal Dimension (D_2) as described in [17]. Values for correlation fractal dimension computed for each time snapshot t fall within a range for each city indicating the degree of self-similarity, and the consistent weekly pattern. For our 2-dimenional space, we impose a 2D-grid with square of side ϵ [3]. For the i-th square, let $C_{\epsilon,i}$ be the count of requests in each square. The correlation fractal dimension is defined as:

$$D_2 \equiv \frac{\partial \log \sum_i C_{\epsilon,i}^2}{\partial \log \epsilon} = \text{constant} \qquad \epsilon \in (\epsilon_1, \epsilon_2) \tag{2}$$

For self-similar data sets, we expect the derivative to be constant for a certain range of ϵ [35]. We observe that this range varies for our four cities, and each city exhibits a distinct value range for its correlation fractal dimension (D_2).

We use the *Densification Power Law Exponent* (α) and the *Correlation Fractal Dimension* (D_2) to capture and characterize the temporal and spatial variability, respectively, for the ride request patterns for each city. RRG created for every time snapshot captures ridership fluctuations over time; nodes in a RRG do not encode any spatial information. Therefore, we compute Correlation Fractal Dimension for each time snapshot to capture the spatial distribution of both pickup and drop-off locations. The temporal evolution, and spatial distribution at any give time snapshot capture the dynamics of ride requests. We use these two parameters independently to confirm the similarity between the real data sets and the GANs generated synthetic data sets. We can claim strong similarity if the values of these two parameters (α and D_2) of the synthetic data sets match closely the values of the same two parameters of the real data sets.

3 GENERATING RIDE REQUESTS USING GANS

3.1 Image generating using GANs

Generative Adversarial Networks learn to generate high quality samples [10] i.e. sample from the data distribution $p(x)$. Previous works by [5, 19] synthesized images of a higher quality using GANs which were hard for humans to distinguish from real images. Conditional GANs are an extension of GANs to sample from a conditional distribution given each image has an associated label which is true for our case of ride requests.

In our framework, we would apply conditional GANs using ride request data in the form of images; similar to as shown in Figure 1 but without the base map shown in color.

3.2 Using GANs for ride request generation

GANs learn a mapping from a random noise vector z to output image x. Conditional GANs learn a mapping from noise vector z and a label y to x [8, 23]. The additional variable in the model allows to generate and discriminate samples conditioned on y. The generator accepts noise data z along with y to produce an image. The discriminator accepts an image x and condition y to predict the probability under condition y that x came from the empirical data distribution rather than from

[3] ϵ parameter is equivalent to the square length for representing a node in the Ride Request Graph.

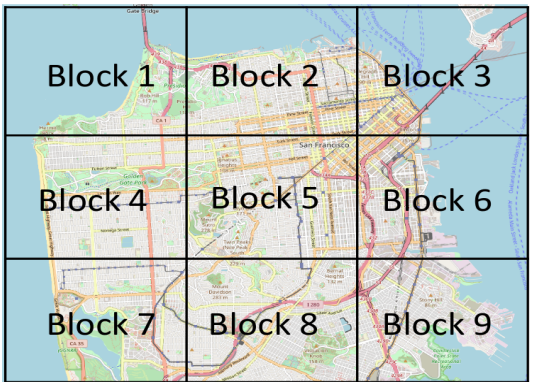


Fig. 3. An illustration of how the geographical region of each city is divided into smaller blocks of equal size and trained independently. Note: image above is not drawn to scale.

the generative model. The objective function can be expressed as:

$$\mathop{\mathbb{E}}_{x,y \sim p_{data}(x,y)}[logD(x,y)] + \mathop{\mathbb{E}}_{z \sim p(z), y \sim p_y}[log(1 - D(G(z,y),y))] \tag{3}$$

where G tries to minimize to this objective function against an adversarial D that tries to maximize it.

3.3 Training Process & Architecture

Every image is assigned a label from the set $\{0, 1, 2, ..., 23\}$ representing the hour of a day. All twelve instances of five minute snapshots within an hour are assigned the same hour label [4]. To accelerate our training using multiple machines, we exploit spatial parallelism by dividing the entire geographical region of a city into an array of blocks. Figure 3 illustrates the division of San Francisco downtown into nine blocks. Keeping our image size similar to MNIST [29], each block is set to represent an image of size 24×24 pixels, with each pixel representing one $50m \times 50m$ square area. Hence, each block covers an area of $1200m \times 1200m$.

Each block, representing a grey scale image of 24×24 pixels, depicts all the ride-locations in that block. Separate images are formed for pickup and drop-off locations; models trained are also separate for pickup and drop-off locations. Each image of a block is labeled with a time interval (for our experiments, the hour in a day) which is similar for both images created from pickup and drop-off locations. The synthetically generated images from an array of blocks with the same time interval label are combined by stitching together all the processed blocks of a city.

The generator network takes an input of a 100-dimensional Gaussian noise sample as well as a one-hot vector encoding of the time snapshot to be generated. It has a single, fully-connected hidden layer without any convolution [9] consisting of 128 ReLU-activated neurons which then passes to a sigmoid activated output layer with the same number of output neurons as the total number of pixels in each block.

The discriminator network has a single hidden layer of 128 ReLU-activated neurons with a single sigmoid activated output neuron. We find that small networks are appropriate for the training data and allow for a quick and stable convergence to be achieved between the discriminator and the generator. Using relatively simple network architectures makes it possible to ensure that the

[4]One could easily extend this approach to a label within the set $\{0, 1, \ldots, 287\}$ if looking at labels associated with any five minute slots of a day or the set $\{0, 1, \ldots, 2015\}$ if looking at labels associated with any five minutes slots of a week.

discriminator and generator are *evenly matched* such that the loss for either network does not saturate early in the training process.

In addition to the standard GANs architecture of generator and discriminator, an additional network is introduced which is referred to as the classifier [20]; it is pre-trained on the training data with the five minute label of the data serving as the classification target. In this way the time information that is encoded into the synthetic data by the generator network is then decoded by the classifier network. The generator is then trained on a weighted sum of the loss from both the classifier and discriminator networks as shown in the following equation:

$$\beta \log D(G(z, y)) + (1 - \beta) \log C(G(z, y)) \tag{4}$$

where β is a tune-able hyper-parameter.

This allows for more explicit loss attribution such that the generator receives two different error signals; one indicating the realism of the synthetic data and the other indicating accuracy relative to the conditioning values. By experiments using MNIST data and [20], we found adding a classifier increases the efficiency of the training process and results in higher quality synthetic data while incurring considerably less training time than other conditional GANs architectures we have experimented.

4 EXPERIMENTAL RESULTS

In this section, we present the cloud infrastructure used for running our experiments. We also present performance results on scaling our GANs workloads on the cloud infrastructure.

4.1 Running GANs on AWS

All experiments are conducted on Amazon Web Services (AWS) using c5.18x instances with each instance containing an Intel Xeon Scalable Processor with 72 virtual cores (vCores) running at 3.0GHz and 144 GB of RAM. We use AWS for easier reproducibility and its large adoption by academia and industry for large scale distributed processing.

In this work we set the block size for each of the four cities to be 1200×1200 meters; each block is trained separately. Enlarging the block size will increase the computational time for training; and the complexity of the model can potentially impact scalability. The total number of blocks for each city are shown in Table 1. The number of blocks are mainly determined by the size of the greater metropolitan area of each city.

To help enhance the scalability of our GANs workload across multiple nodes we make use of Ray [24] from Berkeley, a distributed framework for AI Applications, to efficiently parallelize our workload across cluster of CPU nodes on AWS. Ray provides a convenient API in Python to scale deep learning workloads for numerous libraries, and support for heterogeneous resources like CPUs and GPUs. We also make use of Intel's Math Kernel Library [14] (MKL) which provides machine learning libraries for supporting operations like activation (ReLU), inner product, and other useful functions [13].

4.2 Training Time

Using Ray we scale our training runs by using from 2 to 8 c5.18x instances (containing from 144 cores to 576 cores) on AWS. The scalability results are shown in Figure 4. As can be seen increasing the number of c5.18X Xeon CPU instances can significantly reduce the GANs training time up to 8 c5.18x instances. For the city of Los Angeles, the training time can be reduced from over one hour to less than 20 minutes. For New York City the training time can be reduced to just minutes. Running times for sampling ride requests from the trained models and stitching the images of all

City	*Number of Blocks*
San Francisco	1402
Los Angeles	1978
New York	765
Chicago	1155

Table 1. Training Workload for Different Cities. Each block is trained independently. Doubling the value provided in the table for each city, would give the approximate number of models trained because we have pickup and drop-off locations trained and generated separately.

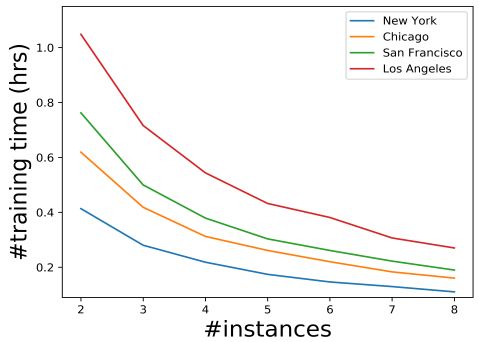


Fig. 4. Training Time Performance Results for training our GANs for pickup locations on AWS with c5.18xlarge Xeon CPU instances. Results for training using drop-off locations show a similar trend.

the blocks together are significantly less than the training times, and are not included in these results.

We also conduct our GANs scaling experiments using GPU instances on AWS. In our initial experiments we observe no real performance improvements using GPUs. Training time using GPUs on AWS was observed to be 5.93 hours on a p3.8xlarge instance using NVIDIA's Multi-Process Service (MPS) [25]. With MPS, the GPU utilization is close to maximum by running multiple of our small GANs training jobs in parallel on a single GPU. Although, the number of jobs which could be executed in parallel on a GPU are not that many in comparison to Xeons. Scaling on GPUs requires more investigation. In this work, we show that it is possible to achieve very nice scalability of our GANs workload using only CPU cores supported by Intel's MKL library and Berkeley's Ray framework.

5 VALIDATION OF SYNTHETIC DATA SETS

We use our trained models to generate synthetic data for a day and validate it using real data by computing metrics for spatial & temporal variations.

5.1 Spatial Variation

The correlation fractal dimension (D_2) gives a bound on the number of ride requests within a geographical region. This is an essential characteristic to match for the data set we are generating using GANs. In Tables 2 & 3, we provide the fractal range (in meters) for each city within which the fractal dimension remains constant. The fractal dimension is computed for every fractal range

City	Fractal Range (m.)	Real Data Sets			Synthetic Data Sets		
		D_2 min.	D_2 max.	D_2 mean	D_2 min.	D_2 max.	D_2 mean
New York	(450, 2500)	1.441	1.753	**1.647**	1.415	1.663	**1.541**
Chicago	(600, 3000)	1.139	1.558	**1.384**	1.188	1.567	**1.435**
San Francisco	(450, 2500)	1.283	1.731	**1.548**	1.242	1.65	**1.426**
Los Angeles	(1500, 4000)	0.962	1.638	**1.355**	1.048	1.489	**1.314**

Table 2. Summary of measured correlation fractal dimensions (D_2) for four cities; computed over a day for every hour using **pickup** locations of real and synthetic data sets.

City	Fractal Range (m.)	Real Data Sets			Synthetic Data Sets		
		D_2 min.	D_2 max.	D_2 mean	D_2 min.	D_2 max.	D_2 mean
New York	(450, 2500)	0.613	1.718	**1.498**	0.914	1.595	**1.414**
Chicago	(600, 3000)	0.315	1.47	**1.216**	0.378	1.539	**1.234**
San Francisco	(450, 2500)	0.499	1.689	**1.363**	0.570	1.685	**1.335**
Los Angeles	(1500, 4000)	0.123	1.487	**1.036**	0.214	1.456	**1.042**

Table 3. Summary of measured correlation fractal dimensions (D_2) for four cities; computed over a day for every hour using **drop-off** locations of real and synthetic data sets.

increment of 100 meters. It is important to note that the fractal range for each city differs [17]. The fractal range provides the ϵ range for which the data exhibits statistical self-similarity [1]. The variation in the fractal ranges for the different cities can be attributed to the geographical shape of the city for which the ride requests are generated. We hypothesize that due to Los Angeles's sprawling nature, a larger ϵ is needed to observe self-similar patterns in comparison to the other three cities, which have a more corridor-like geographical region.

One may also interpret D_2 as a way to measure the fidelity of generated images to that from real data. Comparison of the ranges of values of D_2, in terms of min, max, and mean values, for the real and the synthetic data sets are fairly close although not identical. In most instances the mean value for D_2 is lower for the synthetic data sets in comparison to the real data sets. We believe this discrepancy in the values of D_2 is due to the weakness of the model to capture geographical areas with low frequency of ride requests. Recent works to improve capture learning of high-resolution details of an image [18] can potentially benefit the learning for our ride request images.

5.2 Temporal Variation

DPL provides a characterization of the temporal evolution of ride requests. In the top row of Figure 5 we observe the plot of the DPL exponents α (slop of the line) based on the temporal patterns of the real data sets. For the ride request graph to obey DPL properties, we use graph generator proposed by [16] to connect source and destination locations. In the bottom row of Figure 5 we see the same results based on the synthetic data sets. We can see that the DPL exponent values α correlated quite nicely with that from the real data sets for New York, Chicago, and San Francisco. For Los Angeles, the synthetic exponent is higher than the real observed value; the geographical region for LA is much larger and due to many prominent regions of high request density, the model may likely suffer from bias towards generating more requests in prominent regions leading to a faster increase of the number of edges connecting nodes present in high density regions.

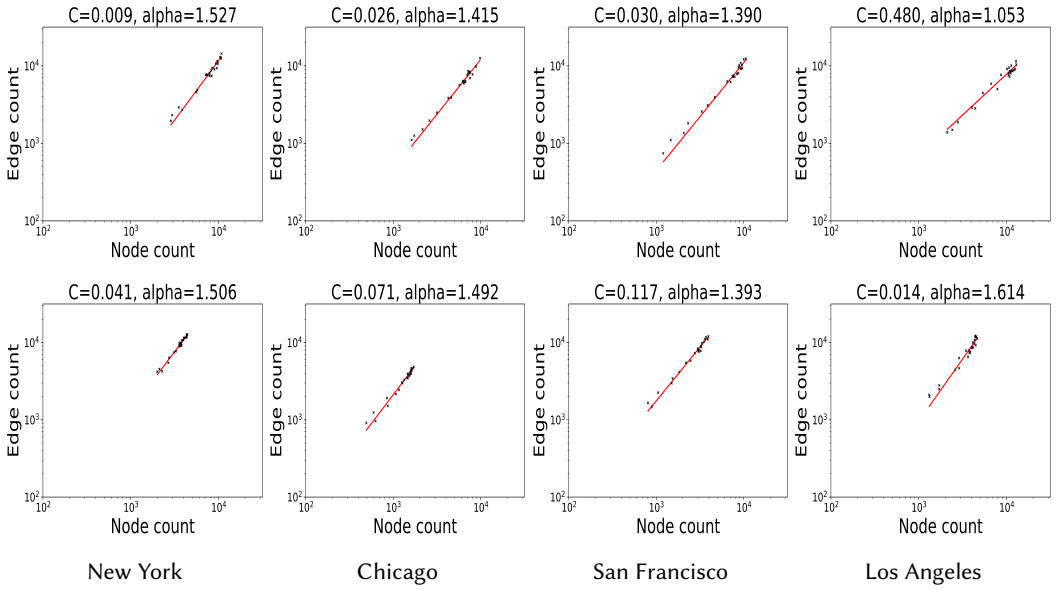


Fig. 5. DPL plots from real data (top row) and synthetic data (bottom row) for four cities. The red line is the least square fit of the form $y = Cx^{\alpha}$, where y and x are number of edges and nodes respectively. $R^2 \approx 1.00$ for all of them.

Another validation of our GANs approach is provided in Figure 6. Here we observe temporal variation of ride requests in terms of the volume of ride requests generated for each hour of a typical weekday. We see that for all four cities, the temporal variation of the synthetic data sets match quite well the temporal variation exhibited by the actual data set. Our model for Los Angeles captures the total volume of ride requests over time intervals correctly but it may not necessarily imply that the model also captures geographical spread of requests within each time interval. For this reason we observe a high α for synthetic data generated in Figure 5; it is important to have multiple metrics for a model to capture human mobility.

6 DYNAMIC VEHICLE PLACEMENT PROBLEM

One practical way to assess the validity of the synthetic data set is to use a specific application and compare the results from the two data sets.

The dynamic vehicle placement problem [17] deals with placing idling vehicles, in real time, to be near future pick-up locations. For every drop-off by a vehicle at time $t-1$ at cell (i, j), its placement at t may be either (i, j) itself or one of the neighbouring cells. This constraint is shown by the bolded cell outlines in Figure 7. The number of neighbouring cells at which the vehicle can be placed is determined by the vehicle's ability to travel to the neighbouring cell within the time snapshot interval.

The goal of dynamic vehicle placement is to choose an adjacent cell for vehicle placement so as to maximize the chance of landing on a new cell which has a pick up request in the next time interval. *Reward* for time interval t is given by the number of pickups which happen on cells where vehicles are placed after drop-off at $(t-1)$. This vehicle placement algorithm is run continuously in moving idling vehicles towards future pick-up locations.

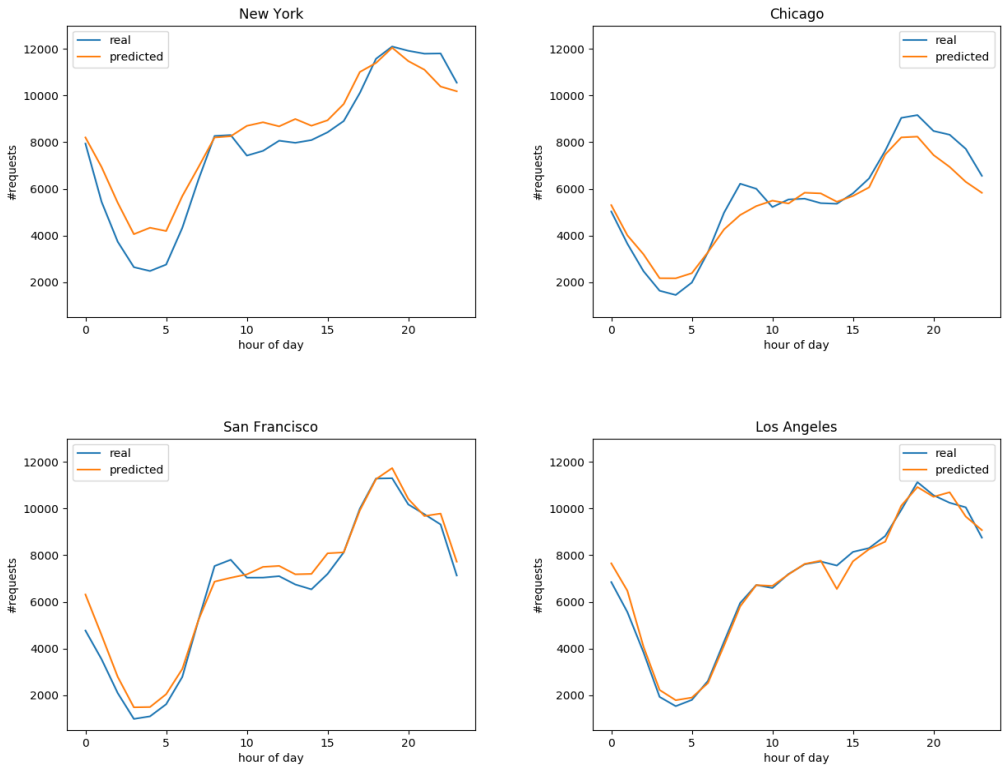


Fig. 6. Plots for four cities highlighting the temporal variability of ride requests visible in both real and our model (predicted) for ride request generation. The pattern is representative of any typical day of week.

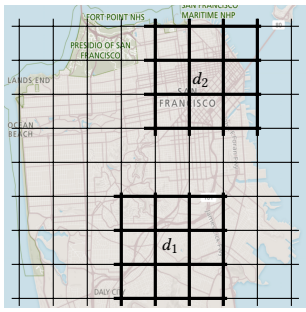
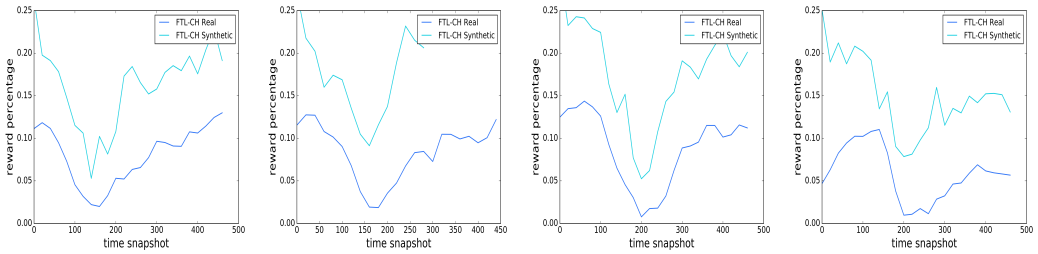


Fig. 7. Geographical space discretized into cells. For example, each drop-off d_i has 9 possible cell placements highlighted by a thick border.

6.1 Dynamic Vehicle Placement Results

For comparison, we used the best performing algorithm in [17], Follow the Leader with Complete History, and apply this algorithm to both the real and synthetic data sets for all four cities. The idea behind the algorithm is to choose the decision, in this case to choose the cell for placement of vehicle, which is most likely to maximize the average reward based on historical pickups.



(a) New York (b) Chicago (c) San Francisco (d) Los Angeles

Fig. 8. FTL-CH algorithm performance on real and synthetic data sets across four cities for a day with three minute time snapshot (aggregate of 480 time snapshots). Certain time snapshots observe no good placements in Chicago using synthetic data; reward percentage is not shown for these time snapshots.

In Figure 8 we compare the performance of this algorithm on the real and synthetic data sets. Performance is measured by a reward function (see [17] for exact definition) reflecting the percentage improvement for earlier or quicker pick up. As can be seen in Figure 8 the synthetic data set tends to give more optimistic results but maintains the same pattern as that from the real data set. Further work is needed for a more thorough comparison.

7 CONCLUSION

The emergence of ride sharing services and the availability of extensive data sets from such services are creating unprecedented opportunities for: 1) doing city-scale data analytics on urban transportation for supporting Intelligent Transportation Systems (ITS); 2) improving the efficiency of ride sharing services; 3) facilitating real-time traffic congestion prediction; and 4) providing new public services for societal benefit. Moreover, the power of neural networks for machine learning has allowed the creation of useful models which can capture human behavior and dynamic real-world scenarios. The key contributions of this paper include:

- We map the ride requests of ride sharing services into a time sequence of images that capture both the temporal and spatial attributes of ride request patterns for a city.
- Based on extensive real world ride request data, we introduce a GANs based workflow for modeling and generating synthetic and realistic ride request data sets for a city.
- We further show that our GANs workload can be effectively scaled using Xeon CPU clusters on AWS, in reducing training times from hours to minutes for each city.
- Using previous work on modelling urban mobility patterns, we validate our GANs generated data sets for ride requests for four major US cities, by comparing the spatial and temporal properties of the GANs generated data sets against that of the real data sets.
- We apply the synthetic data to a real-world problem of dynamic vehicle placement and compare the performance between real and synthetic data sets for four cities.

There are other promising avenues for further research. Some open research topics include:

- Using the GANs generated data sets for experiments on new algorithms for dynamic ride pooling, and real-time traffic congestion prediction.[5]

[5]Code and models of our GANs along with algorithms for looking at transportation related problems like pooling and placement are available at https://github.com/ajauhri/mobility-modeling.

- Using the GANs generated data sets for conducting experiments on *what-if* scenarios related to traffic congestion prediction and mitigation, and planning for future development of transportation infrastructures.

REFERENCES

[1] Alberto Belussi and Christos Faloutsos. 1998. *Estimating the selectivity of spatial queries using thecorrelation'fractal dimension.* Technical Report.

[2] Nicola Bicocchi, Marco Mamei, Andrea Sassi, and Franco Zambonelli. 2017. On recommending opportunistic rides. *IEEE Transactions on Intelligent Transportation Systems* 18, 12 (2017), 3328–3338.

[3] Min Hao Chen, Abhinav Jauhri, and John Paul Shen. 2017. Data Driven Analysis of the Potentials of Dynamic Ride Pooling. In *Proceedings of the 10th ACM SIGSPATIAL Workshop on Computational Transportation Science*. ACM, 7–12.

[4] Boyd Cohen and Jan Kietzmann. 2014. Ride on! Mobility business models for the sharing economy. *Organization & Environment* 27, 3 (2014), 279–296.

[5] Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using aï£ij laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*. 1486–1494.

[6] David Easley and Jon Kleinberg. 2010. Networks, Crowds, and Markets.

[7] Alex Forsythe, Marcos Nadal, Noel Sheehy, Camilo J Cela-Conde, and Martin Sawey. 2011. Predicting beauty: fractal dimension and visual complexity in art. *British journal of psychology* 102, 1 (2011), 49–70.

[8] Jon Gauthier. [n. d.]. Conditional generative adversarial nets for convolutional face generation. ([n. d.]).

[9] Ian Goodfellow. 2018. Introduction to GANs. http://efrosgans.eecs.berkeley.edu/CVPR18_slides/Introduction_by_Goodfellow.pdf. [Online; accessed 31-August-2018].

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT Press.

[11] Robert Hahn and Robert Metcalfe. 2017. *The Ridesharing Revolution: Economic Survey and Synthesis.* Technical Report.

[12] Joseph E Hoag. 2008. *Synthetic data generation: Theory, techniques and applications.* University of Arkansas.

[13] Intel. 2018. Intel MKL-DNN: Primitive Operations. https://intel.github.io/mkl-dnn/group__c__api__primitive.html. [Online; accessed 31-August-2018].

[14] Intel. 2018. Introducing DNN primitives in Intel Math Kernel Library. https://software.intel.com/en-us/articles/introducing-dnn-primitives-in-intelr-mkl. [Online; accessed 31-August-2018].

[15] Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. 2012. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 239–252.

[16] Abhinav Jauhri, Brian Foo, Jerome Berclaz, Chih Chi Hu, Radek Grzeszczuk, Vasu Parameswaran, and John Paul Shen. 2017. Space-time graph modeling of ride requests based on real-world data. *arXiv preprint arXiv:1701.06635* (2017).

[17] Abhinav Jauhri, Carlee Joe-Wong, and John Paul Shen. 2017. On the Real-Time Vehicle Placement Problem. *arXiv preprint arXiv:1712.01235* (2017).

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

[19] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.. In *CVPR*, Vol. 2. 4.

[20] Minhyeok Lee and Junhee Seok. 2017. Controllable Generative Adversarial Network. *arXiv preprint arXiv:1708.00598* (2017).

[21] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 177–187.

[22] Lucas Maystre and Matthias Grossglauser. 2016. ChoiceRank: Identifying Preferences from Node Traffic in Networks. *arXiv preprint arXiv:1610.06525* (2016).

[23] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[24] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael I Jordan, and Ion Stoica. 2017. Ray: A Distributed Framework for Emerging AI Applications. *arXiv preprint arXiv:1712.05889* (2017).

[25] Nvidia. 2018. Multi-Process Service. https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf. [Online; accessed 31-August-2018].

[26] Santi Phithakkitnukoon, Marco Veloso, Carlos Bento, Assaf Biderman, and Carlo Ratti. 2010. Taxi-aware map: Identifying and predicting vacant taxis in the city. In *International Joint Conference on Ambient Intelligence*. Springer, 86–95.

[27] Recode. 2018. How many Americans use ride-sharing services? https://is.gd/Ed14R1. [Online; accessed 31-August-2018].
[28] Recode. 2018. Uber powered four billion rides in 2017. https://is.gd/cgSZFe. [Online; accessed 31-August-2018].
[29] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.
[30] Manfred Schroeder. 2009. *Fractals, chaos, power laws: Minutes from an infinite paradise*. Courier Corporation.
[31] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.
[32] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. 2018. Enhancing urban mobility: Integrating ride-sharing and public transit. *Computers & Operations Research* 90 (2018), 12–21.
[33] Jinjun Tang, Jian Liang, Shen Zhang, Helai Huang, and Fang Liu. 2018. Inferring driving trajectories based on probabilistic model from large scale taxi GPS data. *Physica A: Statistical Mechanics and its Applications* 506 (2018), 566–577.
[34] New York Times. 2018. Taxi Medallions, Once a Safe Investment, Now Drag Owners Into Debt. https://www.nytimes.com/2017/09/10/nyregion/new-york-taxi-medallions-uber.html. [Online; accessed 31-August-2018].
[35] Caetano Traina Jr, Agma Traina, Leejay Wu, and Christos Faloutsos. 2010. Fast feature selection using fractal dimension. *Journal of Information and data Management* 1, 1 (2010), 3.
[36] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, Yanwei Yu, and Zhenhui Li. 2018. Modeling Spatial-Temporal Dynamics for Traffic Prediction. *arXiv preprint arXiv:1803.01254* (2018).
[37] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, and Jieping Ye. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. *arXiv preprint arXiv:1802.08714* (2018).
[38] Xian Zhou, Yanyan Shen, Yanmin Zhu, and Linpeng Huang. 2018. Predicting multi-step citywide passenger demands using attention-based neural networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 736–744.