DILSA+: Predicting Urban Dispersal Events through Deep Survival Analysis with Enhanced Urban Features

AMIN VAHEDIAN KHEZERLOU, University of Wisconsin-Whitewater, USA
XUN ZHOU, XINYI LI, and W. NICK STREET, The University of Iowa, USA
YANHUA LI, Worcester Polytechnic Institute, USA

Urban dispersal events occur when an unexpectedly large number of people leave an area in a relatively short period of time. It is beneficial for the city authorities, such as law enforcement and city management, to have an advance knowledge of such events, as it can help them mitigate the safety risks and handle important challenges such as managing traffic, and so forth. Predicting dispersal events is also beneficial to Taxi drivers and/or ride-sharing services, as it will help them respond to an unexpected demand and gain competitive advantage. Large urban datasets such as detailed trip records and **point of interest** (**POI**) data make such predictions achievable. The related literature mainly focused on taxi demand prediction. The pattern of the demand was assumed to be repetitive and proposed methods aimed at capturing those patterns. However, dispersal events are, by definition, violations of those patterns and are, understandably, missed by the methods in the literature. We proposed a different approach in our prior work [32]. We showed that dispersal events can be predicted by learning the complex patterns of arrival and other features that precede them in time. We proposed a survival analysis formulation of this problem and proposed a two-stage framework (DILSA), where a deep learning model predicted the survival function at each point in time in the future. We used that prediction to determine the time of the dispersal event in the future, or its non-occurrence. However, DILSA is subject to a few limitations. First, based on evidence from the data, mobility patterns can vary through time at a given location. DILSA does not distinguish between different mobility patterns through time. Second, mobility patterns are also different for different locations. DILSA does not have the capability to directly distinguish between different locations based on their mobility patterns. In this article, we address these limitations by proposing a method to capture the interaction between POIs and mobility patterns and we create vector representations of locations based on their mobility patterns. We call our new method DILSA+. We conduct extensive case studies and experiments on the NYC Yellow taxi dataset from 2014 to 2016. Results show that DILSA+ can predict events in the next 5 hours with an F1-score of 0.66. It is significantly better than DILSA and the state-of-the-art deep learning approaches for taxi demand prediction.

CCS Concepts: • **Computing methodologies** → **Neural networks**; • **Information systems** → *Geographic information systems;*

This work is partially supported by the NSF under Grant No. IIS-1566386. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used in this research. X. Z. was funded partially by Safety Research using Simulation University Transportation Center (SAFER-SIM). SAFER-SIM is funded by a grant from the U.S. Department of Transportation's University Transportation Centers Program (69A3551747131). However, the U.S. Government assumes no liability for the contents or use thereof. Y. L. was supported in part by NSF grants no. IIS-1942680 (CAREER), no. CNS-1952085, no. CMMI1831140, and no. DGE-2021871.
Authors' addresses: A. V. Khezerlou, University of Wisconsin-Whitewater, 809 West Starin Road, Whitewater, WI 53190, USA; email: vahediaa@uww.edu; X. Zhou, X. Li, and W. N. Street, The University of Iowa, 21 East Market Street, Iowa City, IA 52242, USA; emails: {xun-zhou, xinyi-li, nick-street}@uiowa.edu; Y. Li, Worcester Polytechnic Institute, 100 Institute Road Worcester, MA 01609, USA; email: yli15@wpi.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
2157-6904/2021/08-ART49 $15.00
https://doi.org/10.1145/3469085

Additional Key Words and Phrases: Data mining, deep learning, survival analysis, dispersal events

ACM Reference format:
Amin Vahedian Khezerlou, Xun Zhou, Xinyi Li, W. Nick Street, and Yanhua Li. 2021. DILSA+: Predicting Urban Dispersal Events through Deep Survival Analysis with Enhanced Urban Features. *ACM Trans. Intell. Syst. Technol.* 12, 4, Article 49 (August 2021), 25 pages.
https://doi.org/10.1145/3469085

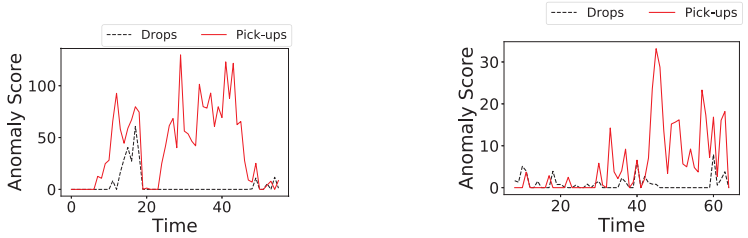
1 INTRODUCTION

Dispersal events occur when an abnormally large number of people depart an area during a relatively short period. Such events can happen after large gatherings such as concerts, protests, or any other unexpected gatherings. Dispersal events, given their unexpected nature, can potentially cause public safety risks, congestion, and unanswered demands of public transportation (e.g., taxis). Therefore, predicting dispersal events as well as the volume of the events in terms of demand size can be greatly beneficial to city authorities and businesses. The safety and the flow of the traffic in the urban area can benefit from such a technique since resources can be allocated to mitigate potential risks or congestion. Transportation businesses, such as ride-sharing platforms and traditional taxi drivers are enabled to gain competitive advantage by covering such events with larger fleets if they can be predicted in advance.

Dispersal event prediction is not a task that can be automated in a trivial way. While most events are scheduled, it is often uncertain when the dispersion will occur. Moreover, there are occasions when events are unplanned or people attend in larger numbers than intended, such as protests or celebrations. In addition, when events are organized privately, the schedule is only known for the attendees and it is not possible obtain advance knowledge of the event if one is outside the group of attendees. For instance, the players of the Pokemon Go game gather for events organized inside the game. In any case, searching, screening, and verifying event schedules is a task that is either labor-intensive or equally challenging to automate.

One potential solution to predict dispersal events, would be to use taxi demand prediction approaches [8, 21, 34, 37, 39] and identify high-demand locations as dispersal events. However, such approaches are designed to learn the regular demand and cannot predict the unexpectedly high demand at the time of dispersal events. Another approach to predict dispersal events could be to observe abnormal gatherings and use them as a signal to predict future dispersals. Figure 1(a) shows an example of such an incident. The dashed and solid lines represent the anomaly scores [23] of the drops and the pick-ups, respectively. However, the evidence of a dispersal event in the future is not always that clear. Figure 1(b) shows such a case, where there are no abnormal drops preceding the dispersal event. Our prior work [32] addressed such challenges and the limitations of the related work, by proposing an approach called DILSA. We formulated the dispersal event prediction as a "Survival Analysis" problem [20], where we treated the dispersal event as the "death" event in survival analysis. By learning the complex patterns of arrival and departures using deep learning models, we predicted the probability of "death" at given points in time in the future. Specifically, given the historical taxi trip records and other relevant features (e.g., weather, **point of interest** (**POI**)), we predicted (1) when and where abnormally high taxi demand will occur, and (2) the volume of demand during the dispersal event.

This article is a significant extension to our prior work [32]. DILSA uses global models for all locations to predict dispersal events and predict abnormal taxi demand in the case of such events. Different locations are likely to have different patterns and one model might not be able to distinguish between them. In our prior work, we added POI features to help the model distinguish



(a) An anomalously high pick-up, preceded by an anomalously high number of arrivals.

(b) An anomalously high pick-up, not preceded by a drop event.

Fig. 1. Examples of abnormally large number of pick-ups.

between different "types" of locations. However, POI features are constant through time and they will fail to distinguish between mobility patterns through time. Moreover, our prior work did not include any features to directly represent and differentiate locations based on their mobility patterns, such as pick-up time series.

In this article, we address these limitations, by making two major contributions. (1) We propose to address the first limitation by capturing the interaction between POIs and mobility patterns, i.e., pick-ups and drop-offs. (2) We propose a method to create a vector representation of a location solely based on its mobility patterns. We call our new method **DILSA+**.

We evaluate our methods using real-world data from New York City. Our evaluations show our method identifies dispersal events with an F1-score of 0.71. Also, our method predicts the pick-up demand in the case of anomaly with superior accuracy compared to the baselines.

The rest of the article is organized as follows: In the next section, we discuss the related work, followed by problem formulation. Then we present **Gathering-based Dispersal** (**GbD**) event prediction as a baseline solution. In the following section, we present our survival analysis formulation and prior work, DILSA, as a second baseline. Next, we present our proposed computational solution. Finally, we present the evaluations and conclude the article.

2 RELATED WORK

Prior related works include (1) event detection and forecasting, (2) taxi demand prediction, and (3) survival analysis.

Event Detection and Forecasting: Event detection has been widely studied in various domains, including public health, urban computing, and social network analysis. The works [16, 17, 23] and other recent works on event detection [12, 18] use already-observed counts. An event is defined as a region with significantly higher counts, such as disease reports or number of taxi drops. Social media posts and geo-tagged tweets have been used as well to detect and forecast events such as social unrests and protests [6, 19, 27, 36, 40, 41]. Regions and time windows where the frequency of certain keywords exhibit abnormal changes are identified as events. These works do not use mobility data. The **dynamic patterns** of the events such as gathering or dispersing **are not captured**.

Works [10, 13, 42] use traffic flow data to detect gathering events. Vahedian et al. use destination prediction to predict gathering events [14, 31]. However, **such methods are not applicable** to dispersal events, as trajectories and traffic flow are **observed only after such events**.

Taxi demand prediction has been studied closely in recent years, due to access to public taxi datasets [39]. To the best of our knowledge, none of the proposed methods directly address the prediction problem in the case of anomaly. State-of-the-art methods for predicting taxi demand

use historical data and time series analysis. Yao et al. [35] propose a deep learning framework that captures the spatial and temporal dependencies to predict taxi demand. Xu et al. [34] formulate an LSTM Network to learn the regular pattern of taxi demand. Zhao et al. [39] show that regular taxi demand is highly predictable and test different algorithms to approach the maximum accuracy. Zhang et al. [37] used spatial clustering to predict demand hotspots. They predict areas with high density of demand using DBSCAN. Such areas, despite having high demand, are part of the regular pattern. Moreira-Matias et al. [21] used streams of taxi data as time series to predict taxi demand in the next 30-minute period. Davis et al. [8] used time series analysis to solve the demand prediction problem, giving recommendations to drivers. Mukai and Yoden [22] used a simple multi-output **Artificial Neural Network** (**ANN**) to predict demand, using features created from recent demand, time and weather information. Okawa et al. [25] use a deep learning method to predict taxi pick-ups, while they treat each pick-up as an event. While their work is a direct address of the event prediction, the fact that they treat each pick-up as an event makes their prediction about the regular pick-up patterns, as they do not distinguish between normal and abnormal pick-ups. Ding et al. [9] propose a Memory-Network approach to predict extreme time series events, that can potentially be used to predict dispersal events. However, their technique uses simple thresholding to identify events. Such method results in predictions that have no statistical significance.

The above-mentioned research aims at **learning the regular pattern** of taxi demand in **absence of anomaly**. Considering the regular demand is highly predictable, in this article, we take on the **harder challenge of predicting anomalous taxi demand**, which we believe is of greater importance.

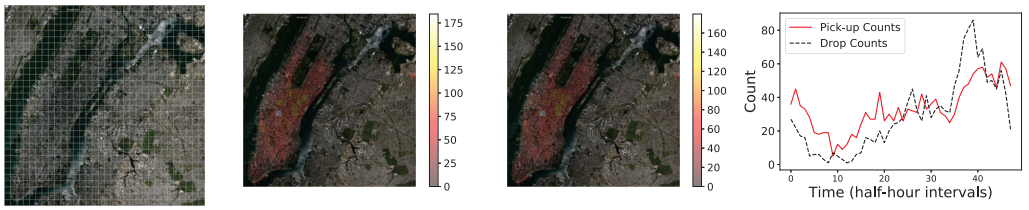
Survival analysis is the analysis of duration of time until an event. It has been applied in engineering as well as health practices [29], for which it was originally developed [20]. To the best of our knowledge, this is the first time survival analysis is used in the context of urban event prediction. In this article, we propose to use a deep ANN to predict the probabilities of survival. Predicting the probabilities of survival at different time points using a common internal representation (the hidden nodes of a deep ANN) allows the learned model to share information across the time points, resulting in better predictive results.

In our prior work [32], we proposed DILSA to predict the occurrence of dispersal events and predict unusually high taxi demand in the case of a dispersal event. DILSA took advantage of a variety of features, most notably including mobility features. However, mobility patterns vary by space and time. To account for this variation, DILSA used a set of features that represented the types of POIs in the area. POI features are time-invariant and failed to capture the variation of human mobility through time. Moreover, POI features are not a direct representation of differences in human mobility patterns, which is the focus of attention in predicting dispersal events. Therefore, in this article, we propose to address these limitations by (1) formulating the interaction between human mobility and POIs through time (MobPOI features) and (2) representing each location with a vector that is directly obtained from the pairwise similarities of mobility patterns among all locations.

3 PROBLEM FORMULATION

3.1 Concepts and Definitions

We define a spatio-temporal field $Z = (S, T)$ as a two-dimensional geographical region S paired with a period of time T. S is partitioned by a grid. Each grid cell $l_1, l_2, \ldots, l_{|S|}$ represents a distinct location in the geographical region. T is partitioned into fixed-length timesteps. Given Z, the location of any moving object can be mapped into a grid cell in S and a timestep in T. For example,



(a) A spatial grid over New York City. (b) Heatmap of pick-up baselines (2014). (c) Heatmap of drop baselines (2014). (d) Pick-up and drop counts of a location in Lower Manhattan on Jan. 2nd, 2014.

Fig. 2. Example of grid, counts, and baselines (best viewed in color).

pick-up and drop locations from a taxi trip record can be represented by a tuple of length four, using the following definition.

Definition 1. A **trip** is a tuple (l^s, t^s, l^d, t^d), where the elements represent trip source location, source time, destination location, and destination time, respectively.

Figure 2(a) shows an example of a grid overlaid on New York City. Based on the definition of trips in the spatio-temporal field, we define the following statistics for each location and time.

Definition 2. The **pick-up count** of grid cell l at time t, denoted by $C^p_{l,t}$, is the number of trips that have l and t as their source location and time. Similarly, **drop count**, denoted by $C^d_{l,t}$, is the number of trips that have l and t as their destination location and time.

Figure 2(d) shows an example of a pick-up count time series and drop count time series for a grid cell in Lower Manhattan on January 2nd, 2014. Pick-up and drop counts show the arrivals and departures at every location and time in Z. If T encompasses countable periods of time, we can assume that defined counts will demonstrate a repetitive pattern at each location throughout those periods. For instance, if T encompasses a year, the countable periods can be represented as seasons, months, weeks, or days. Therefore, the same timesteps within different periods can be expected to have similar values. Thus, we define the baseline counts for arrivals and departures at every location for a timestep in a period, considering the days as the countable periods.

Definition 3. The **pick-up baseline** of grid cell l at time t, denoted by $B^p_{l,t}$, is the average of pick-up counts at l at the same time of day. Similarly, **drop baseline**, denoted by $B^d_{l,t}$, is the average of drop counts at l at the same time of day.

Figure 2(b) and (c) show heatmaps of baseline pick-up and drop counts for the grid in Figure 2(a) for 2014.

The counts and baselines are defined for a grid cell at a given timestep. However, they can be obtained for a spatial region too. We define spatial region l^* to be the surrounding region of a grid cell.

Definition 4. **Surrounding area** of grid cell $l = (a, b)$, where a and b are the grid coordinates of l, is the rectangular area bounded between grid cells $(a - \lambda, b - \lambda)$ and $(a + \lambda, b + \lambda)$, and is denoted by l^*.

l^* is the spatial region containing $(2\lambda + 1)^2$ grid cells, consisting of l and a set of its nearby locations. Similarly, the counts and baselines can be obtained for a time interval longer than one timestep. In other words, they can be obtained for spatio-temporal regions. We define spatio-temporal regions as follows.

Definition 5. A **spatio-temporal region** $R = (S_R, T_R)$ is a pair of rectangular sub-fields of S and a continuous subset of T.

To study the unexpectedly high taxi demands, in this article, we are interested in the spatio-temporal regions at which we observe significantly higher counts than expected, i.e., when C_R^p is significantly higher than B_R^p. To this end, we assume C_R^p follows a Poisson distribution and test the following hypotheses: H_0: C_R^p is from a Poisson distribution of parameter B_R^p, H_1: C_R^p is from a Poisson distribution of a parameter larger than B_R^p. We use the Expectation-based Likelihood Ratio Test of Neill et al. [23]:

$$LLR(R) = \begin{cases} C_R^p \log \frac{C_R^p}{B_R^p} + (B_R^p - C_R^p) & \text{if } C_R^p \geq B_R^p \\ 0 & otherwise. \end{cases} \tag{1}$$

Zhou et al. [42] showed that $LLR(R)$ is at α-level significance if $1 - Pr(X \leq C_R^p) \leq \alpha$, where $X \sim Poisson(B_R^p)$. Based on this test, we define dispersal events.

Definition 6. There is a **dispersal event** at spatio-temporal region R, if $LLR(R)$ is significant at α-level.

In an urban area, every location has specific attributes other than the pick-up and drop counts. In this article, we consider two of these attributes: weather and POI vector. Every location has a daily maximum and minimum temperature, average wind speed, and total precipitation, which impact the traffic and people's movement. In addition, every location consists of several places that can be categorized into different functions. For instance, one grid cell in S might be home to many hotels and few shopping centers, while another grid cell might contain many shopping centers. The distribution of categories of places over the space has an impact on people's movement in spatial field. To account for this impact, we define the POI vector as follows.

Definition 7. **POI vector** of location l is a vector $V^l = (v_1^l, v_2^l, \ldots, v_n^l)$, where v_i^l is the number of places in category i at l.

3.2 Survival Analysis

Survival analysis is used to analyze the expected duration of time until an event happens [20]. The event could be death or failure, or in this article, a dispersal event. The analysis is primarily done using the survival function defined as follows:

$$S(t) = Pr(E > t). \tag{2}$$

In Equation (2), $S(t)$ is the probability of the event not happening until t (subject has survived at t). Another commonly used function in survival analysis is the hazard function $h(t)$. The hazard function is the rate of event at time t, given that it has not occurred by then. Hazard function is defined as follows:

$$h(t) = \frac{-S'(t)}{S(t)}. \tag{3}$$

$-S'(t)$ is the rate with which $S(.)$ decreases at t. It is divided by $S(t)$, the remaining mass of survival probability, because it is conditional to the survival of the subject at t. We use this analysis to calculate the expected time duration until dispersal events occur.

3.3 Problem Statement

Given a time and location, we are interested in predicting future dispersal events. Assuming we will encounter an event in the future, it is important to predict the volume of the anomalous demand. It is also important to know when the event is. Once we know the starting time of the event, we can start predicting the demand volume from that point in time onward. In other words, the most important thing about the dispersal event is its starting time, after which, we can directly start predicting the pick-up count. Therefore, we formally state the problem as follows.

Input:

— Spatio-temporal field $Z = (S, T)$.
— Historical and real-time trip records in Z.
— Weather information of Z.
— POI vectors of S.
— Significance threshold α.
— Input time and location (l, t) and future target time t_g.

Output:

— Earliest time $t_e \leq t_g$ at which a dispersal event will happen at l.
— C^p_{l,T_g}, in the case of a dispersal event at t_g, where $T_g = [t_e, t_g]$.

We state this problem for a specific time and location. However, when the method is used in practice, it can be applied to every location or a list of desired locations at current time.

3.4 Datasets

We use the trip records data of Yellow Taxis in New York City from years 2014, 2015, and 2016. This dataset contains the pick-up and drop locations and times for the passengers the taxis serve. This data is released by New York City Mayor's Office of Data Analytics as part of the city's open information initiative [1]. Yellow taxis of New York City operate in all five boroughs of the city.

The weather data is obtained from the National Centers for Environmental Information [3]. This data is recorded using two weather stations in the City area, Central Park station and the La Guardia Airport.

The POI data is obtained from Google Maps Places API [2]. This API categorizes places into 129 categories and it can be queried for a list of places within a spatial region. Each place is assigned to a number of categories by the API.

4 BASELINE: GATHERING-BASED DISPERSAL EVENT PREDICTION

In this section, we present a baseline solution that is based on the assumption that, for people to disperse in unexpectedly large numbers, they must first gather in numbers that are unexpectedly large. We use an observation of an abnormal gathering as a signal to predict a dispersal event in the future. First, we need to define gathering events.

Definition 8. There is a **gathering event** at spatio-temporal region R, if $LLR(C^d_R, B^d_R)$ is significant at α-level.

The definition of gathering events is similar to dispersal events. However, for gathering events, we consider the drop counts and baselines, instead of pick-ups.

Our goal is to predict dispersal events at location l, upon observing a gathering event there. To calculate this probability, we define a Gathering-Dispersal Pair for l.

ALGORITHM 1: Gathering-Based Dispersal (GbD) Prediction

Input: Baselines and counts in Z.
Output: Conditional dispersal probabilities for all times of day.

```
M_{|S|×|T^d|×|T^d|} = {0}
G_{|S|×|T^d|} = {0}
for l in |S| do
    for t_c in T do
        t_c^d = get_time_of_day(t_c)
        if gath_{t_g}^l then
            G[l, t_c^d] = G[l, t_c^d] + 1
            for t_g in (t_c^d, t_c^d + τ] do
                if disp_{t_g}^l then
                    M[l, t_c^d, t_g] = M[l, t_c^d, t_g] + 1
return M, G
```

Definition 9. Time periods t_1 and t_2 constitute a gathering-dispersal Pair at l, $\delta^l_{t_1,t_2}$, if $t_2 > t_1$, $t_2 - t_1 < \tau$, there is a gathering event at t_1 in l and there is a dispersal event at t_2 in l. We define Δ^l to be the set of all gathering-dispersal pairs at location l.

Then we define $\Delta^l_{t_c,t_g}$ and $\Gamma^l_{t_c}$ as follows.

Definition 10. $\Delta^l_{t_c,t_g}$ is the set of all $\delta^l_{t_1,t_2} \in \Delta^l$ where $t_1, t_2 \in T$ and t_1 is the same time of day as t_c and t_2 is the same time of day as t_g. $\Gamma^l_{t_c}$ is the set of all $t_1 \in T$ where t_1 is the same time of day as t_c and there is a gathering event at t_1 in l.

Using the above definitions, we calculate the conditional probability of a dispersal event in the future timestep t_g, given a gathering event at current time t_c at location l:

$$P(disp^l_{t_g} | gath^l_{t_c}) = \frac{|\Delta^l_{t_c,t_g}|}{|\Gamma^l_{t_c}|}, \tag{4}$$

where $gath^l_t$ means there is a gathering event at t in location l and $disp^l_t$ means there is a dispersal event at t in location l. Equation (4) calculates the probability of dispersal events at t_g by dividing the number of Gathering-Dispersal Pairs at the same time of day as t_c and t_g, by the number of gathering events at the same time of day as t_c, in the historical data. This equation produces an empirical conditional probability of a dispersal event in the future, given a gathering event is observed now.

Algorithm 1 calculates all the conditional probabilities of dispersal events for all timesteps of the day. The algorithm counts the number of gathering-dispersal pairs for all the locations and timesteps during a day. It also counts the number of all gathering events for all the locations and timesteps of a day. It stores these counts in the output matrices M and G. Given a location l, time of an observed gathering event t_c, and time of a potential dispersal event t_g, $M[l, t^d_c, t^d_g]$ contains the number of all gathering-dispersal pairs that have happened during that time of day. And $G[l, t^d_c]$ contains the number of all gathering events at l at timestep t^d_c, which is the same time of day as t_c. Therefore, using M and G, we can calculate the probability of a dispersal event at t_g given there is a gathering event at t_c. We call this baseline GbD.

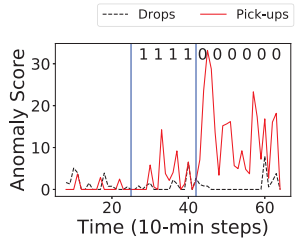


Fig. 3. An example of the values of the survival function $S(.)$ in the case of a dispersal event.

5 SURVIVAL ANALYSIS FORMULATION AND DEMAND PREDICTION

In this section, we present our survival analysis formulation by showing how the survival function values are obtained for training. Then, we present how we obtain training instances for the dispersal event demand predictor. Finally, we present our prior work, DILSA [32], as a second baseline.

5.1 Calculating the Survival Function

As mentioned earlier, we treat the dispersal event prediction problem as a survival problem. We would like to train f_s to estimate the survival function. Therefore, the output vector for f_s is the survival probabilities. In this case, the dispersal event is the death event in the survival problem. To this end, the survival function is defined as follows:

$$S(t) = Pr(E^p > t), \tag{5}$$

where E^p is the time of dispersal event. In our proposed framework, we train a model to predict $S(t)$.

At location l and time t_c, we use the following output vector:

$$\mathbf{y}_s = \left\langle S(t_c + 1), \ldots, S(t_g) \right\rangle. \tag{6}$$

Ideally, we would like to have a labeled event list for training f_s. However, such lists are not available. Therefore, we use an algorithm to obtain $S(.)$ for a given time and location (l, t_c) by determining if any dispersal event has occurred, or is underway, or will happen in the future of l from time t_c. This procedure is presented in Algorithm 2. We put a limit on the length of a dispersal event, assuming the events that are shorter than e_{min} or longer than e_{max} are not interesting. Then, we test every sub-period between $t_c - e_{max}$ and t_g that are longer than e_{min}, using Definition 6. The survival value will be set to one before a dispersal event and to zero after the start of the dispersal event in the future. For example, consider Figure 3, which shows the dispersal event of Figure 1(b). The first vertical line is the current time, and the second vertical line is the starting time of the dispersal event. The survival function is set to 1 before the start of the event and is set to zero afterwards. Algorithm 2 calculates the survival function. $(t_c - e_{max}, t_c + t_g)$ has exponential number of sub-periods. However, we are only interested in the earliest dispersal event, because the survival function will be zero afterwards. Algorithm 2 takes advantage of this fact and runs in $O(nm)$, where n is the length of time being searched $(end - start)$ and m is the number of different lengths the sub-periods can have $(e_{max} - e_{min})$.

$\mathbf{y}_s$ is obtained for every spatio-temporal grid cell in Z. They constitute the training labels for f_s that estimates the survival function.

Once we use f_s to estimate the survival function, we need to determine whether the predicted survival curve shows a dispersal event in the future or not. Assuming $S(0) = 1$, we calculate the

ALGORITHM 2: Calculate survival function (get_St)

Input: Baselines and counts, current location l, current time t_c, target time t_g
Output: Survival function $S(t)$ where $t \in (t_c, t_g]$

```
1 S(.) ← {1}; start ← t_c − e_max; end ← t_g
2 for k from e_max to e_min do
3     for t_0 ∈ [start, end − k] do
4         t_1 ← t_0 + k
5         if LLR(l, [t_0, t_1]) is significant and t_1 > t_c then
6             for t ∈ [max(t_0, t_c), t_g] do
7                 S(t) ← 0
8             return S(.)
9 return S(.)
```

probability of event at future time using the hazard function:

$$H(t) = \frac{S(t-1) - S(t)}{S(t)}. \tag{7}$$

Equation (7) calculates the cumulative hazard of event happening between $t-1$ and t given that it has not happened as of $t-1$. This value is calculated by dividing the amount of drop in the survival function from time $t-1$ to t, by the total remaining amount, which is $S(t)$, given that $S(.)$ is monotonically non-increasing. The function in Equation (7) considers two aspects of the predicted survival function at the same time. The numerator is higher if the recent drop in risk is big, thus resulting in higher hazard. Similarly, the denominator is bigger if the current risk is low. We predict an event, when value of $H(.)$ exceeds a threshold γ, which is tuned using a tuning set.

5.2 Demand Prediction

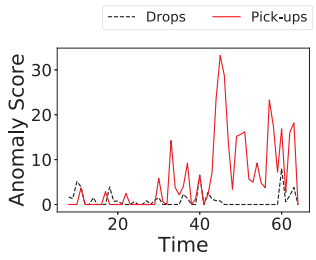
We use model f_e to predict the pick-up counts in the case of dispersal events. The output vector of f_e is as follows:

$$\mathbf{y}_e = \left\langle C^p_{l,t_c+1}, \ldots, C^p_{l,t_g} \right\rangle. \tag{8}$$

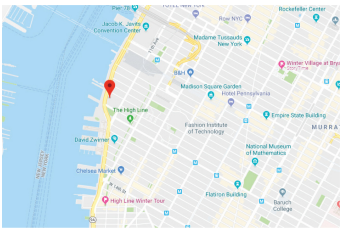
The difference between the training sets of f_s and f_e is not just their output vector. f_s is trained on instances of all locations and timesteps, while f_e is trained on instances of locations and timesteps that are involved in a dispersal event. This is a key point in our approach. The reason is, we will only use f_e to predict the pick-up counts in the case of abnormally high pick-up counts. Thus, we train it with just those instances. Algorithm 3 determines which instances should be included in the training set for f_e. By instance, we mean a pair of time and location (l, t), for which we will build features and labels and corresponds to one training point. To make sure f_e learns a full cycle of a dispersal event in its internal state, for each event, we include all the instances starting from the time when the event is first observed in the target period (line 5). For example, let t_c be current time and the target period be 4 time-steps long. If the survival function is $\langle 1, 1, 1, 0 \rangle$, then the instances of timesteps $[t_c, t_c + 4)$ will be included in the training set (line 7).

5.3 DILSA: Dispersal Event Prediction Using Survival Analysis

We theorize that every dispersal event follows other violations in repetitive patterns. For instance, a dispersal event can be preceded by a gathering event hours before, as shown in Figure 1(a), because the unexpectedly large number of people leaving now, arrived hours earlier. We take advantage of this phenomenon in developing the baseline in Section 4. However, the preceding pattern is not



(a) An anomalously high pick-up, not preceded by a drop event.



(b) Location of the pick-ups.

Fig. 4. An example of an abnormally large number of pick-ups.

necessarily as simple as a gathering event. For instance, consider the pattern in Figure 4, which shows a dispersal event around McKittrick Hotel in Manhattan, with no preceding drop event. Both Figures 1(a) and 4(a) use Equation (1) as the anomaly score. DILSA proposes to learn such complex patterns using a two-step framework. First, it predicts the time of earliest dispersal event, then it predicts the volume of the pick-up demand for the predicted event.

ALGORITHM 3: Training instances for f_e (get_Xe)

Input: Spatio-temporal field $Z = (S, T)$, Survival Function calculator (get_St).
Output: Training instances for f_e

```
. ED = An empty list
for l ∈ S do
    for t ∈ T do
        S(t) = get_St(l,t)
        if S(t_g) = 0 AND S(t_g − 1) = 1 then
            for t_c ∈ [t, t_g] do
                ED.push_back((l, t_c))
            t = t_g
return ED
```

In the first step, we are interested in the start time of the event. As mentioned earlier, we formulate this problem as a survival analysis problem [20]. The occurrence of the dispersal event is a clear analogy for the death event in this formulation of survival analysis. We propose a framework based on deep learning to train f_s that estimates the survival probabilities. To estimate the multiple values of the survival function in the future, we propose to use a multi-output deep ANN with the capability of maintaining an internal state through time, shown in Figure 5, as a learning method that natively supports multiple outputs while being capable of learning the relationships among current outputs as well as the relationship of previous states to them. Figure 5 shows that our deep learning structure uses convolutional layers in its input to learn the spatial dependencies, while LSTM [11] layers learn the dependencies in time.

We adopted the convolutional layers [38] of the network to create the capability of learning spatial relationships. Such networks have proven immensely successful in image processing [7], where objects are identified based on spatial patterns in the image. We model the geographical region as a grid and treat each individual cell as a pixel of an image. This way, the convolutional

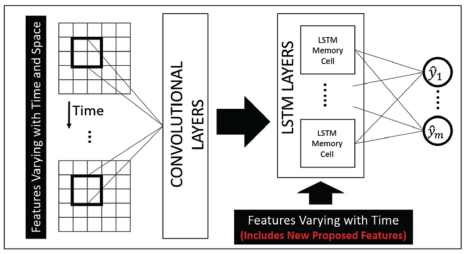


Fig. 5. Deep Learning structure used to learn spatial and temporal dependencies.

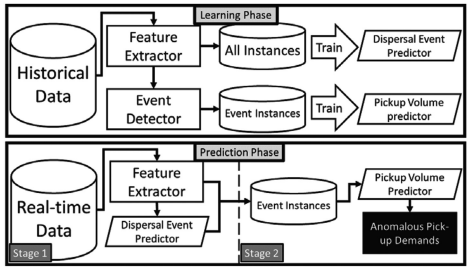


Fig. 6. Dispersal event prediction framework.

layers can learn the spatial relationships among nearby cells. The LSTM layers are used because of their capability in learning temporal relationships. This type of network maintains an internal state, known as memory, to be able to connect signals that are distant in time. Since the problem-at-hand includes features as time series, this type of layer makes it possible to learn the patterns through time that can signal the prediction of future dispersal events.

In the second step, we use f_e to predict the volume of the pick-up demand for the predicted dispersal events, also in a supervised learning framework. Since we have multiple time periods in the future, we propose to use a model with similar capabilities for the reasons mentioned above.

Figure 6 shows both phases of the proposed framework. In the learning phase, we first extract features from historical data, creating training sets for the dispersal event predictor and pick-up demand volume predictor. In stage one of the prediction phase, we extract features from real-time data and predict dispersal events. In stage two, we predict the demand volume for the predicted dispersal events. Stage one only marks the beginning of a dispersal event. This is due to the nature of the survival analysis formulation. Although SA is very powerful in determining an occurrence of an event in the future, it cannot tell us how long the event will last. However, the second stage is designed to predict the abnormally high demand, which is concerned with the size of the dispersal event. Even though stage two does not directly give a duration for the event, it addresses it by offering a prediction of the abnormal demand of the event. This prediction informs the user of the size of the abnormal demand and when the demand is predicted to go back to normal.

We discussed the training labels for f_s and f_e (survival probabilities and abnormal demand) earlier in this article. Next, we discuss how we extract the features.

5.3.1 Mobility Features. To do supervised learning, we need to have a training set with instances of inputs and outputs. In this section, we define the input variables, or the building blocks of the feature vector of the supervised learning framework. Let (l, t_c) be the current location and time. We build the variables through the following definitions.

Definition 11. **Daily profile** of (l, t_c) is defined as

$$M^l_{t_c} = \left\langle \sum_{t\in[t_d,t_c)} C^p_{l,t}, \sum_{t\in[t_d,t_c)} B^p_{l,t}, \sum_{t\in[t_d,t_c)} C^d_{l,t}, \sum_{t\in[t_d,t_c)} B^d_{l,t} \right\rangle. \tag{9}$$

The daily profile is a vector containing the sum of pick-up and drop counts and baselines since the start of current day. It is important, because a gradual gathering during the day can result in an accumulation of people in l at t_c, which might not be obvious in individual timesteps. Next, we define the recent profile of (l, t_c).

Definition 12. **Recent profile** of (l, t_c) is defined as

$$N^l_{t_c} = \left\langle C^p_{l,t_c-\tau}, C^d_{l,t_c-\tau}, \ldots, C^p_{l,t_c}, C^d_{l,t_c} \right\rangle, \tag{10}$$

where τ is a parameter.

The recent profile contains all the pick-up and drop counts of the recent τ timesteps at the current location. We define the target profile as follows.

Definition 13. **Target profile** of (l, t_c) is defined as

$$G^l_{t_g} = \left\langle B^p_{l,t_c+1}, \ldots, B^p_{l,t_g} \right\rangle, \tag{11}$$

where $(t_c, t_g]$ is the target period, i.e., the time period for which we are going to make predictions.

The target profile is the expected pick-up counts of the prediction target time period in the future.

5.3.2 Auxiliary Features. In addition to mobility features, we include features that can potentially influence the dispersal events probability. First, we define the Time Profile as follows.

Definition 14. **Time profile** of (l, t_c) is $Q^l_{t_c} = \langle d_y, d_w, t_c - t_d \rangle$, where d_y and d_w are the day of the year and day of week for t_c, and t_d is the first timestep of the current day.

Next, we include weather-related features. We argue that mobility patterns in an urban area are afffected by changes in the weather. We define Weather Profile as follows.

Definition 15. **Weather profile** of (l, t_c) is $W^l_{t_c} = \langle \omega, \eta, \zeta, \theta_{max}, \theta_{min} \rangle$, where ω is average daily wind speed, η is total rainfall of the day, ζ is total snowfall of the day, and θ_{max} and θ_{min} are the maximum and minimum temperatures of l at t_c.

Finally, for location and time (l, t_c), DILSA uses the following input vector for f_s and f_e:

$$\mathbf{x} = \left\langle Q^l_{t_c}, W^l_{t_c}, M^l_{t_c}, G^l_{t_g}, V^l, N^i_{t_c}, S^i(t_c) \right\rangle, i \in l^*, \tag{12}$$

where l^* is the surrounding area of $l = (a, b)$ defined in Section 3. Input vector $\mathbf{x}$ consists of time, weather, daily, target profile, and the POI vector (V^l) of (l, t_c) and the recent profile of (l^*, t_c), plus the current value of the survival function $(S^{l^*}(t_c))$ in l^*.

5.3.3 The Prediction Procedure. The training sets built in the previous section contain temporal and spatial dependencies. Thus, we use a Deep Artificial Neural Network that uses Convolutional layers to capture spatial dependencies and LSTM layers to capture temporal dependencies. Figure 5 shows the employed structure.

The first step in our framework is to estimate the cumulative probabilities of event, i.e., the survival curve. Then we use Equation (7) to determine whether an event is predicted.

Once a dispersal event is predicted, we predict the pick-up count for the event using f_e. Since our estimators maintain an internal state, we must make predictions in the same order as training. This is not a problem for estimator f_s, because it was trained using all the instances, which is the same order of real-time data. To train f_e, Algorithm 3 establishes a specific order that must also be followed in the prediction phase. In the training phase, we included instances when the start of the event first appears in the target period, i.e., the survival function turns to 0 in the last timestep of the target period ($S(t_g) = 0$ and $S(t_g - 1) = 1$). Therefore, we must start predicting the pick-ups using f_e once Equation (7) predicts the last timestep of the target period to be 0. However, Equation (7) might not predict the occurrence of the event until the start time gets closer. In such a case, f_e will not have its correct internal state. Therefore, to bring f_e to its correct internal state,

we feed the input vectors of previous timesteps to f_e before the input vector of current time. For example, suppose we are at time t_c and the target time period is 4 timesteps long. Then we predict a dispersal event at time $t_c + 2$. For f_e to make predictions for $t_c + 2$ and $t_c + 3$, we feed the input vectors of time $t_c - 2$, then $t_c - 1$ to f_e. Now f_e has the correct internal state to make predictions.

Algorithm 4 shows the proposed dispersal event demand predictor. First, $H(.)$ is calculated for future periods and compared with threshold γ to predict the dispersal events (lines 4–9). A value of 1 in $\hat{\mathbf{y}}_s[t] = 1$ means a dispersal event is predicted for t timesteps after current time. In the case of a predicted event, the internal state of f_e is corrected and pick-up counts are predicted (lines 10–13).

ALGORITHM 4: Dispersal event predictor (DILSA)

Input: Estimators $f_s(.)$ and $f_e(.)$, current time t_c, target time t_g, threshold γ
Output: Predicted dispersal events $\hat{\mathbf{y}}_s$, predicted counts of the predicted events $\hat{\mathbf{y}}_e$

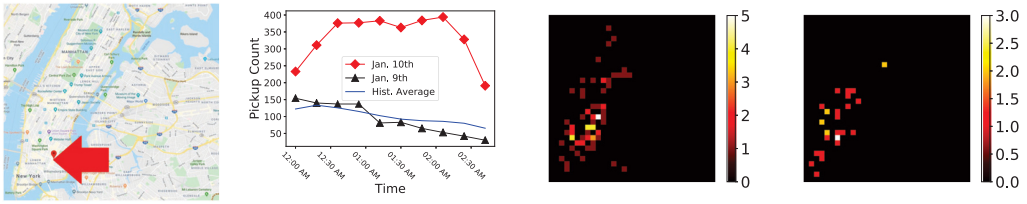
1 **for** $l \in S$ **do**
2 $\hat{\mathbf{y}}_s[l] = \{0\}$; $\hat{\mathbf{y}}_e[l] = \{-1\}$
3 $\mathbf{x}$ = construct_$\mathbf{x}$(l,t_c)
4 $St = f_s(\mathbf{x})$
5 is_event=**False**
6 **for** $i \in [1, t_g - t_c)$ **do**
7 $H = (S(i-1) - S(i))/S(i)$
8 **if** $H \geq \gamma$ **then**
9 **for** $j \in [i, t_g)$ **do**
10 $\hat{\mathbf{y}}^s_j[l] = 1$
11 is_event = **True**; event_time = i; break
12 **if** *is_event* **then**
13 Correct the internal state of f_e
14 $\hat{\mathbf{y}}_e[l] = f_e(\mathbf{x})$
15 Return $(\hat{\mathbf{y}}^s, \hat{\mathbf{y}}^e)$

6 DILSA+: PROPOSED SOLUTION BASED ON ENHANCED URBAN FEATURES

6.1 Limitations of DILSA

Our prior work, DILSA [32], employed a variety of different groups of features to capture the hidden patterns that lead to a dispersal event and abnormally high pick-up demands. We used one global model for each of the tasks of predicting the dispersal events and predicting pick-up demand. We acknowledged that different locations are likely to have different patterns and one model might not be able to distinguish between them. We added POI features to help the model distinguish between different "types" of locations based on the number of POIs of each category at each location. This way we argued that different POIs result in different mobility patterns and the model learns such differences.

However, there are two major limitations with this approach: (1) POI features are constant through time, as they consist of already existing establishments in the location. This static nature of POI features will fail to distinguish between mobility patterns at different periods of time. For instance, if two locations with similar POI features demonstrate different mobility patterns during different periods of a day, the model will not be able to capture those differences by only relying



(a) Location of the example in Lower Manhattan.

(b) Pickup counts for the five-hour period after 11:30 PM on Jan 10th and Jan. 9th.

(c) Origin locations to the example location on Jan. 10th.

(d) Origin locations to the example location on Jan. 9th.

Fig. 7. An example of varying pickup patterns and origin locations for an area in Lower Manhattan.

on POI features (refer to the example in Section 6.2). (2) There is no guarantee that POI features are directly related to mobility patterns. For instance, locations A and B with similar number of hotels will have similar POI features. However, if the hotels in A primarily serve business customers and hotels in B primarily serve tourists, the two locations will have very different patterns of pick-up and drop-offs.

To address these limitations, in this section, we present DILSA+ as a significant extension to DILSA. In the following sections, we first address the first limitation by creating features as a result of recent interaction of mobility and POIs. Then, we address the second limitation by creating features directly from the mobility patterns that capture the characteristics of the location directly from its mobility patterns.

6.2 POI-Mobility Interaction Over Time

When learning pick-up and drop-off patterns, it is important to take into account what "type" of places exist in an area. The reasoning is that the type of POIs in an area affects the volume and timing of arrivals and departures. However, the people arriving in an area with certain POI features do not appear out of nowhere. They travel to their destination from other areas and their origin might determine the nature of their interest in the destination. Moreover, the origin locations to a particular area change over time. Incorporating information of origin locations to a location can help learn patterns that are not detectable by only considering the POI features of that location. For instance, consider the example in Figure 7. This example shows different pickup patterns for a location at the same time of day. Figure 7(a) shows the example location on the map. Figure 7(b) shows the pickup counts for the 5-hour period after 11:30 PM at the same location on January 10th and 9th (both weekdays). The figure shows two very different pickup patterns for the same location and same time of day. Figure 7(c) shows the heatmap of the origin locations that ended up at the example location at 11:30 PM on January 10th, 2014. Figure 7(d) shows the heatmap of the same data on January 9th, 2014. These two heatmaps show very different patterns, despite the fact that they show the same information for the same location at the same time of day. Our theory is that considering the origin locations to a location can help capture variations such as the one in Figure 7(b).

To make it possible to capture those variations, we propose a new group of features that formulate the interaction between mobility and POIs. We call this group of features POI-Mob. POI-Mob features are obtained from the POI features of all recent origin locations to a particular location. To obtain these features, we first define source the POI vector.

Definition 16. V_s is a source POI of location l at time t, if there is a trip (l^s, t^s, l^d, t^d) where $l^d = l$ and $t - t^d < \tau$. ζ_t^l is the set of all source POI vectors to location l at time t.

POI-Mob features are obtained using the following equation:

$$\xi_t^l = \sum_{V \in \zeta_t^l} V. \tag{13}$$

In other words, the Mob-POI feature vector ξ_t^l is the sum of all POI vectors of origins from which a traveler arrived at l in the past τ timesteps from t.

6.3 Mobility Characteristics of Locations

As mentioned before, the models need to be able to distinguish between locations so that they can learn different patterns associated with them. We proposed both POI and POI-Mob features to differentiate between locations. However, these features are both based on the types of POIs in the area. Although the POIs affect the mobility patterns such as the patterns of pick-up count time series, this effect is indirect. In this section, we would like to develop features directly from these time series to be able to explicitly differentiate between locations based on their pick-up patterns. We call these features Mobility Characteristics features.

Locations with similar time series of pick-up counts must have similar Mobility Characteristics features. Moreover, since the models operate in real-time, there is no explicit "time zero," thus perfectly aligned time series similarity measures, such as correlation coefficient and Euclidean distance, will potentially miss important similarities. Therefore, we use **Dynamic Time Wrapping** (**DTW**) [5] to capture similarities in the fluctuations of pick-up counts time series among different locations. It is important to note that DTW will not work effectively in the case of data sparsity (many zeros in the time series). However, that is not the case in our dataset, as we use a large dataset with at least tens of pickups for a location during each time period.

DTW belongs to a group of measures called elastic dissimilarity measures [33]. It works by optimally aligning the two time series to minimize a cost function, which is usually a local dissimilarity (or distance) measure. It has been used in many applications such as building decision trees [26], detecting similar shapes [4], time series matching in medical applications [30], and so on. DTW employs a dynamic programming paradigm that can be shown by the following equation [28]:

$$D_{t_1,t_2} = f(C_{t_1}^{l_1}, C_{t_2}^{l_2}) + \min\{D_{t_1,t_2-1}, D_{t_1-1,t_2}, D_{t_1-1,t_2-1}\}. \tag{14}$$

D_{t_1,t_2} is the DTW distance at times t_1 and t_2 for the corresponding time series. $f()$ is the local dissimilarity measure (usually Euclidean distance). If the study period is from time 0 to T, $D_{T,T}$ is the DTW distance between two given time series. Next, we define a similarity measure based on DTW, called Normalized Wrapping Similarity:.

Definition 17. If D^{max} is the maximum DTW distance among all pairs of time series, Normalized Wrapping Similarity of two time series is defined as

$$NWS = 1 - \frac{D_{T,T}}{D^{max}}. \tag{15}$$

NWS gives us a normalized similarity score for every pair of locations, based on their pick-up time series. Matrix W represents the pairwise similarities among locations. Element W_{ij} is the NWS of locations i and j.

Next, we need to transform these pairwise similarities into feature vectors. In the space of these feature vectors, similar locations should be "closer" to each other than non-similar locations. In other words, similar locations should cluster together in the space of Mobility Characteristics features.

To create these features, we use ideas from Spectral Clustering [24]. This technique first creates a lower dimension representation of the instances using their similarity matrix. Then it performs

clustering on the low-dimension representation. We apply this technique to the similarity matrix obtained using NWS and use the low-dimension representation as the Mobility Characteristics feature vector. Algorithm 5 shows the detailed steps of how these features are obtained. First, the algorithm constructs the similarity matrix (lines 2–6). Then, it builds the Laplacian matrix L (lines 7–9). Last, the top n eigenvectors of L are calculated and row-normalized to form K, which contains the n-dimensional representation of every location based on their pick-up count time series.

ALGORITHM 5: Obtain Mobility Characteristics Features

Input: Pick-up time series of all locations (C). Feature length (n).
Output: Mobility Characteristics featurs for all locations (K).

1 $W_{|S|\times|S|} \leftarrow \{0\}$; $K_{|S|\times n} \leftarrow \{0\}$; $Y_{|S|\times|S|} \leftarrow \{0\}$
2 **for** $l \in S$ **do**
3 **for** $j \in S - l$ **do**
4 $W[l, j] \leftarrow DTW(C^l, C^j)$
5 $D^{max} \leftarrow \max(W)$
6 $W \leftarrow 1 - W/D^{max}$
7 **for** $l \in S$ **do**
8 $Y[l, l] \leftarrow \text{sum}(W[l, 0 \rightarrow |S|])$
9 $L \leftarrow I - Y^{-1/2} \times W \times Y^{-1/2}$
10 $K \leftarrow$ get_top_eigenvectors(L, n)
11 $K \leftarrow$row_normalize(K)
12 **return** K

7 EVALUATIONS

7.1 Settings and Baseline Solutions

We use the trip records of Yellow Taxis in New York City from years 2014, 2015, and 2016. This dataset contains the pick-up and drop locations and is released by New York City Mayor's Office.[1] It is important to note that DILSA+ is not limited to taxi-related applications. It is designed to take any trip records (arrival and departure records) of any mode of transportation and predict the events and the transportation demands resulting from all transportation modes. Moreover, focusing on a particular mode or all the modes can be valuable to a specific class of users. For instance, a focus on taxi records can be of interest to ride-sharing companies and a focus on bike rental records can be of interest to bike-sharing businesses. On the other hand, focusing on all the transportation modes can be of interest to the city administration. As mentioned, DILSA+ is equipped to be applied to all those scenarios.

The weather data is obtained from the National Centers for Environmental Information[2] from two weather stations, Central Park and the La Guardia Airport. The POI data is obtained from Google Maps Places API,[3] which assigns POIs into one or more of 129 categories. We partition the New York City area into a grid of 32×32 with cell size of 400×400 meters. We use 30-minute timesteps. Every record is mapped into the grid to obtain counts and baselines. The values of weather profile for each spatio-temporal grid cell is an average of the measurements reported by

[1] https://opendata.cityofnewyork.us/overview/.
[2] https://www.ncei.noaa.gov/.
[3] https://developers.google.com/places/.

Table 1. Parameter Settings

λ	α	$t_g - t_c$	τ	e_{min}	e_{max}
4	0.001	5 hrs	5 hrs	30 min	5 hrs

Table 2. Choice of Network Parameters

	Convolutional Layers	LSTM Layers
Number of Nodes/Filter Size	$7 \times 7, 8 \times 8, 9 \times 9$	32, 64, 128, 256
Number of Layers	2, 4, 6	1, 2, 3

the two stations, weighted inversely by their distance. We train the models using data in year 2014 and evaluate using data in 2015 and 2016. All datasets are standardized by subtracting the minimum and dividing by the maximum value of each feature. The test sets are standardized using parameters from the training set. Obtaining the ground truth for all dispersal events is challenging and requires labor-intensive search of public records. Moreover, not all events are publicly announced. To overcome this challenge in a meaningful way, we obtain the ground truth of the dispersal events by applying Definition 6 to true pickup counts in our dataset. This is consistent with our formulation and the goal of this research in predicting events resulting from abnormal patterns of demand. Table 1 shows our parameter settings.

In Table 1, $t_g - t_c$ is the duration of the target period. Given the half-hour timesteps and the size of the grid, we have $32 \times 32 \times 48$ (timesteps per day) $\times 6$ (weeks) $\times 7$ (days per week) $\approx$ 2 million training and testing instances. One week of the training data is used for validation throughout the training epochs. We perform the training for 3,000 epochs for all models and pick the model from the epoch that performs best on the validation set. One week of the testing set is used for tuning the threshold for Equation (7). The class ratio is 92% negative (no event). We use 20 features to capture the mobility characteristics of our 1,024 locations in the grid. Our Deep Learning Network uses two convolutional layers with window size of 9×9, 1 LSTM layer of 69 memory cells, and 10 output nodes. These parameters were chosen based on a grid search of parameters. Table 2 shows the searched values. We compare DILSA+ to our prior work as a baseline (DILSA [32]). We use three additional baselines for comparison. First, we compare with the GbD we presented in Section 4. Second, we use the state-of-the-art taxi demand prediction DMVST-Net [34]. We apply our definition of the dispersal event to the predicted count by DMVST-Net. If it satisfies the definition, we mark it as a predicted dispersal event. In addition, to show the impact of features, we define the baseline +Spectral. This baseline is DILSA with Mobility Characteristic features added. The training and testing sets and the tuning process for all the deep networks are equal.

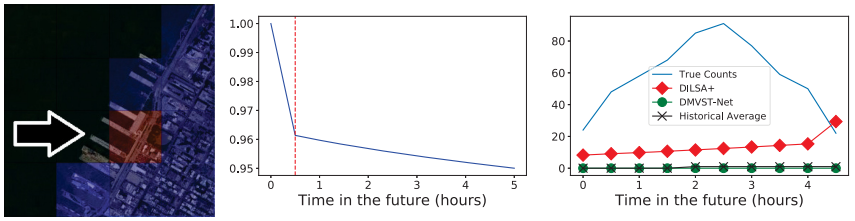
The models were trained using the stochastic gradient descent method proposed by [15]. The training and prediction times for both of the models used in DILSA+ are presented in Table 3. For the first model, the time to predict for whole region is the time the model takes to estimate a survival curve for every grid cell and calculate Equation (7). For the second model, the time to predict for whole region is the time the model takes to predict demand, if all grid cells in the region were predicted to have a dispersal event. The times for DILSA are not significantly different, as the majority of the time is spent on training the convolutional layers and DILSA+ is not different from DILSA in that aspect.

7.2 Case Studies

Here, we present two of the predicted events from year 2016. On March 19th, 2016, we predicted a dispersal event at 1:00 PM around an exhibition center in Pier 92/94 in Manhattan. We predict the

Table 3. Training and Predictions Times

	Training Time	Time to Predict for Whole Region
Event Predictor	742 (s) × 500	0.22 (s)
Event Demand Predictor	36 (s) × 500	< 0.1 (s)



(a) Event location. (b) Case 1 survival curve. (c) Predicted pick-up counts vs. true counts.

Fig. 8. First case study (best viewed in color).

event 2.5 hours before (at 11:30 AM). Public records show a home design exhibition at the time.[4] Figure 8(b) shows the predicted survival curve at 11:30 AM. The red vertical line is the predicted time of the dispersal event, which is inferred by Algorithm 4. Figure 8(c) shows the predicted counts by the baseline and the proposed method. The proposed method successfully predicts the increase, while the baseline DMVST-Net [34] stays with the historical average, missing the spike in demand.

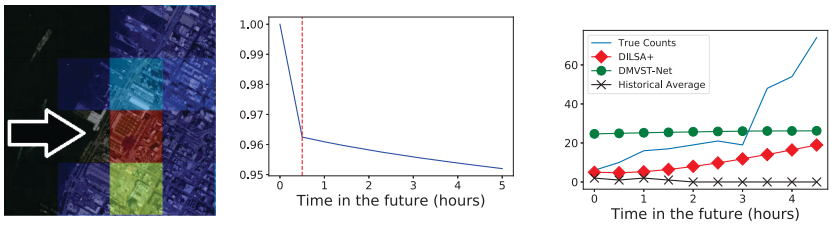
We also predicted a dispersal event around 12:30 PM on June 26th, 2016, at Jacob K. Javits Convention Center, 2.5 hours before. Public records show there was a food show at the convention center.[5] Figure 9(b) shows the predicted survival curve and the event prediction time, indicated by the vertical red line. It is worth noting that "elbow" in the curve has become visible because the y-axis is zoomed-in in values close to 1. In fact, the curve never drops below 0.95, but DILSA+ is able to predict the event because the threshold on the hazard function (Equation (7)) is properly tuned. Figure 9(c) shows the proposed method predicts an increasing trend following the trend of the true pick-up counts and counter to the historical average. The baseline DMVST-Net [34] predicts high counts; however, it predicts a slight drop, which is the opposite of the trend of the true counts and the prediction offered by DILSA+. Figures 8(a) and 9(a) show heatmaps of LLR scores (Equation (1)) based on the true counts in the predicted periods. The black arrows show the verified locations. The figures show a clear hotspot of pick-ups. Overall, these two case studies demonstrate examples of DILSA+ successfully predicting dispersal events and their corresponding demand.

These cases are presented to show examples of success for DILSA+ in real-world situations and are not an evidence of the method's overall performance. The experiments presented in the following sections are the quantitative analysis of the method's performance where we document the rate of success and failure for the model.

7.3 Experiments

In this section, we first evaluate the prediction performance of DILSA+, i.e., the performance of Algorithm 4, to predict events. We compare our results with the baselines mentioned above. Baseline

[4] https://architecturaldigest.com/story/architectural-digest-design-show-video.
[5] https://specialtyfood.com/news/article/2016-summer-fancy-food-show-largest-ever/.



(a) Event location. (b) Case 2 survival curve. (c) Predicted pick-up counts vs. true counts.

Fig. 9. Second case study (best viewed in color).

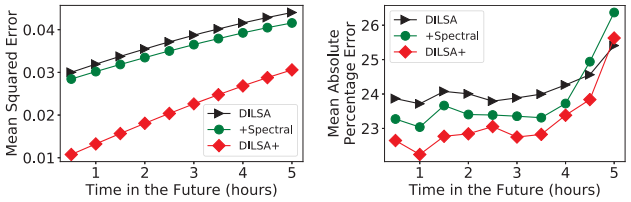
Table 4. Impact of Features on DILSA+ and Comparison With Baselines

	DILSA+	+Spectral	DILSA [32]	DMVST-Net [34]	GbD
Precision	**0.88**	0.38	0.37	0.32	0.26
Recall	0.53	0.73	0.69	0.35	**0.75**
F1-score	**0.66**	0.50	0.49	0.34	0.38

DMVST-Net predicts taxi demand. We apply Definition 6 to the predicted counts to determine if there is a dispersal event. Table 4 shows that DILSA+ outperforms all the baselines in terms of F1-score (0.66). A prediction is considered a true positive if the predicted event period overlaps with the true event period. The results show the proposed survival analysis method predicts dispersal events with high accuracy. This means that POI-Mob and Mobility Characteristics features have improved the event prediction performance. This is better demonstrated by comparing DILSA to +Spectral and finally DILSA+. It is clear how each contribution improved the accuracy. Although the results show a better recall for DILSA, it does not mean that DILSA outperforms DILSA+ in terms of recall. The measures, precision and recall are not fixed and change based on the threshold that is tuned for Equation (7). For the purpose of this experiment, the threshold is tuned to maximize the F1-score. The fact that DILSA+ has a higher F1-score means it has a superior ability to discriminate, given that it gives equal importance to precision and recall. However, in the case of an application of this method where a high recall is more valuable (for instance, emergency response), the threshold can be tuned to increase the recall at the expense of precision. For instance, consider the best achievable precision for DILSA+, which is 0.88 as opposed to 0.64 for DILSA. The best achievable recall is 1 for both methods. This means that DILSA+ provides much wider range for precision and recall for the user that can be achieved by adjusting the threshold based on Equation (7).

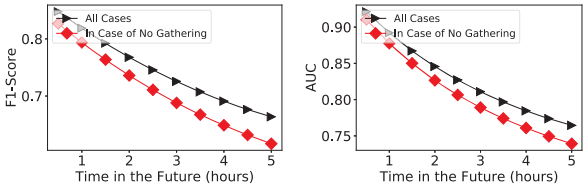
Second, we evaluate the impact of new contributions, POI-Mob and Mobility Characteristics features, on the performance of survival function estimator and the demand predictor. We use **Mean Squared Error** (**MSE**) and **Mean Absolute Percentage Error** (**MAPE**) as the measures on a test set. The results in Figure 10 show that the new contributions have significantly improved the test error of both models. Moreover, the results in Figure 10(a) show that the reason DILSA+ performed better in the previous experiment is that the survival function estimator performs considerably better than the baselines.

Third, we examine the model's performance in the case of no gathering events in the past. As we discussed earlier, one potential way to predict dispersal events is to monitor the occurrence of gathering events. The theory is that for a dispersal event to happen, we need a gathering event ahead of the time. However, we argue that the signal to predict dispersal events is not that clear and simple. To test this theory, we developed the Gathering-based Dispersal event predictor in



(a) Survival function estimator. (b) Abnormal demand predictor.

Fig. 10. Impact of choice of features on performance.



(a) F1-Score. (b) AUC.

Fig. 11. Performance of DILSA+ in the case of no gathering events, where the F1-score and AUC for GbD would be *zero* due to not observing a gathering event.

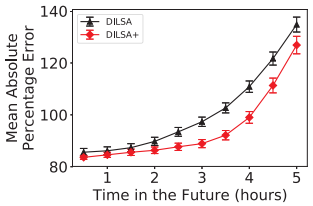


Fig. 12. Performance of the proposed pick-up counts predictor vs. baselines, on predicted events.

Section 4. This method predicts the dispersal events given that there is a gathering event. GbD predicts no dispersal events, in the case of no gathering events. In other words, GbD relies on the observation of a gathering event for its predictions. To demonstrate that DILSA+ does not rely on such observations, we evaluate its accuracy in cases where there were no gathering events observed ahead of time. To set up this experiment, we excluded all the instances of dispersal events where there was a gathering event within 5 hours before the dispersal event. This way, we are making sure that DILSA+ will not use a gathering event to predict the dispersal event. Figure 11(a) shows that the F1-score is slightly lower in such cases. Figure 11(b) shows a similar story. This experiment shows that DILSA+ is robust in the challenging case of no gathering events.

Lastly, we compare our demand predictor to the demand predictor in DILSA in the case of predicted dispersal events. In prior work [32], we showed that DILSA outperforms state-of-the-art demand prediction methods in the case of dispersal events (see the Appendix). In this experiment, we show that DILSA+ outperforms DILSA in the case of predicted events. Figure 12 shows the mean absolute percentage error of both DILSA+ and DILSA. We can see DILSA+ stays lower, longer in the future. It is important to note that the performance of DILSA in Figure 12 differs from the performance reported in [32]. This difference is because the experiments in the prior work were performed on true events. In this article, we have evaluated the performance on event instances predicted by DILSA+. Since the set of predicted events contains false positives, the accuracy is lower. Moreover, this experiment evaluates the demand predictor in the context of the whole DILSA+ framework, as it applies the demand predictor to the events that are predicted by stage one of DILSA+ and show its considerable improvement. These results, along with the results of the previous experiments, tell a full and consistent story to demonstrate the superior capabilities of DILSA+ as a whole framework over its baselines.

7.4 Summary and Discussion of Results

The results for predicting dispersal events using the proposed survival analysis framework shows a high F1-score. We have obtained this F1-score by tuning a threshold on Equation (7). This score

puts equal importance on both precision and recall. However, based on what application our method will be used for, either precision or recall might end up being more important than the other. DILSA+ is designed in a way that can be tuned to put more importance on either of these measures. For instance, a ride-sharing company would be more interested in **high precision** than recall, because those businesses are already making a profit by following the regular patterns and would only be interested in disrupting their usual routine if the occurrence of the dispersal event is predicted with very high precision. On the other hand, if our method is used for public safety applications, the users would be interested in very **high recall**, because the consequences of safety risks are potentially so high that the public safety officials cannot afford missing a single incident. Therefore, they would put higher importance on recall.

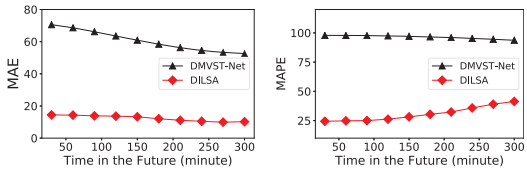
We argued that observing a gathering event at a location can potentially signal a dispersal event in the future. We developed a GbD to test this hypothesis. Our experiments show that this method has very low accuracy, because we also showed that not all preceding patterns to a dispersal event are that simple and there is no guarantee that a dispersal event will follow a gathering event. In the third experiment, we showed that our method's accuracy is still high when there is **no observation of a gathering** event, i.e., cases where GbD will have no prediction capability.

The experiments for **demand prediction** accuracy show that DILSA+ has a better accuracy than DILSA in the case of a predicted event. In our prior work [32], we showed that DILSA outperforms state-of-the-art demand prediction algorithms in the case of dispersal events. Those results are included in the Appendix of this article as well. Therefore, DILSA+ will also outperform those algorithms, because the first, second, and last experiments show that the new contributions (Mob-POI and Mobility Characteristics) have considerably improved the accuracy of the method in both event prediction and demand prediction.

DILSA+ is designed to be deployed in a real-time setup. At every timestep, the deployed system needs to gather all the time-variant features and run the prediction procedure in Algorithm 4. For this deployment to be successful, DILSA+ needs to have a reasonable **running time**. In other words, all these tasks should complete within the timestep. Given our current timestep of 30 minutes and the time to run the prediction procedure for one timestep is below 1 second, the deployment of DILSA+ in real-time is comfortably feasible.

8 CONCLUSIONS

This article solved the problem of predicting dispersal events. Such events are defined as situations where a large number of people leave an area in a short period. The solution for this problem has value to businesses and city administration and management. In our prior work [32], we proposed a framework to learn spatial and temporal patterns to predict such events. We formulated the dispersal event prediction as a survival analysis problem and proposed a two-stage framework (DILSA), where a supervised model predicted the probability of a dispersal event. In the in which case such an event was predicted, DILSA used another supervised model to predict the volume of the event, in terms of number of taxi pick-ups. However, different locations have different patterns and our existing features were constant through time and failed to distinguish between mobility patterns through time. Moreover, no features were included to directly represent the locations based on their mobility patterns. In this article, we proposed a method to capture the interaction between POI information and mobility patterns and we created vector representations of locations based on their mobility patterns. We evaluated our proposed framework, called DILSA+, by conducting extensive case studies and experiments on a real dataset from 2014 to 2016. Our method outperformed our prior work and the baselines in terms of dispersal event prediction as well as abnormal demand prediction.



(a) MAE. (b) MAPE.

Fig. 13. Performance of the proposed pick-up counts predictor vs. baselines, on events.

APPENDIX

In this Appendix, we present an experiment from prior work [32]. The demand predictor is compared to DMVST-Net in the case of dispersal events. Both models were trained on the same dataset from 2014 (all events) and tested on data from 2015, both with 30 epochs. Figure 13 shows MAE and MAPE in future timesteps. Figure 13 shows DILSA outperformed the baseline in the case of a dispersal event. This experiment shows methods proposed to capture the regular pattern of taxi demand are not reliable in the case of dispersal events.

REFERENCES

[1] 2017. NYC Open Data—Overview. Retrieved January 26, 2018 from https://opendata.cityofnewyork.us/overview/.
[2] 2018. Google Places API. Retrieved February 11, 2018 from https://developers.google.com/places/.
[3] 2018. National Centers for Environmental Information. Retrieved February 11, 2018 from https://www.ncei.noaa.gov/.
[4] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2005. Warp: Accurate retrieval of shapes using phase of Fourier descriptors and time warping distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 1 (2005), 142–147.
[5] Donald J. Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, Vol. 10. Seattle, WA, 359–370.
[6] Feng Chen and Daniel B. Neill. 2014. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1166–1175.
[7] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*. IEEE, 3642–3649.
[8] Neema Davis, Gaurav Raina, and Krishna Jagannathan. 2016. A multi-level clustering approach for forecasting taxi travel demand. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC'16)*. IEEE, 223–228.
[9] Daizong Ding, Mi Zhang, Xudong Pan, Min Yang, and Xiangnan He. 2019. Modeling extreme events in time series prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
[10] Minh X. Hoang, Yu Zheng, and Ambuj K. Singh. 2016. FCCF: Forecasting citywide crowd flows based on big data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 6.
[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[12] Liang Hong, Yu Zheng, Duncan Yung, Jingbo Shang, and Lei Zou. 2015. Detecting urban black holes based on human mobility data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 35.
[13] Amin Vahedian Khezerlou, Xun Zhou, Lufan Li, Zubair Shafiq, Alex X. Liu, and Fan Zhang. 2017. A traffic flow approach to early detection of gathering events: Comprehensive results. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 6 (2017), 74.
[14] Amin Vahedian Khezerlou, Xun Zhou, Ling Tong, Yanhua Li, and Jun Luo. 2019. Forecasting gathering events through trajectory destination prediction: A dynamic hybrid model. *IEEE Transactions on Knowledge and Data Engineering* 33 (2019), 991–1004.
[15] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.
[16] Martin Kulldorff. 1997. A spatial scan statistic. *Communications in Statistics-Theory and Methods* 26, 6 (1997), 1481–1496.

[17] Martin Kulldorff, Richard Heffernan, Jessica Hartman, Renato Assunçao, and Farzad Mostashari. 2005. A space-time permutation scan statistic for disease outbreak detection. *PLoS Medicine* 2, 3 (2005), 216.
[18] Zhongmou Li, Hui Xiong, and Yanchi Liu. 2012. Mining blackhole and volcano patterns in directed graphs: A general approach. *Data Mining and Knowledge Discovery* 25, 3 (2012), 577–602.
[19] Yu Liu, Baojian Zhou, Feng Chen, and David W. Cheung. 2016. Graph topic scan statistic for spatial event detection. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 489–498.
[20] Rupert G. Miller, Jr. 2011. *Survival Analysis*. Vol. 66. John Wiley & Sons.
[21] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. 2013. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1393–1402.
[22] Naoto Mukai and Naoto Yoden. 2012. Taxi demand forecasting based on taxi probe data by neural network. *Intelligent Interactive Multimedia: Systems and Services*. Springer, 589–597.
[23] Daniel B. Neill. 2009. Expectation-based scan statistics for monitoring spatial time series data. *International Journal of Forecasting* 25, 3 (2009), 498–517.
[24] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*. 849–856.
[25] Maya Okawa, Tomoharu Iwata, Takeshi Kurashima, Yusuke Tanaka, Hiroyuki Toda, and Naonori Ueda. 2019. Deep mixture point processes: Spatio-temporal event prediction with rich contextual information. arXiv:1906.08952.
[26] Juan J. Rodríguez and Carlos J. Alonso. 2004. Interval and dynamic time warping-based decision trees. In *Proceedings of the 2004 ACM Symposium on Applied Computing*. ACM, 548–552.
[27] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 851–860.
[28] Joan Serra and Josep Ll Arcos. 2014. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems* 67 (2014), 305–314.
[29] W. Nick Street. 1998. A neural network model for prognostic prediction. In *1990 IJCNN International Joint Conference on Neural Networks*. 540–546.
[30] Paolo Tormene, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. 2009. Matching incomplete time series with dynamic time warping: An algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine* 45, 1 (2009), 11–34.
[31] Amin Vahedian, Xun Zhou, Ling Tong, Yanhua Li, and Jun Luo. 2017. Forecasting gathering events through continuous destination prediction on big trajectory data. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'17)*. ACM, New York, NY, Article 34, 10 pages. https://doi.org/10.1145/3139958.3140008
[32] Amin Vahedian, Xun Zhou, Ling Tong, W. Nick Street, and Ynahua Li. 2019. Predicting urban dispersal events: A two-stage framework through deep survival analysis on mobility data. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI.
[33] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. 2013. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery* 26, 2 (2013), 275–309.
[34] Jun Xu, Rouhollah Rahmatizadeh, Ladislau Bölöni, and Damla Turgut. 2017. Real-time prediction of taxi demand using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems* 19, 8 (2017), 2572–2581.
[35] 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:1802.08714.
[36] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Tim Hanratty, and Jiawei Han. 2017. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 595–604.
[37] Kai Zhang, Zhiyong Feng, Shizhan Chen, Keman Huang, and Guiling Wang. 2016. A framework for passengers demand prediction and recommendation. In *2016 IEEE International Conference on Services Computing (SCC'16)*. IEEE, 340–347.
[38] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. 1990. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics* 29, 32 (1990), 4790–4797.
[39] Kai Zhao, Denis Khryashchev, Juliana Freire, Cláudio Silva, and Huy Vo. 2016. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In *2016 IEEE International Conference on Big Data (Big Data'16)*. IEEE, 833–842.
[40] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2015. Multi-task learning for spatio-temporal event forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1503–1512.

[41] Xiangmin Zhou and Lei Chen. 2014. Event detection over Twitter social media streams. *The VLDB Journal* 23, 3 (2014), 381–400.

[42] Xun Zhou, Amin Vahedian Khezerlou, Alex Liu, Zubair Shafiq, and Fan Zhang. 2016. A traffic flow approach to early detection of gathering events. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 4.

Received December 2019; revised July 2020; accepted May 2021