Using Deep Learning for Trajectory Classification

Nicksson C. A. de Freitas, Ticiana L. Coelho da Silva, José Antônio Fernandes de Macêdo, Leopoldo Melo Junior and Matheus Gomes Cordeiro
Insight Data Science Lab, Fortaleza, Brazil

Keywords: Trajectory, Classification, Machine Learning, Deep Learning.

Abstract: The ubiquity of GPS-enabled smartphones and automotive navigation systems connected to the Internet allows us to monitor, collect, and analyze large trajectory data streams in real-time. Trajectory classification is an efficient way to analyze trajectory, consisting of building a prediction model to classify a new trajectory (or sub-trajectory) in a single-class or multi-class. The classification trajectory problem is challenging because of the massive volume of trajectory data, the complexity associated with the data representation, the sparse nature of the spatio-temporal points, the multidimensionality, and the number of classes can be much larger than the number of motion patterns. Machine learning methods can handle trajectories, but they demand a feature extraction process, and they suffer from the curse of dimensionality. On the other hand, more recent Deep Learning models emerged to link trajectories to their generating users. Although they minimize the sparsity problem by representing the input data as an embedding vector, these models limit themselves to deal with multidimensional data. In this paper, we propose DeepeST (Deep Learning for Sub-Trajectory classification) to identify the category from a large number of sub-trajectories extracted from GPS services and check-ins data. DeepeST employs a Recurrent Neural Network (RNN), both LSTM and Bi-directional LSTM (BLSTM), which operates on the low-dimensional to learn the underlying *category*. We tackled the classification problem and conducted experiments on three real datasets with trajectories from GPS services and check-ins. We show that DeepeST outperforms machine learning approaches and deep learning approaches from state-of-the-art.

1 INTRODUCTION

The recent advances in the sensors and communication technologies and the popularity of Location-Based Social Networks (LBSNs) such as Foursquare, Twitter, and Facebook contribute to the explosive growth of trajectory data. We strongly believe that these data provide a unique opportunity for understanding the patterns and behaviors of several moving objects, such as people, animals, transportation modes, hurricanes, among others.

Trajectory data can be recorded in different formats according to device types. For instance, GPS tracking devices usually generate raw trajectories as a consecutive sequence of spatio-temporal points sorted in time (x, y, t), where x and y represent the spatial coordinates of the moving object at timestamp t. LBSNs collect information about their users' visited places, and a trajectory is a continuous sequence of check-ins or Points of Interest (POI). Trajectories collected from LBSN are called semantic trajectories. In the literature, there exist several research problems for trajectory data. In this paper, we focus on the trajectory classification.

The trajectory classification problem consists of building a prediction model to classify a new trajectory in a single-class or multi-class. The model is trained and learns the patterns (or classes) from a historical labeled trajectory (or sub-trajectory) data. Basic examples of trajectory classification are: (i) determining the transportation mode of the moving object like car, bus, bike, taxi, airplane, and train; (ii) determine what a user's next stopping point like home, school, cafe, office, and restaurant; (iii) identifying who is the user of a trajectory (TUL problem).

In general, the trajectory classification problem is challenging because of: (1) the massive volume of trajectory data continuously generated by multiple users; (2) the complexity associated with the data representation (how can we represent latitude, longitude, and timestamp features in our models without losing information?); (3) the sequence of spatio-temporal points can be sparse, for instance, LBSNs usually contain samples in days; (4) the nature of multiple dimensions: as technologies advances, more and more properties are linked to a trajectory, such as

A. de Freitas, N., Coelho da Silva, T., Fernandes de Macêdo, J., Melo Junior, L. and Cordeiro, M.
Using Deep Learning for Trajectory Classification.
DOI: 10.5220/0010227906640671
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 664-671
ISBN: 978-989-758-484-8

weather condition, POI category, adverse events, and so on; and finally; (5) the number of the classes can be much larger than the number of motion patterns (e.g., in TUL problem there may be more than one hundred users).

The trajectory classification topic has been exploited for over a decade (Lee et al., 2008). In the beginning, most of the works in the literature focus on extracting features from GPS trajectories (e.g., velocity, distance) and use them as input to machine learning classifiers such as Random Forest and Multilayer Perceptron (Patterson et al., 2003; Zheng et al., 2008). These approaches are limited to extract the best features from the spatio-temporal data and suffer from the curse of dimensionality.

More recently, studies involve Deep Learning models for Trajectory-User Linking (TUL) problem as (Gao et al., 2017; Zhou et al., 2018, 2019), and they aim at identifying and linking trajectories to their generating-users. These studies involving two Deep Learning models are limited because they cannot handle multidimensional data, but only the spatial dimension. We believe this the major limitation, considering that more properties have been linked to trajectories over the years and can improve the model's performance.

In this work, we propose the DeepeST (Deep Learning for Sub-Trajectory classification) model for identifying the category from a large number of sub-trajectories by jointly the embedding of many factors, i.e., location, time, or any features associated with a trajectory (or sub-trajectory). Our approach also employs a Recurrent Neural Network (RNN), more specifically, LSTM Schuster and Paliwal (1997) that has been extensively used to process variable-length input and can allow highly non-trivial long-distance dependencies to be easily learned. We also apply the Bi-directional LSTM (BLSTM) model to take into account an effectively infinite amount of context on both sides of a sub-trajectory position. The main reason we use RNNs is the capacity of these networks to learn complex patterns from a sequence, unlike feed-forward neural network models.

In this paper, we tackle the trajectory classification problem addressing two different classification tasks because we would like to evaluate DeepeST using GPS and LBSNs data: (1) a problem called Trajectory-User Linking (TUL), where the classification task is to link users to their sub-trajectories using LBSNs trajectories. Correlating sub-trajectories with users could help in identifying terrorists/criminals from sparse spatio-temporal data (e.g., the transient phone signals and check-ins) and is also helpful in making better, more personalized recommendations (Gao et al., 2017); (2) identify criminal patterns to link criminal activities to GPS sub-trajectories independent of the user's trajectory. We trained DeepeST with some categories, specified by the National Department of Public Security, which are related to the strategic POIs and the rules or specific constraints that offenders need to follow, e.g – they need to charge the ankle bracelet battery and to ensure the correct functioning of the equipment that transmits his/her location, that is, the signal that sends trajectory data to the police department cannot be blocked; they need to be at his/her home during the night. Figure 1 shows an example of an offender's sub-trajectories, where the green, blue and purple points represent a sub-trajectory link to a supermarket, its home and a block signal, respectively. In this case, the signal block was detected by the DeepeST based on the Spatio-temporal pattern, where the same spatial coordinates (latitude and longitude) are attributed to sequences of sub-trajectories for a long time before a sub-trajectory link to its home. In general, offenders prefer to block the signal in their homes to avoid unsuspecting behavior.

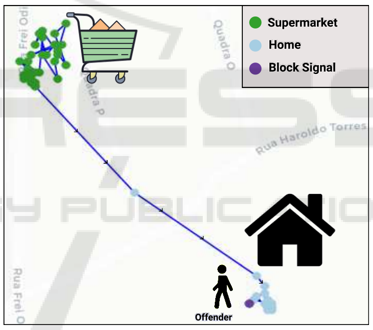

Figure 1: GPS Sub-trajectories classified by DeepeST.

The remainder of the paper is structured as follows: Section 2 presents the related works and formally defines the problem. Section 3 introduces our proposal. Section 4 discusses the experimental evaluation. And finally, Section 5 draws the final conclusions.

2 TRAJECTORY CLASSIFICATION

In this paper, we tackle the trajectory classification problem, which consists of classifying a sub-trajectory into a label or class. We claim that we classify sub-trajectories since our training set is derived from the segmentation of trajectories as we explain later. For sake of brevity, hereinafter we will use the term *trajectory classification* in place of *sub-*

trajectory classification. So, given a set of trajectories and labels, we want to build a model for predicting and assigning such labels to every sub-trajectory. The labels are a set of category features, be the users are the owner of the trajectory, the transportation modes (car, bus, bike, walk), or criminal activities.

Trajectory classification is one of the widely studied problems on trajectory pattern mining over the years. In the beginning, trajectory classification focused on detecting patterns of mobility from raw trajectories. Most of the supervised learning approaches classify trajectories or sub-trajectories into categories that represent activities (as hiking and walking) or different transportation modes (like a car, bike, and bus) (Zheng et al., 2008; Patterson et al., 2003; Fang et al., 2016). One of the first methods for trajectory classification was TraClass, proposed by Lee et al. (2008) that supports only the spatial dimension. Patel (2013) extended the TraClass to support both the spatial and the time dimensions. Basically, these works use machine learning methods demanding a feature extraction process to categorize raw trajectories or (sub-trajectories) into different motion patterns considering features that are extracted from the spatial and temporal dimensions like velocity, acceleration, and distance.

More recently, studies have investigated Deep Learning models to link trajectories to their generating users. TULER was the first model introduced in (Gao et al., 2017) for identifying and linking a large number of check-in trajectories to their generating-users using RNN based models. More specifically, TULER receives a sequence of POIs as input and represents this sequence in a new low-dimensional space (an embedding vector) similar to word embedding in natural language (Mikolov et al., 2013). Finally, the trajectory was characterized via trained RNN models to link them to their users. TULVAE was proposed in (Zhou et al., 2018) after the TULER and enhanced in (Zhou et al., 2019), a generative model to mine human mobility patterns, which aims at learning the implicit hierarchical structures of trajectories and alleviating the data sparsity problem with the semi-supervised learning. TULVAE achieved a significant performance improvement for the TUL problem in comparison to existing methods.

In summary, Machine learning methods can handle trajectories, but they demand extraction features process and suffers from the curse of dimensionality. On the other hand, Deep Learning models as TULER and TULVAE minimize the sparse data problem by representing the data in an embedding vector but only support a sequence of POI identifier. These approaches do not cope with other essential features, commonly used to describe semantic trajectories. Therefore, we propose the DeepeST model for identifying the category from a large number of raw (sub)trajectory, semantic (sub)trajectories, or any (sub)trajectory of other domains. DeepeST also employs a Recurrent Neural Network (RNN), both LSTM (Schuster and Paliwal, 1997) and Bidirectional LSTM (BLSTM), which are designed to recognize the sequential characteristics of data and thereafter using the patterns to predict the future scenario. Moreover, DeepeST is able to minimize the computational complexity since it also operates on the low-dimensional to learn the underlying *category* from the sub-trajectory data. Finally, DeepeST is capable of handling raw trajectories (collected by GPS) and semantic trajectories (generate from LBSNs) with several features, since the model is capable of receiving sequences of attributes linked to a trajectory.

2.1 Preliminaries

In this section, we will introduce some necessary notations and the basic terminology before we formally present the problem formulation of the trajectory classification problem. Table 1 presents a list of commonly used notations.

Table 1: A list of commonly used notations.

Notation	Description
A	a finite sequence of attributes linked to a trajectory
l	a location point
$L = \{l_1, ...l_m\}$	a finite set of location indexes
$T = \{t_1, ...t_w\}$	a finite set of time slot indexes
$p_i = (l_i, t_i, A_i)$	a spatio-temporal position
$TR_j = [p_1, ...p_{len_j}]$	trajectory
$S_j = [p_{q_1}, ...p_{q_n}]$	a sub-trajectory
y	a label of a sub-trajectory
$Y = \{y_1, ...y_o\}$	a set of labels
$\tilde{S} = \{S_1, ...S_z\}$	A finite set of sub-trajectories

There are several concepts for trajectory. In this work, we consider trajectory raw (from GPS tracking devices, it is a consecutive sequence of spatial coordinates ordered by time) and semantic trajectories (generated by LBSNs that collect information about visited places of their users as a consecutive sequence of check-ins or stop episodes, for instance) in a single definition, formalized as follows:

Trajectory: Let a trajectory TR_j be a sequence of points sorted in time $[p_1, .., p_{len_j}]$. Here, p_i $(1 \leq i \leq len_j)$ is a tuple (l_i, t_i, A_i), such that l_i is a location point at time t_i, and $A_i = [a_1, ..., a_m]$ is a sequence of m attributes linked to the trajectory (e.g velocity, acceleration, geographic information, among others).

For sake of brevity, the location point l_i is a spatial coordinate e.g., latitude and longitude collected from a GPS device, or l_i can refer to check-in or stop location (it can be a POI location, for instance). It is worth to mention that if a trajectory is not linked to any semantic information, then $A_i = \emptyset$.

For simplicity, in this work, we represent each location l_i composed of latitude and longitude from a trajectory TR in a spatial grid cell, however it could be in any well-defined geographical space. We also map each timestamp t_i to a time slot in $T = \{t_1, t_2, ..., t_w\}$, such that $T \in \mathbb{R}^w$. A time slot could be a regular time interval, for instance, some minutes, hours, days or weeks. Finally, in order to reduce the computational complexity and capture richer knowledge of sub-trajectory patterns from trajectories, we segment the trajectories into sub-trajectories. There are several methods for trajectory segmentation based on the shape of a trajectory, time interval, and semantic meanings (Zheng, 2015). Since trajectory segmentation is not at the core part of this work, we adopt the simplest method based on the time interval to trajectories. A trajectory $[p_{q_1}, p_{q_2}, ..., p_{q_n}]$, such that $(1 \leq q_1 < q_2 < ... < q_n \leq len_j$, where $l_k = l_{k-1}+1)$ is called a sub-trajectory of S_j. We are now ready to formulate our classification problem for sub-trajectories.

2.2 Problem Statement

Given a set of sub-trajectories $\tilde{S} = \{S_1, S_2, ..., S_z\}$, the task is to classify the category by linking each sub-trajectory $S_i \in \tilde{S}$ to a label $y \in Y = \{y_1, ..., y_o\}$.

Notice that our problem is generic, Y can be a set of transportation mode $\{car, bus, bike, walk\}$, a set of users that are the owner of the trajectory or any category feature of other domains.

3 DeepeST: DEEP LEARNING FOR SUB-TRAJECTORY CLASSIFICATION

In this paper, we propose a deep learning model, called DeepeST, to the trajectory classification problem. DeepeST receives as input a fit sequence of features contain location, time, and any other attributes annotated in a sub-trajectory.

3.1 DeepeST Architecture

DeepeST architecture is composed of embedding layers to each input, a concatenation layer, a recurrent layer (LSTM or BLSTM), and a fully connected layer with softmax as the activation function. The overview of DeepeST is illustrated in Figure 2.

DeepeST incorporates embedding layers to receives sequences from the sub-trajectory. An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words. There are two main reasons we use sub-trajectories embedding: (1) Traditional methods, such as one-hot encoding, are binary, usually sparse (mostly made of zeros), and very high-dimensional (same dimensionality as the number of distinct labels) (Chollet, 2018); (2) the frequency of location in sub-trajectories can follow a power-law distribution. In general, likewise, words embedding in natural language (Mikolov et al., 2013). Sub-trajectory embedding alleviates the curse of dimensionality and maintains the input data's proximity with similar patterns in a new dimensional space. For instance, suppose a student user has visited the POI sequence [Home, Bus Station, University], while another user has visited the POI sequence [Home, Subway, University]. Note that the embeddings of Bus Station and Subway will be similar because they happened in the same context (after Home and before University).

Deep learning models TULER and TULVAE explore only the spatial dimension to embedding vector. In these works, we explore the spatial and temporal dimensions and other different features linked to sub-trajectories. The more important features we linked to a sub-trajectory, more information to improve classification accuracy and can be used directly and hence save more time.

We used a recurrent layer that receives input from a feature vector, but DeepeST presents embedding layers to each sub-trajectory attribute. So, a concatenation layer is defined between the embedding layers and the recurrent layer to join embedding vectors in a unique input features that will be used in the recurrent layer, as shown in Figure 2.

The main reason we use a Recurrent Neural Network (RNN) is the capacity of these networks to learn complex patterns from a sequence, unlike feedforward neural networks. DeepeST employs a LSTM (Hochreiter and Schmidhuber, 1997), which has been extensively used to process variable-length input and can allow highly non-trivial long-distance dependencies to be easily learned. We also experimented DeepeST with the Bi-directional LSTM (BLSTM) model (Schuster and Paliwal, 1997), which can take into account an effectively infinite amount of context on both sides of a sub-trajectory and eliminates the problem of limited context that applies to any feed-forward model. Both LSTM and BLSTM operate at the location, time, and all attribute embedding levels to

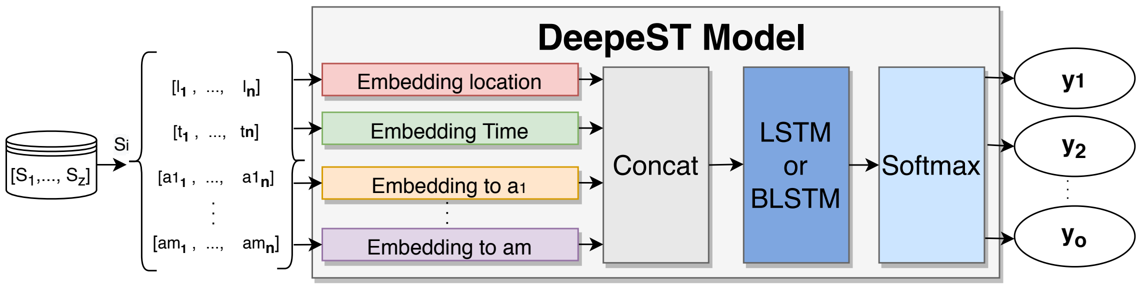

Figure 2: DeepeST model to sub-trajectory classification.

learn the underlying pattern (or label) from the sub-trajectory data.

The output of LSTM or BLSTM passes by the softmax function, which converts the recurrent layer's result into the set of probabilities to be assigned to each label. Softmax takes as input a vector of real numbers and normalizes it into a probability distribution consisting of o probabilities (for each label) proportional to the input numbers' exponentials.

After applying softmax, we have probabilities for each label $y \in Y$ on the interval $[0,1]$.

3.2 Optimization in DeepeST

Overfitting is a major problem in RNN due to a large number of weights and biases. To alleviate overfitting, we determined a dropout layer for regularization. Dropout is a strategy radically different from other approaches, since it changes the network structure itself, instead of the cost function. Suppose we have a training set X and the corresponding desired output y. Normally, we train by direct propagation of X across the network, and then the backpropagation algorithm computes the error to the gradient. When we use a layer dropout, this process is modified. We eliminate by randomly (and temporarily) some of the neurons hidden in the network, but leave the input and output neurons untouched. Heuristically, if we abandon different sets of neurons, we are training with various neural networks. Therefore, dropout can reduce overfitting, whereas other networks adapt in different ways.

4 EXPERIMENTS

In this section, we present the experimental evaluation to evaluate DeepeST in terms of quality prediction. We start by providing details about the datasets, the baseline algorithms, the evaluation metrics, followed by the experimental evaluation. For reproducibility purposes, we made the source code available on GitHub[1].

4.1 Datasets

To evaluate the performance of DeepeST for the classification trajectory problem, we conduct our experiments on three datasets: (1) a public dataset that contains check-ins of users extracted from Brightkite[2] between April 2008 and October 2010. (2) a public dataset that contains trajectories of check-in extracted from Gowalla[3] between February 2009 and October 2010; finally, (3) a private dataset of offender trajectories extracted from GPS services. To validate the models, we split the three datasets into training (70%), validation (15%), and test sets (15%). We shuffle sub-trajectory data, run the baselines algorithms ten times for each dataset, and compared the models using Accuracy, Macro Precision, Macro Recall, and Macro F1-Score.

For Brightkite and Gowalla datasets the classification task is to predict the corresponding user who generated a given trajectory. We use a segmentation based on time to created weekly sub-trajectories from each user check-in. Notice that there are no overlapping sub trajectories, there are weekly sub-trajectories for each user (label). We also created a grid covering all spatial points of trajectories with a cell of $30m^2$. We selected only sub-trajectories of users who have at least 15 weekly trajectories because we will have at least two samples for each user in the validation set and the test set. Finally, the Brightkite dataset contains 4565 sub-trajectory samples to train, 996 to validation, and 1085 to test. Gowalla dataset contains 2325 sub-trajectories sequences to train, 517 to validation, and 572 to test. Table 2 describes the attributes of trajectory points to both Brightkite and Gowalla datasets.

The Criminal dataset contains trajectories of offenders in June 2019 that moved around Fortaleza,

[1]https://github.com/nickssonarrais/ICAART2021
[2]https://snap.stanford.edu/data/loc-Brightkite.html
[3]https://snap.stanford.edu/data/loc-gowalla.html

Table 2: Description of the check-ins trajectories to Brightkite and Gowalla.

Attributes	Type	Range/example	N.
User	Nominal	{58186, ..., 58190}	{197, 100}
Weekday	Nominal	{Monday, ..., Sunday}	7
Hour	Numeric	[0, 23]	24
Index grid	Numeric	{0, ..., 46458}	{3742, 11345}
POI	Nominal	{dsda411, ..., ee8b8e}	{4085,15977}
Subtraj ID	Numeric	{0, ..., 10000}	{6646, 2335}

Ceara. The classification task is to identify criminal activities, as well as what the criminal is doing - at home, blocking the signal from the equipment that transmits his/her location, selling stolen car parts, in a hearing with the judge, among other categories. In this dataset, we randomly selected trajectories of ninety offenders. To criminal activity dataset we used a different segmentation based on time and label to avoid overlapping sub trajectories. We created sub-trajectories group each user and criminal activity for up thirty minutes. In order words, for each offender the algorithm returns sub-trajectories contains a single criminal activity for up to thirty minutes. We define a 2D grid content cells of $30m^2$ for an area around Fortaleza city. Table 3 describes the attributes of trajectory points to the Criminal dataset. Finally, the Criminal dataset contains 116255 sub-trajectories sequences to train, 24912 to validation and 24912 to test.

Table 3: Description of the GPS trajectories to the Criminal Dataset.

Attributes	Type	Range/example	N.
Offender	Nominal	{58186, ..., 58190}	90
Weekday	Nominal	{Monday, ..., Sunday}	7
Hour	Numeric	[0, 23]	24
Index grid	Numeric	{0, ..., 46458}	36690
Criminal Activity	Nominal	{home, ..., blocked signal}	9
Subtraj ID	Numeric	{0, ..., 10000}	166079

4.2 Baselines Algorithms

We compare DeepeST with four state-of-the-art approaches from the field of machine learning and deep learning classification: XGBoost (Chen and Guestrin, 2016), Random Forest (Breiman, 2001), BITULER (Gao et al., 2017), and TULVAE.

To find the optimal set of hyperparameters for each model, we apply the grid-search technique to combine several hyperparameters. For the DeepeST models (the one that uses LSTM, the one with BLSTM), BITULER and TULVAE, we keep 64 as the batch size and 0.001 as the learning rate and vary the units (**un**) of the recurrent layer, the embedding size to each attribute (**es**) and the dropout (**dp**). For TULVAE, we also vary the latent variable (**z**). We determine an early stopping callback, that is a stop training when, in our case, the accuracy has stopped improving. We set the early stopping as 20 for patience argument to minimize overfitting, i.e., the number of epochs that produced the model's accuracy with no improvement after which training should be stopped. For further details, we refer to Keras library[4].

For the XGBoost model, we vary the number of estimators (**ne**), the maximum depth of a tree (**md**), the learning rate (**lr**), the gamma (**gm**), the fraction of observations to be randomly samples for each tree (**ss**), the sub sample ratio of columns when constructing each tree (cst), the regularization parameters (**l1**) and (**l2**). We also set the early stopping round to 20 for XGBoost.

For Random Forest, we vary the number of trees (**ne**), the maximum number of features to consider at every split (**mf**), the maximum number of levels in a tree (**md**), the minimum number of samples required to split a node (**mss**), the minimum number of samples required at each leaf node (**msl**), and finally, the method of selecting samples for training each tree (**bs**).

For more details about parameters, we shows refer to Git Hub repository[5].

4.3 Performance Comparison

There are two variations for DeepeST with respect to the network layers: one with LSTM (DeepeST-LSTM) and another one with BLSTM (DeepeST-BLSTM). Our experiments tackle two main objectives: evaluate individually the DeepeST models with the state of art machine learning and deep learning models to the trajectory classification problem from check-ins, as we experimented for Gowalla and Brightkite datasets. And finally, evaluate individually Deepest variation and the machine learning approaches to GPS-based trajectories. The idea behind is to measure how resilient DeepeST is to learn the categories in comparison to the baselines.

Table 4 and 5 shows the best set of parameters from the grid search, it also summarizes the performance comparison between the variants of DeepeST, XGBoost, RandomForest, BITULER and TULVAE for the Brightkite dataset. The two best values are highlighted in **bold** and the third one is shown as *underlined*. For what concerns to DeepeST variations, a sub-trajectory S is a sequence with each of the following attributes $(ig_i, hr_i, poi_i, wk_i)$, where ig is the index grid cell and poi is the POI identifier, wk is the weekday, and hr is the hour); BITULER and TULVAE only deal with one feature, so the input is a se-

[4]https://keras.io/

[5]https://github.com/nickssonarrais/ICAART2021

quence of POI identifier as presented in Gao et al. (2017); Zhou et al. (2019). For XGBoost and RandomForest, a sub-trajectory is a unique concatenated sequence $[ig_1, ig_2, ..., wk_{n-1}, wk_n]$ with all attributes $(ig_i, hr_i, poi_i, wk_i)$.

From the results reported in Tables 4 and 5, we can see that the embedding of four features combined with RNN layers in sub-trajectory classification yields accuracy improvements over the baselines. In summary, in comparison with machine learning approaches, we can notice that DeepeST outperforms XGBoost and Random Forest across all metrics by up to 11% in Brightkite and by up 31% in Gowalla considering F1-Score. DeepeST takes advantage of the LSTM/BLSTM and operates at embedding levels to learn the underlying user categories from check-ins sub-trajectory data. It is worth to mention that RNN models (LSTM/BLSTM) are proper models to learn from temporal sequences as sub-trajectories. We can notice that DeepeST also outperforms BITULER and TULVAE across all metrics by up to 5% in Brightkite and by up 7% considering F1-score. This is because DeepeST built a more representative model using a set of variables instead of only using the points of interest identification. As we can see, only POIs identification are not sufficient to distinguish different users. It is worth noting that the results could be higher if there were more important features to separate the classes from different users in the dataset (maybe features based on external events and weather conditions). The expert's view of the application domain can be essential to increase the performance of the model. In this case, we applied only the spatial and time features (weekday, hour, index grid, and POI) extracted from the original Gowalla and Brightkite datasets. DeepeST-LSTM only preserves the past information because the only inputs it has seen are from the past. DeepeST-BLSTM run your inputs in two ways, one from past to future and one from future to past. In our experiments, DeepeST-BLSTM achieved slightly more significant results than DeepeST-LSTM for Brightkite dataset since DeepeST-BLSTM takes into account an effectively infinite amount of context on both sides of a sub-trajectory position and eliminates the problem of limited context that applies to any feed-forward model. On the other hand, DeepeST-LSTM achieved slightly more significant results than DeepeST-BLSTM for Gowalla dataset. It is important to highlight that the results between DeepeST-BLSTM and DeepeST-LSTM are very close. However, using future information can usually be easier and faster for the network to understand the next label.

Table 6 shows the best set of parameters from the grid search, it also summarizes the performance comparison between the variants of DeepeST, XGBoost, RandomForest. It is important to mention that BITULER and TULVAE were not included in the experiments since they are applied from a sequence of POI identifier (one-dimensional data) in check-ins trajectories. To DeepeST variations, a sub-trajectory S are sequence for each attribute in (ig_i, hr_i, wk_i), where ig is the Index grid, hr is the hour of day, and wk is the weekday. For XGBoost and RandomForest, a sub-trajectory is a unique concatenate sequence $[ig_1, ig_2, ..., wk_{n-1}, wk_n]$ with all attributes (ig_i, hr_i, wk_i). The two best values are highlighted in **bold** and the third one is shown as *underlined*. We can notice that DeepeST outperforms XGBoost and Random Forest on all of the metrics by up 25% considering F1-macro. DeepeST again takes advantage of the LSTM/BLSTM and operates at the location and time embedding levels to learn the underlying criminal categories from the offenders' sub-trajectory data. DeepeST-BLSTM achieved slightly more significant results than DeepeST-BLSTM for the Criminal dataset.

5 CONCLUSION

In this paper, we investigate the trajectory classification problem to classify a category from a set of labels. A category can be anything, such as transportation mode (car, bus, bike, walk), criminal activity (home, ..., blocked signal), or a user who is the owner of the trajectory. We propose a new model, called DeepeST (Deep Learning for Sub-Trajectory classification), to identify any category from a large number of sub-trajectories extracted from GPS services or generate from LBSNs. DeepeST employs a Recurrent Neural Network (RNN), both LSTM and Bi-directional LSTM (BLSTM), which operate on the low-dimensional to learn the underlying *category*. We attacked two trajectory classification tasks and conducted extensive experiments on three real datasets to evaluate DeepeST performance with state-of-the-art approaches from the field of machine learning classification − XGBoost and Random Forest − and Deep Learning − BITULER and TULVAE. DeepeST achieves more expressive values of accuracy, precision, recall, and f1-score in all experiments. As future directions, first, we aim at analyzing more features in our models (as the risk facilities (Chainey and Ratcliffe, 2013) for criminal activities and weather features to Gowalla and Brightkite datasets). Moreover, to provide operational advantages, optimization, and innovation, e.g − incorporate modules to trajectory

Table 4: Results to Brightkite.

Method	Best set of the grid search	Accuracy		Precision		Recall		F1-Score	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
DeepeST-LSTM	un:400, es:100, dp:0.5	**0,9591**	0,0018	**0,9644**	0,0048	**0,9514**	0,0025	**0,9512**	0,0033
DeepeST-BILSTM	un:100, es:100, dp:0.5	**0,9632**	0,0030	**0,9658**	0,0038	**0,9563**	0,0044	**0,9557**	0,0041
BiTULER	un:100, es:100, dp:0.5	0,9372	0,0052	0,9417	0,0055	0,9211	0,0068	0,9234	0,0066
TULVAE	un:100, es:300, dp:0.5, z:300	0,9452	0,0044	0,9439	0,0062	0,9325	0,0048	0,9308	0,0054
Random Forest	ne:200, md:30, mss:5, msl:1, mf:auto, bs:False	0,8717	0,0043	0,8744	0,0090	0,8431	0,0060	0,8440	0,0072
XGBoost	ne:2000, md:5, gm:0, ss:0.8, cst:0.5, l1:1, l2:100	0,8769	0,0047	0,8717	0,0059	0,8481	0,0063	0,8483	0,0065

Table 5: Results to Gowalla.

Method	Best set of the grid search	Accuracy		Precision		Recall		F1-Score	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
DeepeST-LSTM	un:100, es:400, dp:0.5	**0,9760**	0,0039	**0,9821**	0,0027	**0,9744**	0,0042	**0,9750**	0,0039
DeepeST-BILSTM	un:200, es:100, dp:0.5	**0,9739**	0,0038	**0,9798**	0,0034	**0,9727**	0,0040	**0,9723**	0,0044
BiTULER	un:300, es:400, dp:0.5	0,9122	0,0050	0,9274	0,0070	0,9101	0,0060	0,9078	0,0072
TULVAE	un:100, es:300, dp:0.5, z:300	0,9159	0,0085	0,9338	0,0121	0,9111	0,0096	0,9105	0,0109
Random Forest	ne:400, md:30, mss:2, msl:2, mf:sqrt, bs:False	0,7047	0,0088	0,7020	0,0139	0,6841	0,0093	0,6631	0,0109
XGBoost	ne:2000, md:10, gm:0, ss:0.8, cst:0.5, l1:1, l2:100	0,6545	0,0112	0,6393	0,0197	0,6327	0,0134	0,6143	0,0152

Table 6: Results to Criminal Dataset.

Method	Best set of the grid search	Accuracy		Precision		Recall		F1-Score	
		mean	**std**	**mean**	**std**	**mean**	**std**	**mean**	**std**
DeepeST-LSTM	un:100, es:400, dp:0.5	**0,9188**	0,0010	**0,8792**	0,0075	**0,8283**	0,0043	**0,8504**	0,0040
DeepeST-BILSTM	un:200, es:100, dp:0.5	**0,9203**	0,0013	**0,8826**	0,0071	**0,8365**	0,0052	**0,8564**	0,0026
Random Forest	ne:400, md:30, mss:2, msl:2, mf:sqrt, bs:False	0,7917	0,0002	0,6806	0,0017	0,5515	0,0010	0,5910	0,0012
XGBoost	ne:2000, md:10, gm:0, ss:0.8, cst:0.5, l1:1, l2:100	0,8121	0,0008	0,6671	0,0017	0,5765	0,0015	0,6084	0,0011

preprocessing and features creation. Finally, we aim at studying how to improve accuracy by means of other deep learning techniques, like attention mechanisms.

REFERENCES

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chainey, S. and Ratcliffe, J. (2013). *GIS and crime mapping*. John Wiley & Sons.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-Augu, pages 785–794.

Chollet, F. (2018). *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG.

Fang, S.-H., Liao, H.-H., Fei, Y.-X., Chen, K.-H., Huang, J.-W., Lu, Y.-D., and Tsao, Y. (2016). Transportation modes classification using sensors on smartphones. *Sensors*, 16(8):1324.

Gao, Q., Zhou, F., Zhang, K., Trajcevski, G., Luo, X., and Zhang, F. (2017). Identifying human mobility via trajectory embeddings. In *IJCAI*, volume 17, pages 1689–1695.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Lee, J.-G., Han, J., Li, X., and Gonzalez, H. (2008). Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Patel, D. (2013). Incorporating duration and region association information in trajectory classification. *Journal of Location Based Services*, 7(4):246–271.

Patterson, D. J., Liao, L., Fox, D., and Kautz, H. (2003). Inferring high-level behavior from low-level sensors. In *International Conference on Ubiquitous Computing*, pages 73–89. Springer.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Zheng, Y. (2015). Trajectory Data Mining. *ACM Transactions on Intelligent Systems and Technology*, 6(3):1–41.

Zheng, Y., Liu, L., Wang, L., and Xie, X. (2008). Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*, pages 247–256. ACM.

Zhou, F., Gao, Q., Trajcevski, G., Zhang, K., Zhong, T., and Zhang, F. (2018). Trajectory-user linking via variational autoencoder. In *IJCAI*, pages 3212–3218.

Zhou, F., Yin, R., Trajcevski, G., Zhang, K., Wu, J., and Khokhar, A. (2019). Improving human mobility identification with trajectory augmentation. pages 1–31. Springer.