Classification of Geolife GPS Trajectory Dataset Using Deep Learning Methods

Hasan Pourmahmoodaghababa, u1255635

Ali Hassandokht Mashhadi, u1268609

Behnam Sherafat, u1194864

1. Introduction

Trajectory data, the recorded movement of objects at certain timestamps, has been an important research theme to study human behavior. This data has different types of applications in road transportation systems analysis, such as designing public transit, and demand estimation. In this report, the authors proposed a novel method for binary/multi-class classifications of Geolife GPS trajectory dataset using deep learning techniques. In recent years, deep learning algorithms have been used in different research areas, including Text processing, Object detection, and Image captioning. In this project, a GPS trajectory dataset is utilized to classify the user labels using binary and multi-class classifications. This dataset contains the sequences of latitude, longitude, and altitude of the 182 users' locations. To efficiently develop the model, first, a novel pre-processing algorithm is used to reduce the dimension of the dataset. The dimension of the dataset have been reduced to R^{20}, R^{50}, and R^{250} using the aforementioned method. This approach significantly decreases the computational time complexity of the classification model and does a minimum impact on the accuracies. Moreover, in order to further decrease the amount of unreliable data, only users with more than 100 routes in their trajectories were selected for training the models, and they are chosen in pairs or by a group of 10 users. To be more precise, first we have chosen 40 pairs from 41 users ((user1, user2), (user2, user3), (user3, user4), …, (user40, user41)) and did a binary classification task applying above classifiers. Then we got 40 train/test errors and reported their average in Section 3. Second, we chose 32 set consisting of 10 different users ((user1, user 2, …, user10), (user2, user3, …, user11), …, (user32, user33, ..., user41)) and performed a 10-class classification task among them. Then we got 32 train/test errors and reported their average in the report.

In the following sections, first, the dataset and the preprocessing procedure is elaborated. Then, in the classification section, the implemented methods and their results are presented. In the end, the conclusion is presented.

2. Dataset

The explored dataset is the Geolife GPS Trajectory dataset released by Microsoft. The full version of the dataset could be accessed through the following link:

https://msropendata.com/datasets/d19b353b-7483-4db7-a828-b130f6d1f035

The provided dataset consists of 182 users that each one of them has a number of trajectories. Total number of trajectories is 17,621. According to the provided guide file there, these trajectories were recorded by different GPS loggers and GPS-phones, and have a variety of sampling rates. 91.5 percent of the trajectories are logged in a dense representation, e.g. every 1~5 seconds or every 5~10 meters per point. Although this dataset is widely distributed in over 30 cities of China and even in some cities located in the USA and Europe, the majority of the data was generated in Beijing, China. Figure 1 plots the distribution (heat map) of this dataset in Beijing, which is visualized by the first named author using D3 and Javascript.

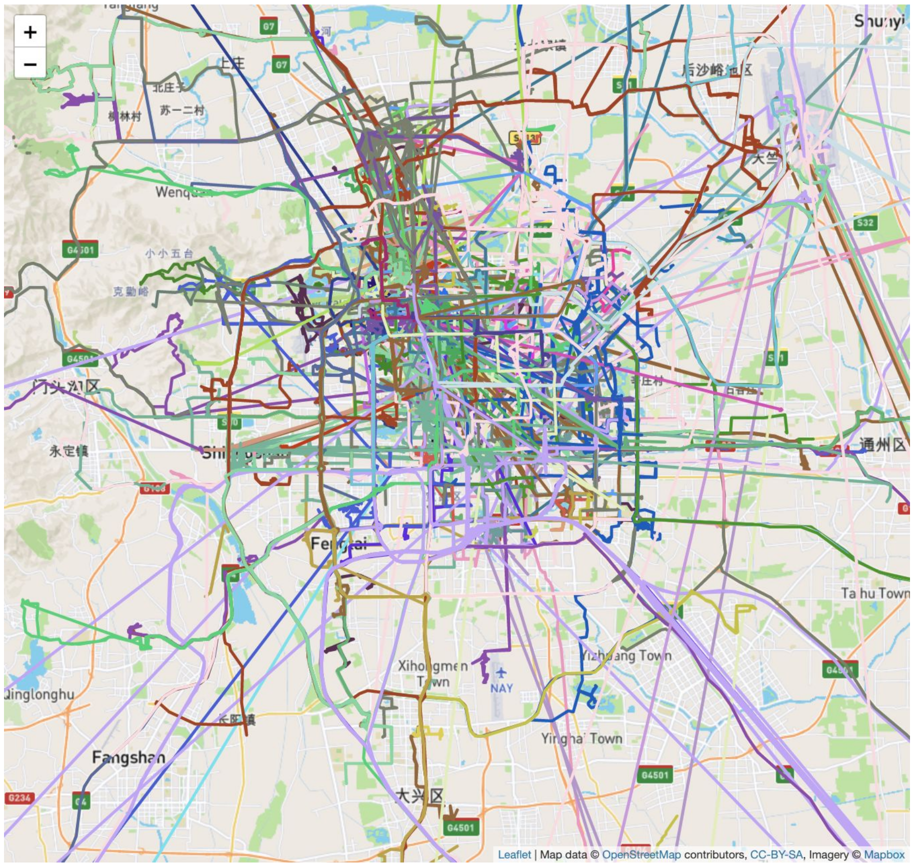

Figure 1: Visualizing Geolife GPS Trajectory Dataset on the Map (about 18,000 trajectories)

2.1. Pre-processing

Different types of pre-processing methods are applied to the dataset to remove the outliers and reduce the dimension of the dataset which are as follows:

1. **Outlier removal:** It is inevitable that there are humans who stay still in some places for a long time. We have removed the trajectories without having a real movement, i.e., if they are the waypoints of the trajectory and, then we will remove these trajectories.
2. **Remove trajectories with many points:** Sometimes, trajectories with too many waypoints can cause computational issues. For instance, we can set an upper bound for

waypoints like 1000. However, we do not think it would be necessary. A lower bound on the number of waypoints can be used for computational issues as well.

3. **Remove users with less than 100 trajectories:** As some users had only a handful of trajectories and some of the had more that 200 trajectories, considering the fact that in deep learning models we need a lot of data to train a network, we decided to remove those users with less than 100 trajectories. Thus, the classification result can be reasonably interpretable. Therefore, we got 41 users to be classified and the number of trajectories in each user differs from 100 to 2124.
4. **Dimension reduction:** In this project, we have used a novel data mapping from our trajectory data to R^{20}, R^{50}, and R^{250} under two different feature mappings [1, 2]. In these papers a novel feature mapping is introduced in such a way that they map a curve into a Euclidean space utilizing some landmarks. The feature mapping used in [1] is not orientation preserving but the feature mapping in [2] captures the orientation of curves. However, there is an extra Gaussian hyperparameter that should be tuned.

Landmarks employed for creating these datasets are stored in the Landmarks folder in CSV format. The first one with size 20 is hand-picked from near the mean and median of waypoints of the trajectories. The second one with size 50 is handpicked from a map visualization of all trajectories in such a way that they are chosen to be near to a group of trajectories. The last one with size 250 is constructed by 200 gridding points plus previously chosen 50 points from the map. To use the feature mapping introduced in [2] we have set the Gaussian hyperparameter σ to different values like 0.25, 0.3, 1, 1000, according to our choice of landmarks, to have more insight in data analysis. Therefore, employing those two feature mappings along with different landmarks and Gaussian hyperparameters, we have transformed Geolife GPS Trajectory Dataset into 6 datasets and we have done the considered classification tasks for all of them separately.

3. Classification

The mapped data is used for classifying the user label. The methods developed in this project are 1-layer Fully-Connected, 2-layer Fully-Connected, 1-layer Convolutional, and 2-layer Convolutional Neural Networks.

Moreover, to better analyse the performance of our model, several machine learning algorithms such as k-nearest neighbors (KNN), Support Vector Machine (SVM) with linear, Gaussian and polynomial kernel, Decision Tree, Random Forest, and AdaBoost are used to compare their results with our models.

Based on the provided guide file, 73 users have labeled their trajectories with transportation mode (car, bus, walk or train) and one of our tasks to do was a classification task based on transportation mode for these users. However, we found those labels not useful as the users have labeled not only some of the trajectories but some parts of those trajectories. Therefore, we are not able to do that sort of classification task. We have tested the performance of the aforementioned models on R^{20}, R^{50}, and R^{250} datasets using both binary and multi-class (10 class) classifications. The following sections show the results, accordingly.

3.1. Neural Networks Architecture

The architecture we have employed for our networks is as follows. After implementing many different architectures, due to the nature of our data, we decided to apply fully connected and convolutional neural networks. For a fully connected network we examined 1 and 2-layer networks with different hidden dimensions and different activation functions. In fact, we found ReLU and LeakyReLU useful for datasets mapped under orientation preserving feature mapping and Tanh for the non-orientation preserving one. For convolutional networks, like fully connected ones, we explored 1 and 2-layer CNN with different number of channels, various activation functions and different kernel sizes depending upon the dimension of mapped data, stride of 1 and padding of 1. Surprisingly enough, we did not find drop out useful at all.

3.1.1. Binary Classification using Fully-Connected Neural Network

The results of the binary classification using 1 and 2-layer fully connected neural networks are reported in Table 1.

Table 1. Train and test errors for various datasets

Feature Mapping	Landmarks Size	Gaussian Hyperparameter	1 Layer Fully Connected		2 Layer Fully Connected	
			Train Error	Test Error	Train Error	Test Error
Orientation Preserving	20	0.3	7%	18%	7%	18%
Non-Orientation Preserving	20	None	4%	**7%**	4%	7%
Orientation Preserving	50	0.25	0.90%	20.0%	0.90%	20.0%
Orientation Preserving	50	1000	15%	28%	15%	28%
Non-Orientation Preserving	50	None	3%	**6%**	3%	6%
Orientation Preserving	250	1	6%	24%	6%	24%
Non-Orientation Preserving	250	None	12%	14%	12%	14%

As it can be seen the best classification is obtained by 20 and 50 landmarks using the non-orientation preserving feature mapping.

3.1.2. Multi-Class Classification using Fully-Connected Neural Network

The results of the multi-class classification using 1 and 2-layer fully connected neural networks are reported in Table 2. Here we have considered 10 classes.

Table 2. Train and test errors for various datasets

Feature Mapping	Landmarks Size	Gaussian Hyperparameter	1 Layer Fully Connected		2 Layer Fully Connected	
			Train Error	Test Error	Train Error	Test Error
Orientation Preserving	20	0.3	23%	40.0%	30.0%	43%
Non-Orientation Preserving	20	None	32%	33%	17%	21%
Orientation Preserving	50	0.25	5%	49%	4%	43%
Orientation Preserving	50	1000	21%	54%	47%	55%
Non-Orientation Preserving	50	None	14%	**19%**	11%	19%
Orientation Preserving	250	1	8%	56%	5%	60.0%
Non-Orientation Preserving	250	None	42%	43%	56%	57%

Like above, the best classification is obtained by 50 landmarks using the non-orientation preserving feature mapping.

3.1.3 Binary Classification using Convolutional Neural Network

The results of the binary classification utilizing 1 and 2-layer 1d-convolutional neural networks are reported in Table 3.

Table 3. Train and test errors for various datasets

Feature Mapping	Landmarks Size	Gaussian Hyperparameter	1 Convolutional Layer		2 Convolutional Layer	
			Train Error	Test Error	Train Error	Test Error
Orientation Preserving	20	0.3	25%	26%	29%	30%
Non-Orientation Preserving	20	None	23%	25%	29%	30%
Orientation Preserving	50	0.25	22%	24%	28%	29%
Orientation Preserving	50	1000	24%	25%	26%	28%
Non-Orientation Preserving	50	None	24%	25%	30%	30%
Orientation Preserving	250	1	25%	26%	29%	30%
Non-Orientation Preserving	250	None	24%	26%	28%	29%

As shown in Tables 3, using 2 convolutional layers adversely affects the accuracy of classification. One possible reason for that could be the limited availability of data for training the model. The more data provided for training the CNN models, the better accuracy we got. Moreover, as shown in Table 3, various dataset produced almost the same level of train and test errors. In other words the accuracy of the classification is not dependent on the dataset.

3.1.4. Multi-class Classification using Convolutional Neural Network

The results of the multi-class (10 classes) classification using 1 and 2-layer convolutional neural networks are provided in Table 4.

Table 4. Train and test errors for various datasets

Feature Mapping	Landmarks Size	Gaussian Hyperparameter	1 Convolutional Layer		2 Convolutional Layer	
			Train Error	Test Error	Train Error	Test Error
Orientation Preserving	20	0.3	42%	48%	41%	46%
Non-Orientation Preserving	20	None	44%	49%	39%	46%
Orientation Preserving	50	0.25	47%	50%	40%	46%
Orientation Preserving	50	1000	44%	49%	40%	46%
Non-Orientation Preserving	50	None	44%	49%	39%	46%
Orientation Preserving	250	1	44%	50%	39%	45%
Non-Orientation Preserving	250	None	44%	49%	41%	46%

As expected, the performance of the developed model has decreased due to involving more number of classes in each classification task. Moreover, as we increase the number of convolutional layers, the train and test errors decrease in all the datasets. However, the computational cost of utilizing two convolutional layers is much more expensive than using one convolutional layer.

3.2. Machine Learning Algorithms

In this section, the results of the binary classification using machine learning models are given in Tables 5 and 6. We have used the sklearn libraries to do these experiments.

Table 5. Train and test errors for various datasets and classifiers

Dataset	Orientation Preserving		Non-Orientation Preserving		Orientation Preserving		Orientation Preserving	
Landmark Size	20		20		50		50	
Gaussian Hyperparameter	0.3		None		0.25		1000	
Classifier Error	Train Error	Test Error	Train Error	Test Error	Train Error	Test Error	Train Error	Test Error
KNN (n_neighbors = 5)	1.3%	1.7%	0.4%	0.6%	1.6%	2.2%	1.9%	2.6%
Linear SVM	2.7%	3%	2.1%	2.2%	1.9%	2.7%	2.5%	3%
Gaussian Kernel SVM	2.5%	2.7%	2.4%	2.5%	0.9%	1.8%	2.5%	2.8%
Polynomial Kernel SVM	2.8%	3.1%	2.8%	2.9%	1.8%	2.82%	2.9%	3.2%
Decision Tree	0.08%	0.12%	0%	0.5%	0%	1%	0%	1.1%
Random Forest (50 Estimators)	0.08%	0.08%	0%	0.4%	0%	0.5%	0%	0.7%
AdaBoost (50 Estimators)	0.2%	0.9%	0%	0.4%	0%	0.6%	0%	0.7%

Table 6. Train and test errors for various datasets and classifiers

Dataset	Non-Orientation Preserving		Orientation Preserving		Non-Orientation Preserving	
Landmark Size	50		250		250	
Gaussian Hyperparameter	None		1		None	
Classifier Error	Train Error	Test Error	Train Error	Test Error	Train Error	Test Error
KNN (n_neighbors = 5)	0.4%	0.6%	1.9%	2.6%	0.4%	0.6%
Linear SVM	2%	2.1%	0.7%	3.3%	1.6%	1.7%
Gaussian Kernel SVM	2.6%	2.6%	1.4%	2.4%	2.3%	2.4%
Polynomial Kernel SVM	2.9%	2.9%	2.4%	3.1%	2.9%	2.9%
Decision Tree	0%	0.5%	0%	1.1%	0%	0.5%
Random Forest (50 Estimators)	0%	0.4%	0%	0.7%	0%	0.3%
AdaBoost (50 Estimators)	0%	0.4%	0%	0.8%	0%	0.3%

As shown in Tables 5 and 6, the machine learning models outperform deep learning models in all datasets.

4. Conclusion

In this project, several deep learning models are developed and the results of the models are compared with machine learning algorithms. Based on the results, deep learning models have poor performance in both time and accuracy of classification. Based on the results in Tables 5 and 6, we reached 100% accuracy in the training dataset and 99.7% in the test dataset using machine learning algorithms. However, the accuracy of deep learning models are much less than these values. One explanation for this could be the limited amount of available data for training the deep learning models. Due to the complexity of deep learning networks, we need a considerable amount of data in order to have a reliable network. As seen, with a limited size dataset, the performance of machine learning models are much better. The other reason possibly is that the loss functions we are optimizing are different. For instance, we have employed cross entropy loss for networks but hinge loss is used in linear SVM. Therefore, optimizing different loss functions gives different results.

For future studies, the authors aim to collect more trajectory data from different sources such as departments of transportations (DOTs) in order to train more reliable deep learning models. Also, combining different deep learning architecture such as CNN, ANN, and RNN in order to take the advantages of all models and cover their drawbacks is another suggestion of the authors. Additionally, using a data augmentation method could help generate more data and develop a more stable model.

5. References

1. Jeff M. Phillips and Pingfan Tang. Simple distances for trajectories via landmarks. In *ACM GIS SIGSPATIAL*, 2019.
2. Jeff M. Phillips, and Hasan Pourmahmood-Aghababa, 2020. Orientation-Preserving Vectorized Distance Between Curves. *arXiv preprint arXiv:2007.15924*.

6. Github Repository For Mapped Datasets

The mapped datasets are stored in our Github repository in the “New-Data” folder at
https://github.com/aghababa/deep-learning-project.
Data is in zipped csv format. We have also stored landmarks, which are used to map the original dataset to Euclidean space, in the “Landmarks” folder in csv format.