PyTorch-based implementation of label-aware graph representation for multi-class trajectory prediction

Men, Qianhui; Shum, Hubert P.H.

Published in:
Software Impacts

Published: 01/02/2022

Document Version:
Final Published version, also known as Publisher's PDF, Publisher’s Final version or Version of Record

License:
CC BY

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1016/j.simpa.2021.100201

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.

Download date: 23/05/2022
PyTorch-based implementation of label-aware graph representation for multi-class trajectory prediction

Qianhui Men a, Hubert P.H. Shum b,*

a City University of Hong Kong, Hong Kong
b Durham University, United Kingdom

A R T I C L E   I N F O

Keywords:
Trajectory prediction
Graph convolution network
Multi-class prediction
Traffic analysis
Human motion understanding

A B S T R A C T

Trajectory Prediction under diverse patterns has attracted increasing attention in multiple real-world applications ranging from urban traffic analysis to human motion understanding, among which graph convolution network (GCN) is frequently adopted with its superior ability in modeling the complex trajectory interactions among multiple humans. In this work, we propose a python package by enhancing GCN with class label information of the trajectory, such that we can explicitly model not only human trajectories but also that of other road users such as vehicles. This is done by integrating a label-embedded graph with the existing graph structure in the standard graph convolution layer. The flexibility and the portability of the package also allow researchers to employ it under more general multi-class sequential prediction tasks.

1. Introduction

Multi-class trajectory prediction is challenging due to the needs to model different types of trajectory with diverse velocities and patterns. The class information of the trajectory that indicates the type of a road user is essential for guiding an accurate estimation of the future moving trend. As an motivating example, the trajectory of a car or that of a bike may have different impacts on the future movement of a pedestrian, as pedestrians tend to keep a larger distance with a car. For such a multi-class situation, we model the trajectory of the road user to be predicted as an undirected spatial graph. Our main insight is to incorporate class labels into the graph representation, which will be harnessed to inform the prediction of the trajectory.

Our idea is developed on a spatial–temporal graph convolutional neural network (STGCNN) with a purposefully designed adjacency matrix to enhance the trajectory prediction with semantic meanings. We introduce the use of class labels in the geometric-based graph structure of the trajectories to be encoded. This way, the predicted trajectory will be informed by not only the observed trajectories of nearby road users, but also their categorical properties. The successful adoptions
to two popular tasks in the trajectory prediction domain, i.e., traffic trajectory prediction [1] and skeleton-based motion prediction [2], have supported the usability and the generality of the proposed method in complicated real-world scenarios.

We implement the solution in PyTorch. We adapt the backbone (STGCNN) framework from Mohamed et al. [3], due to its superior performance and well-organized graphical structure of modeling spatial–temporal trajectory dependencies. We also utilize standard Python libraries for numerical calculations and data visualization. To enhance the effectiveness of the algorithm, we introduce a class-balancing scheme to re-weight the loss of each class with respect to their quantities. The implemented software has formed a part in demonstrating superior prediction accuracy for crowd trajectories [1].

2. Software description

As an advanced deep-learning approach, recent articles usually model the social relationship of the trajectories under a predefined graph structure [3–6], i.e., $G = (V, E)$, where the node set $V$ contains all entities in the scene. $E$ represents the edge connecting any two nodes in the graph $G$. An adjacency matrix $A_1$ that decided by the relative distance of the object trajectories is usually used to represent the correlation attributes of the graph [1,3,7]. Such a matrix indicates if two nodes should be considered as connected, and if so, to what extent.

In this research, we propose a label-guided adjacency matrix $A_l$, which is defined by:

$$A_l = [A, A']W_1 + b_1,$$

where $W_1$ and $b_1$ represent the weight and the bias parameters of a linear layer, respectively. Here, $A$ is a logical square matrix, where $1$ represents the same trajectory label of the row and column indices, and $0$ otherwise. The one-hot embedding before the linear mapping ensures that the same label pairs will have the same features learned. Furthermore, the overall adjacency matrix $\hat{A}$ merges the features from the distance-based adjacency, $A_d$, and the label-based adjacency, $A_l$, with a linear layer, which is given by:

$$\hat{A} = [A_d, A_l]W_2 + b_2,$$

where $W_2$ and $b_2$ are the weight and the bias of a linear layer, respectively.

Compared with a non-parametric merging, the linear layer introduced here has two advantages. It is to better fit the two levels of adjacency correlations and to increase the model capacity. Note that such graphical representations can be deployed in many types of graph-based deep modeling such as the most prevalent graph convolution networks (GCNs) proposed by [8], or graph attention networks (GATs) [9] to learn high-level node embeddings. The architecture of learning a label-aware graph representation and modeling the predicted trajectory is given in Fig. 1.

Furthermore, we use Spatial–Temporal Graph Convolutional Neural Network (STGCNN) following [3] as an architecture backbone to model the spatial and temporal representation under the new label-enhanced graph structure. This is because such a network is effective in modeling trajectory data with both spatial and temporal features. To process the temporal information, we stack multiple convolutional layers to model the in-depth sequential dependencies of the trajectory. During inference, bi-variant distributions of the future trajectories are predicted including mean, standard deviation, and correlation for each object at each timestamp. Positional trajectories can then be sampled from such distributions. An algorithm flow is given in Alg. 1 to show the architecture of the code.

The software implemented based on the above introduced model is programmed under the well-supported PyTorch machine learning framework with common packages including numpy$\geq$1.0 and torch 1.2.0. Two extensive experiments on both 2D traffic (Stanford Drone Dataset [10]) and 3D skeleton (CMU Dataset [11]) trajectory predictions are also executable for training and validation.

To enhance the efficiency of the software, we normalize the input data with a proper scaling factor (10 for 2D traffic and 1000 for 3D skeleton). We also adjust the error loss of different categories of objects based on their occurrences. Since in 3D skeleton prediction, all joints as classes appear equally in the motion, the adopted class-balancing scheme only takes effects on the traffic trajectory prediction.

3. Impact overview

Under the wide topic of trajectory prediction, we introduce the software that can benefit many applications including but not limited to autonomous driving, pedestrian behavior understanding, motion prediction and estimation. The software is constructed on a graph convolution neural network basis, which is generally recognized as one of the most effective models in analyzing the trajectories of moving objects. Since the class label is always captured along with the object information, the software can be easily adaptive to a new dataset or prediction environment under multiple class scenarios. In this paper and the attached software, we show two example frameworks of learning a label-aware spatial–temporal GCN under traffic and skeleton trajectory prediction. The software provides a trajectory prediction implementation on transferable tasks that shows improvements upon the standard spatial–temporal GCN baselines when class labels are not considered. Also note that compared to the label-agnostic baseline, label-aware GCN only shows a slight increase of trainable parameters and the model size, and the inference time is hardly increased. The software also provides standard evaluation metrics with distance-based measurements [7,12] that enables the programmers to validate their prediction results on their own dataset.

This software can be extended for a number of potential applications in the real world. First, automatic object and human tracking in surveillance cameras have become more and more popular in recent years [13]. Apart from only analyzing observed trajectories in the past, the accurate prediction of trajectories in this research enables the prevention of potential accidents by providing warnings in advance [14]. This is particular useful in situations with mixed road users, such as people crossing a roads with running cars. Second, in the field of autonomous vehicles, the use of overhead cameras for driving assistant has shown encouraging results, particularly in challenging scenarios.
such as roundabouts [15]. Our research adds values to autonomous driving systems by predicting the trajectories of nearby road-users, thereby improving the decision making of these systems. Finally, crowd trajectory analysis systems [16] have demonstrated their capacity to inform city planning and design, such as minimizing blockage of people [17]. As our system considers multiple types of road users, it allows predicting possible collisions from different road users as a mean to evaluate collisions risks. Such information is useful for civil engineers to design roads and environments that create a safe environment.

4. Conclusion and future improvements

In this work, we explore the possibility of utilizing label information to enhance multi-class trajectory prediction. This is done by constructing a label-aware graphical representation that is informed by the trajectory semantics. The easy-to-use software is also advantageous to researchers in many different areas. Since in some specific tasks such as traffic prediction, the static objects such as trees and buildings will also influence the predicted trajectory. Considering these high-level label semantics would be an interesting future direction to explore.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Ben A. Rainbow for his contributions to the preliminary version of this code. This work was supported in part by the Royal Society, United Kingdom (Ref: IES\textbackslash R2\textbackslash 181024).

References