Deep Neural Network For Trajectory-based Driver Identification

Jian Li*, Weishan Dong†
*Software Institute, Nanjing University, Nanjing, China
†IBM Research - China, Beijing, China
lj12@software.nju.edu.cn, dongweis@cn.ibm.com

***Abstract*—In this paper, we proposed to apply deep neural network on trajectory to model the driving behavior and evaluate it through the driver identification task. The main idea of our paper is to organize the original trajectory data into double windowed feature matrix and then apply Convolutional Neural Network and Recurrent Neural Network on it. The network architecture are designed intuitively and empirically. The experimental results indicate that deep neural network such as CNN and RNN are practical at modeling behaviors based on trajectory data.**

***Index Terms*—trajectory classification; driving behavior modeling; driver identification**

I. Introduction

Nowadays, driving has become a most essential part of our daily life. However, safety, as one of the problem it came with, is becoming more severe. Addressing this problem is not only crucial for every driver, but also for the city and country. Thanks to the development of technology, the transportation tends to be smarter and more orderly. In this progress, driving behavior modeling is one of the popular research fields since behavior can tell us about the status of the driver and help to predict the future, which may contributes to build a more intelligent driving assistant system. In this paper, we focus on the trajectory data derived from car Global Position System(GPS) and try to mine the driving patterns about behaviors and habits of drivers from this.

Instead of using some traditional mining method of trajectory, we proposed to apply the hot deep neural network to the trajectory to let the model learn more from the trajectory itself. Generally speaking, a deep neural network refers to a feedforward neural network with more than one hidden layer, contrasted with shallow learning algorithms by the number of parameterized transformations a signal encounters as it propagates. There are two popular deep neural networks in past years: Convolutional Neural Network(CNN) and Recurrent Neural Network(RNN). For CNN, the higher layer takes the all the outputs from lower layer as input. It uses tied weights and pooling layers and shows superior results in both image and speech applications. As for RNN, it is a kind of neural network that allows connections between the neurons even in the same layer. RNN is able to process arbitrary sequences of inputs such as handwriting and speech. Intuitively, CNN is deep in organization of the data and RNN is deep in the time series of the data. In short, Both of them are able to detect much deeper and more features from raw input.

As mentioned, deep neural network is really a popular and powerful method especially in computer vision and natural language processing. Specifically, Take speech recognition for example, both convolutional neural network [1] and recurrent neural network [2], [3] were adopted to this task which eventually got convincing result. And meanwhile, when we connect the speech to trajectory, we can find both of them are sequences of information, which means we can get inspiration from the methods used for speech recognition such as deep learning. Hence, In this paper, we will apply deep neural networks to trajectory for building the model of driving behavior and evaluate it on the task of driver identification.

II. Proposed Method

A. Organization of the Input Data

From the very beginning, each input trajectory data is composed of sequences of tuples (x, y, t) in various length's trips, in order to get our data organized and make them suitable for neural network training, we reshape the raw data and generate feature map for each trip.

Inspired from the idea of N-Gram probabilistic model as illustrated in [8], where each word depends only on the last $n-1$ words, we can also utilize the context window to make our vision focused on fixed length's trajectory in periods of time. During each window of trajectory, we may discover potential patterns through observing the behaviors of the driver over different situations. For examples, some drivers may go through the corner quickly while others would like to slow down, and heavy acceleration often happened to a group of people while most of others never have such kind of dangerous driving behavior. And in fact, the behaviors are interdependent in periods of time where the length of this period can be L_s, we can even roughly conclude that each driving behavior depends on what happened in last $L_s - 1$ time points. Because of this, there will be more possibility to discover driving patterns when we focused on the trajectory in fixed length, which means, let machine 'understand' or 'define' this period of behaviors and label them. To avoid missing too much information, the original trajectory will be segmented with shift so that there will be overlap between neighboring segments.

Comparing to the frequency feature map of speech used for neural network as in [1], [2], the next step after we getting segments from the original trajectory is to generate feature maps corresponding to them just like the spectrogram

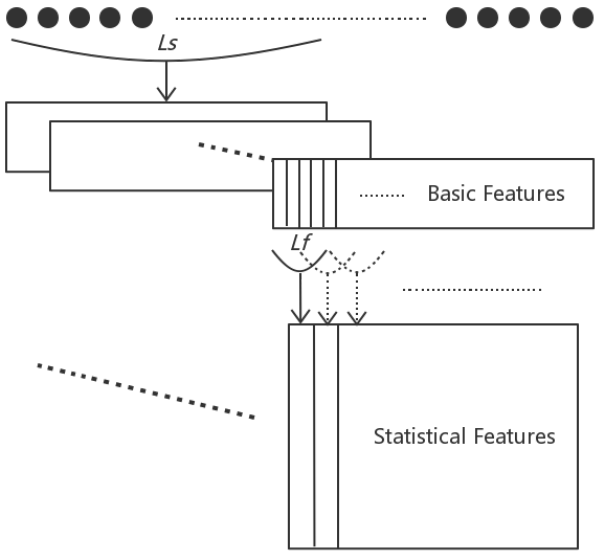

Fig. 1. Flowchart of organizing the input data. 1. Generate segments at the length of L_s and calculate five basic instant features to each segment from raw trajectory sequence of tuples (x, y, t). 2. For each segment, assign L_f neighboring points into frames and produce the statistical feature matrix.

of speech which has both axes of frequency and time. Unlike speech, trajectory doesn't have well-developed method to get feature map directly, we have to produce features manually to replace the frequency axis of speech's feature map. There are five intuitive features: speed, norm of speed, acceleration/deceleration, norm of acceleration/deceleration, angular speed, which here are called basic features. These features can be calculated from the raw data at each time point, which means, every segment will have a basic feature matrix sized $5*L_s$. However, these are all instant features and the result will be more affected by each point especially anomalous outlier. Therefore, we decide to calculate statistical information by 'framing' the segments. In each segment, we put every L_f neighboring points into a frame, then calculate the mean, minimum, maximum, standard deviation and quartile of their basic features in each frame. In fact, each frame's statistical feature vector can be regarded as a more stable representative of features in every period time of L_f. Finally, the statistical feature matrix is exactly the well-organized input data for further training.

So far, in fact, we use a large window to segment the original trajectory into fixed length to observe the interdependency of each kind of behavior, and meanwhile, we employ a small window to get each segment's frames so as to robustly catch the features of specific driving action in a short time period. The double windowed feature matrix not only has instant behavior patterns but also contains the patterns of changing behaviors. Figure 1 shows how we organize the input data.

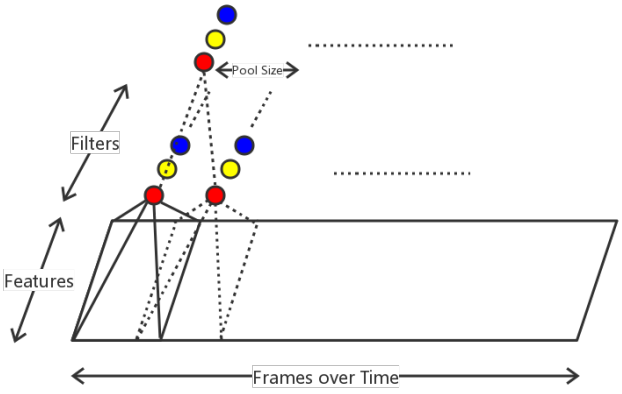

Fig. 2. Convolutional Neural Network for the input feature matrix

B. Convolutional Neural Network based Training

Convolutional Neural Network (CNN), typically LeNet[9] and AlexNet[10], is a kind of biologically-inspired neural network, which consists of alternating so-called convolution and pooling layers. it is always used for processing images by running small receptive windows over inputs to learn the weights of network. And it is known of two main characteristics: locality and weight sharing, both of which are also beneficial to learn the features from time-series data such as speech and trajectory as well as images.

In training step, we apply convolution to trajectory only over time domain because the convolution over frequency has no practical significance. This kind of convolution is useful to detect features like driving behavior or driver's pattern from trajectory. Intuitively, the lower layer is able to detect the action during driving such as acceleration or deceleration while the higher layer has the ability to find driving patterns or driving habits in certain trajectory. There is a similar work [4] at speech field, where the time-axis convolution helps to learn more effective acoustic features such as phoneme and gender. And because of the locality of the convolution network, our model is able to compute features robustly against sporadic outliers with a smaller number of weights. Weight sharing, as another key property, can also reduce the number of weights and improve model's robustness as each weight is learned from multiple time frames.

Besides of convolution layer, pooling layer is another significant component of Convolutional Neural Network. The most typical one is Max-Pooling, which is a form of non-linear down-sampling. In pooling, the same feature values computed at different locations are pooled together and represented by the maximum value. These pooling units is translation invariant, which means that the same feature such as certain kind of behavior will be active even when the pattern undergoes small translations.

As in Figure 2, we build an intuitive as well as empirical Convolutional Neural Network to train a model for driver identification. The overall net contains six layers with weights,

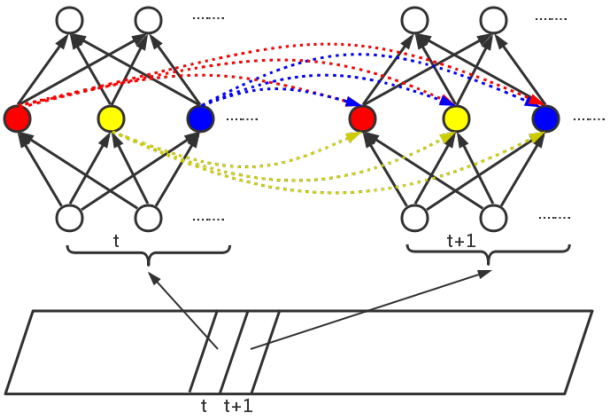

Fig. 3. Recurrent Neural Network for the input feature matrix

the first three are convolution-pooling layers and the remaining three are fully-connected. Specifically, Assuming the number of frames in each segment is 128, then the first layer filters the 35×128 input data with 32 kernels of 35×5 with a stride of 1 frame. The second convolutional layer takes as input the pooled output of the first convolutional layer where pool size is 1×2 and filters it with 64 kernels of size 1×3. The third convolutional layer also has 64 kernels of size 1×3 connected to the pooled outputs of the second convolutional layer. Then the fourth and fifth layer are fully connected and have 128 neurons each and the last layer is a softmax layer. Meanwhile, the sigmoid is applied to the output of every convolutional and fully-connected layer.

C. Recurrent Neural Network based Training

Recurrent Neural Network (RNN) is another typical deep neural network that has many variations such as Elman's RNN[11],LSTM[12] and Bi-directional RNN[13]. As described in [5], it is a kind of feedforward neural network augmented by the inclusion of edges that span adjacent time steps, introducing a notion of time to the model. Given the input sequence $x = (x_1, ...x_T)$, each neuron in recurrent hidden layer receives input from the current data point x_t and also from hidden node values h_{t-1} in previous time step:

$$h_t = g(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

Where W_{hx} is the input-hidden weight matrix and W_{hh} is the matrix of weights between the hidden layer and itself at adjacent time steps. The b_h denote bias vectors and g is the function of hidden layer.

This network can be interpreted as an unfolded network across time steps which is inherently deep in time. Thanks to this property, RNNs are very successfully employed to sequence learning tasks such as speech recognition and machine translation. Obviously, the trajectory is also a kind of sequence and it is a natural way for us to apply RNN to learn some patterns from trajectory data.

However, as illustrated in [6], training recurrent networks is difficult due to vanishing and exploding gradients. Group of researchers have been working hard to develop optimization techniques and network architectures to solve this problem. Despite of using some sophisticated method, Quoc et al.[7] proposed a recurrent neural network architecture composed of rectified linear units and used the identity matrix or its scaled version to initialize the recurrent weight matrix. Their simple solution is even comparable to a standard implementation of LSTM on certain tasks such as language modeling and speech recognition. For our trajectory classification task, we tend to use such kind of simple but powerful method to train a model shown as Figure 3 which here we called IRNN.

III. Experiments

A. Experiment Method

In experiments, we use the data of the kaggle competition 'Driver Telematics Analysis' which is available online[1]. The data contains a number of drivers and each driver has 200 driving trips that record the car's position (in meters) every second. The trips were centered to start at the origin (0,0) then randomly rotated in order to protect the driver's privacy.

During the experiments, we pick 50 drivers' trips and select 80% trips as train and 20% as test data. Since we segment each trip into fixed length segments, training is performed on segments instead of raw trips. Besides the advantages of segment-level training listed in previous section, the training data set which includes more than 35000 segment samples is greatly augmented compared to 8000 trip samples. As for evaluation, we care not only the segment-level but also the trip-level prediction result. Once the prediction of each segment in a trip is obtained, the trip-level prediction will be calculated through adding up every segment's result. As it is a typical classification problem, we take the precision rate as the evaluation criteria. In practice, we tend to use Top-N precision rate to compare different methods, because many of us have similar driving habits. A group of drivers tend to be aggressive on the road while another group of drivers never have unsafe driving events, such as overly sharp turns, heavy acceleration and hard braking. It is feasible to determine the trip belongs to which kind of driver but is difficult to assign a trip exactly to certain driver merely based on trajectory. Hence, the Top-N precision rate is actually an appropriate evaluation indicator.

B. Result Discussion

To begin with, we trained CNN and IRNN respectively with the same input data as stated in section 2. Then, we tried nopoolCNN to show that pooling is as important as convolution and MutiLayerIRNN to figure out how complexity of the model will affect the model. Besides, we tried to use the features generated at the third convolutional layer in CNN as the pre-trained input data of IRNN but the result did not outperform the single layer IRNN. To get more convincing comparison, we take Gradient Boosting Decision Tree (GBDT)

[1]https://www.kaggle.com/c/axa-driver-telematics-analysis/data

as the baseline model. Different from CNN and RNN, GBDT doesn't contain the concept of locality or time in its model. If we use previous fairly raw features as input, GBDT will definitely get stuck as the result shows in the table below, because sequence of feature vectors are folded together and there are lots of same kind features over different time frames leading to confusion. To overcome this, we train a model directly on trip-level and manually generate altogether 57 higher level of statistical features such as curvature or the distance of the whole trip as input, which turns out to be manualGBDT. After a series of parameter adjustment, the performance of each kind of model is listed in the table.

Method	Seg-Top1(%)	Trip-Top1(%)	Trip-Top5(%)
GBDT	18.3	29.1	55.9
manualGBDT	-	51.2	74.3
nopoolCNN	16.9	28.3	56.7
CNN	21.6	34.9	63.7
PretrainIRNN	28.2	44.6	70.4
SingLayerIRNN	34.7	49.7	76.9
MutiLayerIRNN	34.8	52.3	77.4

TABLE I
EVALUATION RESULT.

As the result shows, Recurrent Neural Network is really powerful in modeling the sequence data and even one hidden layer with 100 neurons which noted as SingLayerIRNN is able to beat six layers Convolutional Neural Network and GBDT. But it worth a mention that RNN's training time is much longer than other methods since it needs 5000 thousands iterations while GBDT and CNN only needs around 200 iterations. If considering the model complexity and training time, we find that complexity doesn't always bring significant improvement. For instance, the MutiLayerIRNN which has 2 stacked recurrent layers with 200 hidden units only has minor progress comparing to SingLayerIRNN, but the cost of time doubled and the phenomenon of overfitting is tend to appear. If we get trajectory data that have high quality and quantity, the more complex deep model may have more extraordinary effect than conventional methods.

IV. CONCLUSION

Inspired from deep learning and its application on sequence data such as speech, we used deep neural network to model the trajectory data and tested it on the driver identification task. The result in experiments shows that deep model does have its superiority and it is effective to be applied on trajectory data. But there is still a long way to find more suitable deep architecture of network specifically for trajectory. Once developed, the trajectory based tasks in urban computing field will benefit a lot from this.

ACKNOWLEDGMENT

The authors would like to thank all the team members in IBM Research - China.

REFERENCES

[1] Abdel-Hamid O, Mohamed A R, Jiang H, et al. "Convolutional Neural Networks for Speech Recognition," *IEEE Transactions on Audio Speech & Language Processing*, 2014, 22(10):1533-1545.
[2] Hannun A, Case C, Casper J, et al. "DeepSpeech: Scaling up end-to-end speech recognition," *Eprint Arxiv*, 2014.
[3] Mohamed A R, Graves A, Hinton G. "Speech Recognition with Deep Recurrent Neural Networks," *Acoustics, Speech and Signal Processing, IEEE International Conference on*, 2013:6645 - 6649.
[4] Lee H, Pham P T, Yan L, et al. "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in Neural Information Processing Systems*, 2009:1096-1104.
[5] Zachary C. Lipton, John Berkowitz, Charles Elkan "A Critical Review of Recurrent Neural Networks for Sequence Learning," *Eprint Arxiv*, 2015.
[6] Pascanu R, Mikolov T, Bengio Y. "On the difficulty of training Recurrent Neural Networks," *Eprint Arxiv*, 2012:1310-1318.
[7] Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units," *Eprint Arxiv*, 2015.
[8] Della P J, V J, Lai J C "Class-based n-gram models of natural language ," *Computational Linguistics*, 1992, 18(4):18–4.
[9] Lecun Y, Bottou L, Bengio Y, et al. "Gradient-based learning applied to document recognition ," *Proceedings of the IEEE*, 1998, 86(11):2278-2324.
[10] Krizhevsky A, Sutskever I, Hinton G E. "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, 2012:2012.
[11] Jeffrey L. "Finding Structure in Time," *Cognitive Science*, 1990, 14(90):179-211.
[12] Hochreiter S, Schmidhuber J. "Long Short-Term Memory," *Neural Computation*, 1997, 9(8):1735-1780.
[13] Schuster M, Paliwal K K. "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, 1997, 45(11):2673-2681.