PREDICTION OF POTENTIAL HUMAN INTENTION USING SUPERVISED COMPETITIVE LEARNING

Masayoshi Ishikawa, Mariko Okude, Takehisa Nishida & Kazuo Muto
Hitachi, Ltd
Omika 7-1-1, Hitachi, Ibaraki, JAPAN
{masayoshi.ishikawa.gv, mariko.okude.uh}@hitachi.com
{takehisa.nishida.cu, kazuo.muto.ny}@hitachi.com

ABSTRACT

We propose a learning method to quantify human intention. Generally, a human being will imagine several potential actions for a given scene, but only one of these actions will subsequently be taken. This makes it difficult to quantify human intentions.

To solve this problem, we apply competitive learning to human behavior prediction as supervised learning. In our approach, competitive learning generates several outputs that are then associated with several potential situations imagined by a human. We applied the proposed method to human driving behavior and extracted three potential driving patterns. Results showed a squared error is reduced to 1/25 that of a conventional method . We also found that competitive learning can distinguish valid data from disturbance data in order to train a model.

1 INTRODUCTION

Progress in advanced driving assistance systems (ADASs) has led to an autonomous driving function applicable for highway driving . The vehicle control logic of ADAS is typically developed by hand. There are a few related works on controlling vehicles comfortably , but the complex nature of vehicle environments prevents development by hand. Therefore, recent research has focused on the application of machine learning to autonomous driving systems. The convolutional neural network (CNN) shows particular promise as a core algorithm (Krizhevsky et al. (2012)). C. Chen & Xiao (2015) estimates affordance for driving directly from a front camera image. They train a CNN to generate key perception indicators from images to easily control a vehicle. Similarly, Bojarski et al. (2016) predict the desired steering command directly from front camera images. They train a CNN with a human command as training data and drive in traffic on local roads with or without lane marking and on highways.

Our objective is to provide drivers with a comfortable trip featuring automatic vehicle control . The above methods provide a uniform control for various scenarios, but drivers have an assortment of preference behaviors, and individuals may drive in different ways even if the scenario or situation is the same. For example, on the highway, one person might drive hard to arrive at a destination quickly while another might drive relatively slowly to arrive at a destination safely. Therefore, we want to provide autonomous driving adapted to each individual to improve ease of driving and to generate control targets that imitate individual behavior. To imitate individual behavior, we predict future vehicle states resulting from an individualfs decision.

There are many studies on the imitation of human behavior within the context of driving. Ma & Andréasson (2006) proposed a vehicle interaction model that predicts future acceleration on the basis of current acceleration, velocity, relative velocity, and relative distance of the preceding vehicle. Moon & Choi (2011) proposed a method to predict human steering behavior. Their model considers path planning, feed-forward steering, and feedback steering. While these methods consider only acceleration or steering, Gindele et al. (2010) proposed a more complex model, the dynamic Bayesian network model, to estimate driver behavior and vehicle trajectory. This model considers vehicle model, trajectory, driver decision, and situation context. Wada et al. (2007) pro-

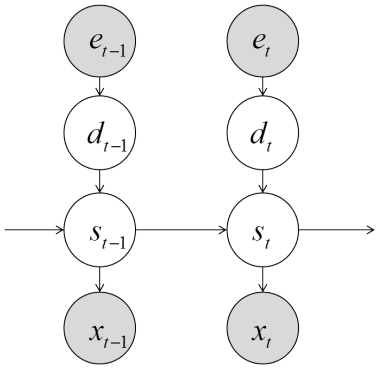

Figure 1: Graphical model of driving behavior.

posed a method to predict braking behavior. This method monitors the size of a preceding vehicle by means of front camera images to detect collision risk.

These methods provide one output for one scene. However, we assume that one output might represent just one part of a driverfs behavior. We assume a driver will imagine several potential situations for one scene and have several potential actions to take associated with each situation. This means the driver is making decisions all the time. Therefore, we want to extract several of the driver's candidate actions or intentions for the autonomous control of a vehicle while considering various potential situations.

In this paper, we propose a method to extract a driver's potential intention by utilizing competitive learning (Ahalt et al. (1990), Shinozaki & Naruse (2013), Osoba & Kosko (2013)). Originally, competitive learning was used for unsupervised neural network algorithms and functioned as clustering. In such competitive learning, neural networks have several output layers and only one path that outputs the smallest loss to be trained . Consequently, neural networks predict the cluster with the most activated units .

Ahalt et al. (1990) compare several competitive learning algorithms. In this paper, competitive learning algorithms is dealt as a kind of vector quantization algorithms. Shinozaki & Naruse (2013) utilize competitive learning for pre-training of deep neural networks and they also use competitive learning as unsupervised learning. Osoba & Kosko (2013) consider supervised competitive learning. However, they use also competitive learning as a kind of clustering methods. Therefore, in our best knowledge, competitive learning is dealt as a kind of unsupervised learning algorithms or clustering algorithms.

We apply competitive learning to time series supervised learning, especially, regression task. And we train neural networks to output several patterns associated with a driver's potential intentions. In this paper, we describe how to model the driving behavior with competitive learning architecture . Additionally, we show the experimental results of tests performed with an actual vehicle.

2 Driver Behavior Model

Here, we describe how to model driving behavior and how to implement the neural network architecture.

Our driving behavior model is a dynamic system that includes both a driver and a vehicle system. In this model, the driver observes the environment and then decides on a driving action, e.g., steering or pedal operation. The vehicle system then changes its state depending on the driving action and the state of the vehicle at a previous time. Finally, we observe the various vehicle states. For example, the environment observed by the driver is the same as that shown by the front camera and the vehicle states include travel speed, accelerations, gas pedal position, brake pedal position, engine speed, engine load, engine temperature, fuel level, fuel temperature, cooling water temperature, etc.

A graphical model of driving behavior is shown in Fig. 1. Here, e stands for the environment observed by the driver, d stands for the driving action decided by the driver, s stands for vehicle states,

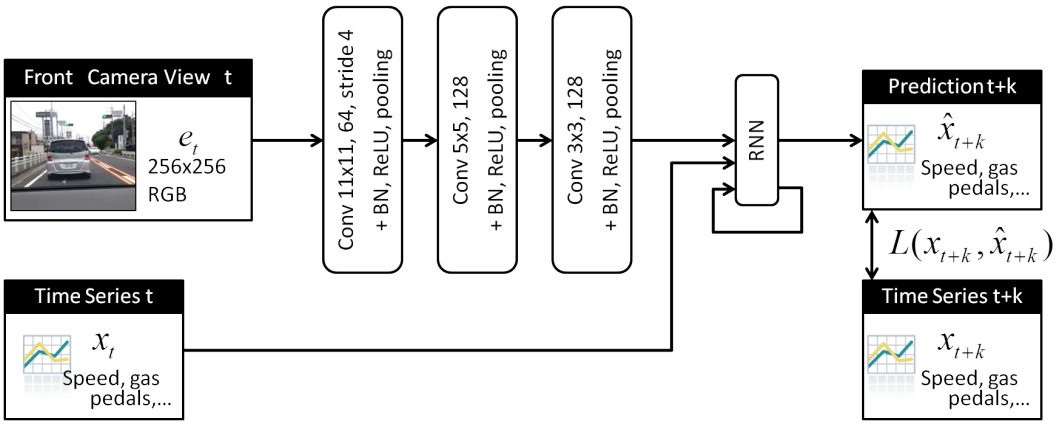

Figure 2: Neural network architecture designed to imitate driving model.

and x stands for observed vehicle states. Subscripts $t-1$ and t refer to time steps. Environment e and parts of vehicle states x are observed. Driving action d and whole vehicle states s are latent variables. We assume a front camera view for environment e. Observed variables x include travel speed, gas pedal position, engine speed, and engine load.

To ensure comfortable driving, we estimate the driver's intention by predicting vehicle states in k step future . The relations between each variable are shown in the following equations.

$$
\begin{aligned}
d_t &= f_e(e_t; \theta_e) && (1)\\
s_t &= f_s(s_{t-1}; \theta_s) + f_d(, d_t; \theta_d) && (2)\\
x_t &= f_x(s_t; \theta_x) && (3)\\
x_{t+k} &= f_{x+k}(s_t; \theta_{x+k}) && (4)
\end{aligned}
$$

Additionally, we compensate for latent variable s_t by observable variable x_t. The compensated s_t are shown in Eq. 5:

$$
s_t = f_s(s_{t-1}; \theta_s) + f_d(, d_t; \theta_d) + f_x^{-1}(x_t; \theta_x^{-1} \quad (5)
$$

We design the neural network architecture in accordance with the driving behavior model shown in Fig. 1. The designed architecture is shown in Fig. 2. First, we select convolutional neural network (CNN) layers to approximate driving action f_e, as the driver decides which action to take depending on the front view and CNN is the best layer to process images. The front camera images are resized to 256×256 with RGB channels. Second, we select a recurrent neural network (RNN) layer to approximate vehicle states f_s, as our driver model is a dynamic system (Graves (2013)). Finally, we select a fully connected layer for other layers f_d, f_x^{-1}, f_{x+k} because of its usability. This model predicts observed variables and we train the driver behavior model by loss function. We select squared error as loss function $L(x_{t+k}, \hat{x}_{t+k})$ and update parameters by backpropagation (Hecht-NielsenWerbos (1990)).

$$
L(x_{t+k}, \hat{x}_{t+k}) = |x_{t+k} - \hat{x}_{t+k}|^2 \quad (6)
$$

The CNN consists of three layers. In the first layer, kernel size is 11×11 with 64 channels and stride 4 . In the second layer, kernel size is 5×5 with 128 channels. In the third layer, kernel size is 3×3 with 128 channels. After all CNN layers, we apply batch normalization, ReLU, and max pooling (Ioffe & Szegedy (2015)). The RNN layer has 1024 units. We use the ADAM algorithm for optimization (Kingma & Ba (2014)).

The architecture shown in Fig. 2 is a baseline architecture and cannot extract potential driver intentions. In the next section, we introduce a competitive learning architecture that can extract potential intentions and discuss how to train it.

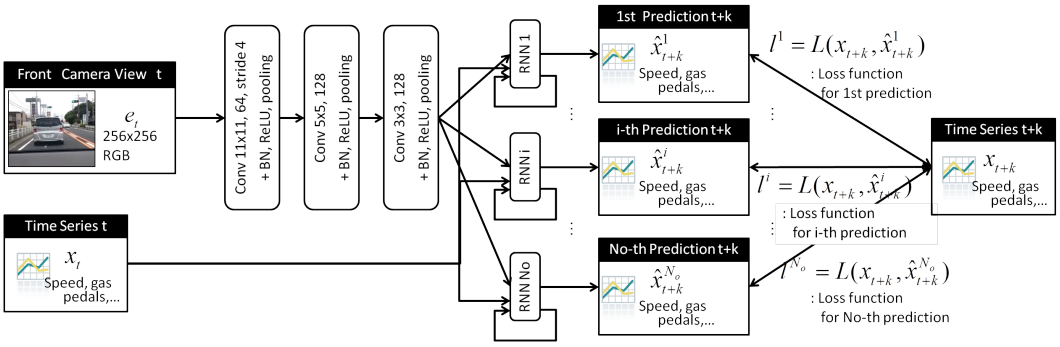

Figure 3: Competitive learning architecture.

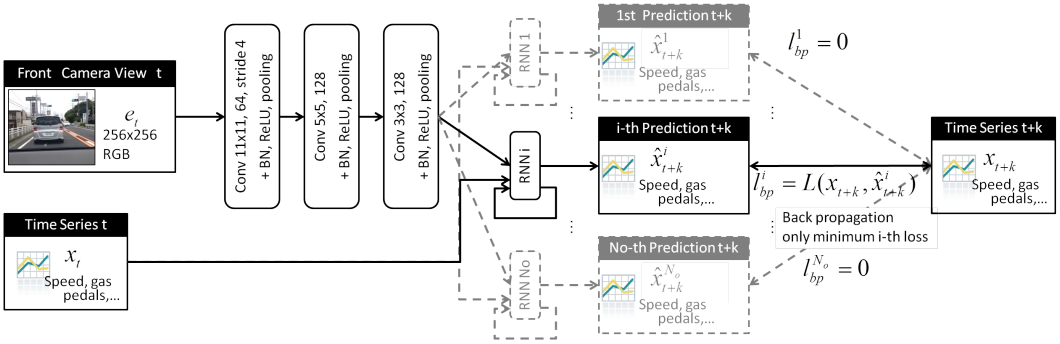

Figure 4: Backpropagation in competitive learning architecture.

3 Competitive Learning for Driver Behavior Model

3.1 Competitive learning architecture

The competitive learning architecture is shown in Fig. 3. It has N_o output layers. Additionally, we set up N_o RNN layers, as the driver's potential intentions are separated depending on the driving actions . Therefore, a separated RNN will be affected by separated intentions. The i-th RNN i and the output layer generate i-th prediction $\hat{x}^i_{t+k}$ at time t+k. We calculate i-th loss using Eq. 7 and decide the loss for backpropagation using Eq. 9. Finally, we update the parameters depending on this loss for backpropagation.

$$l^i = L(x_{t+k}, \hat{x}^i_{t+k}) \tag{7}$$

$$L = [l^1, ...l^i, ...l^{N_o}] \tag{8}$$

$$l^i_{bp} = \begin{cases} 0 i \neq \arg\min_j(L) \\ l^i i = \arg\min_j(L) \end{cases} \tag{9}$$

Backpropagation in the competitive learning architecture is shown in Fig. 4. In the competitive learning architecture, we train only the computation path that output minimum loss. In Fig. 4, the computation path to be trained is indicated by solid lines and the layer whose parameters are not updated is indicated by dashed lines. In this case, the i-th output layer has minimum loss and is trained. The other output layers (i.e., whose losses are bigger than i-th) are not trained and the losses are compensated as zero. Since each output layer is trained by different data, each prediction reflects different intentions. Therefore, competitive learning can extract several potential intentions.

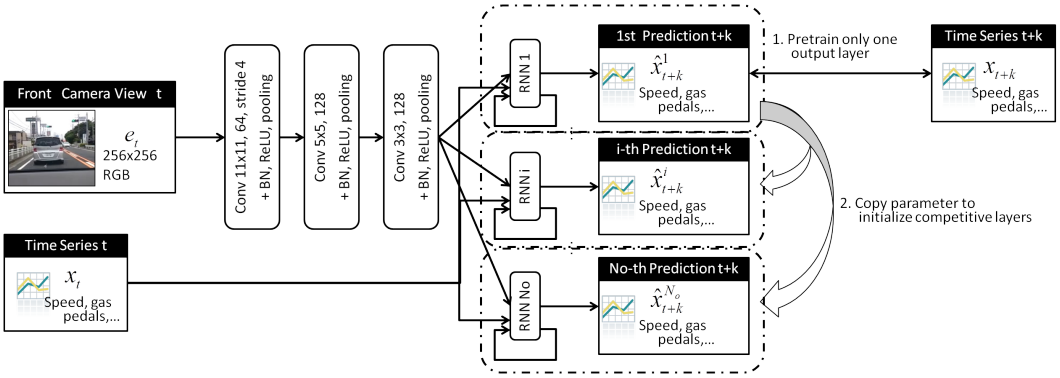

Figure 5: Pre-training for competitive learning architecture.

3.2 Pre-training for Competitive Learning

Here, we discuss the pre-training for competitive learning. In our experiments, competitive layers with simple initialization are difficult to train, and many experiments result in only one output layer being trained. Since only one computation path is trained at a time in competitive learning, how the trained path is decided depends on the initialization. Therefore, we need to ensure appropriate initialization for competitive layers. To this end , we adopt a pre-training technique for competitive learning (Erhan et al. (2010), Erhan et al.). Pre-training is an initialization technique for neural networks and is typically used for stacking neural networks. In this work, we apply pre-training to initialize the parallel competitive layer, as shown in Fig. 5. First, we train only one output layer by means of the ordinal loss function in Eq. 6. Then, we initialize competitive layers by copying pre-trained parameters. Finally, we train each output layer as fine-tuning.

4 Experimental Results

In order to evaluate the proposed method, we record the front camera view using a smartphone and collect actual vehicle data using OBD2 (Birnbaum & Truglia (2000), International (2003)). The data on vehicle states are whitened by removing means and scaled variances. Front camera images and vehicle data are resampled to 2 Hz. Six steps (equivalent to three seconds in the future) are predicted. We collect 80 minutes of data and use 40 minutes to train the neural networks. We set three competitive layers and train 300 iterations for pre-training and 2000 iterations for fine-tuning. We train neural networks with TITAN Z and use the Chainer framework (Tokui et al. (2015)).

4.1 Comparison of Competitive Learning Architecture with Baseline Architecture

Before we show the competitive learning results, we present the results of the baseline architecture shown in Fig. 6 . These data are predictions of vehicle speed over ten minutes. The left figure is the prediction on training data and the right one is on test data. Red line indicates the measured speed and blue line indicates the prediction by baseline architecture. In the training data, the baseline architecture could predict accurately in the low speed band under 30 km/h. However, the prediction error became bigger in the middle-high speed band over 30 km/h. In the higher speed band, the driverfs action had several variations even when the vehicle was operating in the same environment. This variation made prediction difficult. In the test data prediction, prediction error became bigger in not only the high speed band but also the very low speed band around 0 km/h . Additionally, the timing of the acceleration or deceleration shifted later, too. This is because the baseline architecture cannot extract potential driving intentions.

The competitive learning results are shown in Fig. 7. This figure is written up the same way as Fig. 6, except the blue line refers to integrated predictions by the competitive learning architecture. These predictions are a combination of three outputs derived by selecting minimum loss. In the training data prediction, the competitive learning architecture predicted a value quite close to the

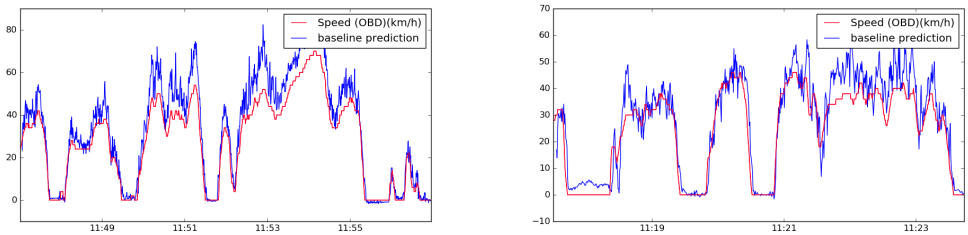

Figure 6: Prediction of baseline architecture on training data (left) and test data (right).

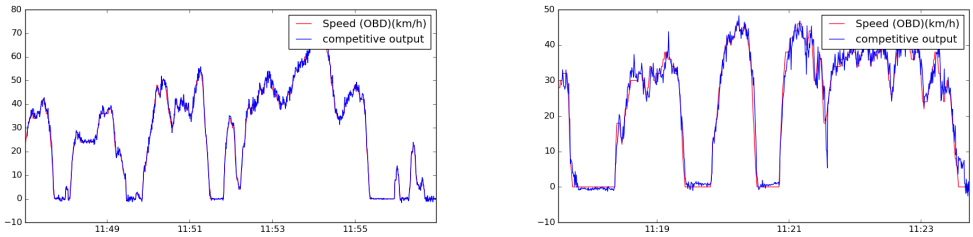

Figure 7: Integrated prediction of competitive architecture on training data (left) and test data (right).

measured vehicle speed before three seconds . In the test data prediction, the competitive learning architecture could also predict the speed accurately. Additionally, the timing shift of acceleration or deceleration became smaller. The loss summation in both the training and test data is listed in Table 1. Competitive learning architecture loss was about 1/25 smaller in the training data and about 1/3 smaller in the test data compared with the baseline architecture.

4.2 Competitive Learning Prediction Associated with Potential Intentions

Here, we show the driving intentions extracted using the competitive learning architecture. The intentions extracted from the training data are shown in Fig. 8. Four time series data are included: upper left is the integrated prediction, upper right is the first output layer's prediction (indicated by green line), lower left is the second output prediction (blue line), and lower right is the third output prediction (purple line). The red lines are the measured vehicle speed. The upper left figure depicts the winning output prediction, where the minimum error prediction is output for each time and the line color refers to the winning output layer . For example, the purple line in the upper left figure is the prediction by the third output layer . We can see that each output layer is affected by other driving intentions. The first output layer accurately predicted the stop timing associated with the stop intention, the second output layer accurately predicted the acceleration or high speed band associated with a rapid intention, and the third output layer accurately predicted the deceleration timing associated with a careful intention. We also show the results of the test data in Fig. 9, which are written the same as the training data in Fig. 8. Trained intentions are also valid in test data because the first output was accurate at stop time, the second output was accurate at acceleration or the high speed band, and the third output was accurate at the time of deceleration. Therefore, extracting these potential intentions enables accurate prediction.

Table 1: Summation loss of each architecture

Architecture	Loss (training)	Loss (test)
Baseline	335.8	2603
Competitive learning	13.14	816.4

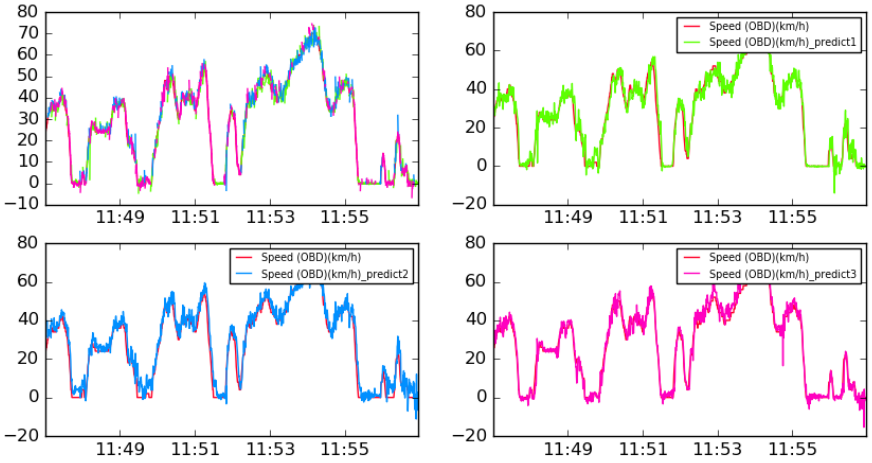

Figure 8: Driving intentions extracted by competitive learning architecture; training data.

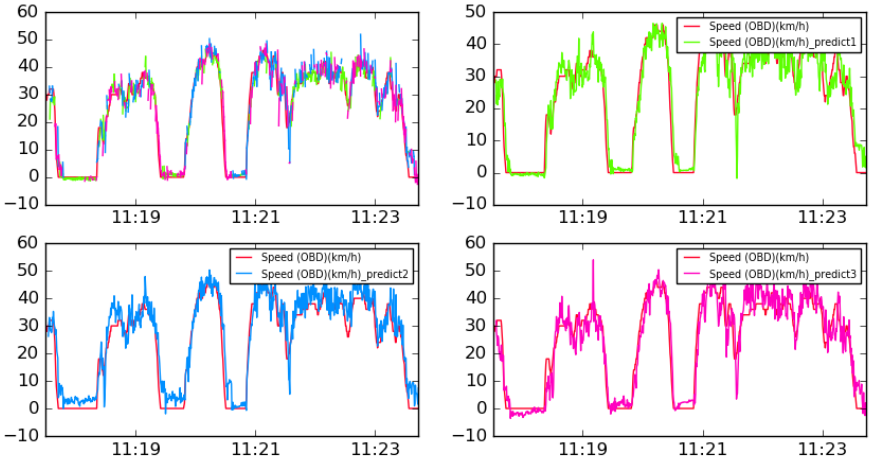

Figure 9: Driving intentions extracted by competitive learning architecture; test data.

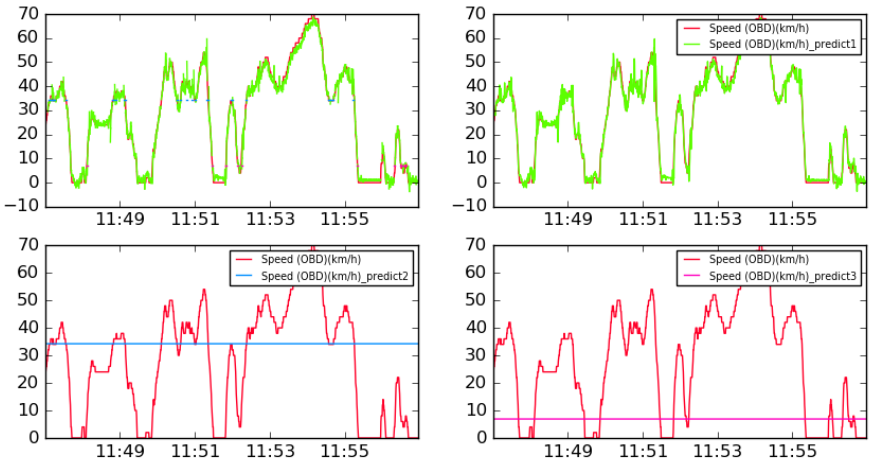

Figure 10: Competitive layer outputs without pre-training in training data.

Table 2: Summation loss without pre-training

Architecture	Loss (training)	Loss (test)
Baseline	335.8	2603
Competitive learning	13.14	816.4
Competitive learning without pre-training	164.0	870.9

4.3 Effect of Pre-training

The competitive learning result without pre-training is shown in Fig. 10. We can train ”only one” output layer to predict driver intention even if we have three output layers. Potential intentions of the driver have similar trends. Therefore, we require the initialization of several of the same output layers . We can extract potential intentions by training each output layer using other data.

We show the summation loss in Table 2. Surprisingly, competitive learning without pre-training significantly improved its losses by about 1/2 for training data and about 1/3 for test data. We consider the second and third output layers to play an important role, even though they cannot learn potential intentions. There is usually a lot of inadequate data to train, and these data disrupt the training of the model. However, in most cases we cannot distinguish useful data from disturbance data. In the competitive learning architecture, each output layer automatically selects data to train its computation path. This selection might distinguish useful data from disturbance data, and the first output layer can be trained using just the useful data, even though other output layers are trained using inadequate data. In this way, competitive learning can train the model such that it is robust against noisy data.

5 Conclusion

We proposed supervised competitive learning to imitate a driver’s potential intentions. Competitive learning was applied to supervised learning and the squared error was reduced to 1/25 that of a conventional method. We also demonstrated that competitive learning can distinguish valid data from disturbance data to train a model.

References

Stanley C Ahalt, Ashok K Krishnamurthy, Prakoon Chen, and Douglas E Melton. Competitive learning algorithms for vector quantization. *Neural networks*, 3(3):277–290, 1990.

Ralph Birnbaum and Jerry Truglia. *Getting to Know OBD II*. New York, 2000.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

A. Kornhauser C. Chen, A. Seff and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of 15th IEEE International Conference on Computer Vision*, 2015.

Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Tobias Gindele, Sebastian Brechtel, and Rüdiger Dillmann. A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 1625–1631. IEEE, 2010.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 593–605. IEEE.

SAE International. *On-Board Diagnostics for Light and Medium Duty Vehicles Standards Manual*. Pennsylvania, 2003.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Xiaoliang Ma and Ingmar Andréasson. Driver reaction time estimation from real car following data and application in gm-type model evaluation. In *Proceedings of the 85th TRB annual meeting*, pp. 1–19, 2006.

Chulwoo Moon and Seibum B Choi. A driver model for vehicle lateral dynamics. *International journal of vehicle design*, 56(1-4):49–80, 2011.

Osonde Osoba and Bart Kosko. Noise-enhanced clustering and competitive learning algorithms. *Neural Networks*, 37:132–140, 2013.

Takashi Shinozaki and Yasushi Naruse. Competitive learning with feedforward supervisory signal for pre-trained multilayered networks. *arXiv preprint arXiv:1312.5845*, 2013.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. URL `http://learningsys.org/papers/LearningSys_2015_paper_33.pdf`.

Takahiro Wada, Shun'ichi Doi, Keisuke Imai, Naohiko Tsuru, Kazuyoshi Isaji, and Hiroshi Kaneko. On driver's braking behavior in car following. In *SICE, 2007 Annual Conference*, pp. 2396–2401. IEEE, 2007.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.