PAPER

Classifying Near-Miss Traffic Incidents through Video, Sensor, and Object Features*

Shuhei YAMAMOTO[†a)], ***Nonmember***, **Takeshi KURASHIMA**[†], ***and*** **Hiroyuki TODA**[†], ***Members***

SUMMARY Front video and sensor data captured by vehicle-mounted event recorders are used for not only traffic accident evidence but also safe-driving education as near-miss traffic incident data. However, most event recorder (ER) data shows only regular driving events. To utilize near-miss data for safe-driving education, we need to be able to easily and rapidly locate the appropriate data from large amounts of ER data through labels attached to the scenes/events of interest. This paper proposes a method that can automatically identify near-misses with objects such as pedestrians and bicycles by processing the ER data. The proposed method extracts two deep feature representations that consider car status and the environment surrounding the car. The first feature representation is generated by considering the temporal transitions of car status. The second one can extract the positional relationship between the car and surrounding objects by processing object detection results. Experiments on actual ER data demonstrate that the proposed method can accurately identify and tag near-miss events.

key words: *deep neural network, event recorder, near-miss traffic incident, multi-modal data, time-series data*

1. Introduction

Recently, the event recorder has become an almost obligatory car accessory. Modern recorders can capture a front video, several sensor streams, and driving operation. The event recorder permanently stores all data dozens of seconds on either side of the trigger of longitudinal/lateral acceleration/deceleration exceeding a certain level. In this paper, we call such data event recorder (ER) data. ER data is being effectively used as traffic accident/violation evidence. In addition, ER data that demonstrates near-miss traffic incidents ("near-miss"), such as near collisions between the car and other obstacles, is being considered for use in reducing traffic accidents. Actual examples of near-miss scenes captured by ERs are shown in Fig. 1. The ER data of near-misses is best utilized pro-active education that targets safer driving. An example of safe-driving education is to have drivers watch actual ER footage of near-miss traffic incidents [2]. In addition, near-miss incidents in ER data are attracting the attention of fleet management companies that need to control scores of commercial motor vehicles such as vans and trucks. For example, car leasing and commercial trucking companies can evaluate each driver's skills by processing the front video captured by Internet-connected cameras [3]. A car insurance company is detecting dangerous areas in town and creating hazard maps based on traffic accidents or near-miss as found in ER data [4]. As just described, various services/applications are using the near-miss events present in ER data; they represent new opportunities for eliminating or minimizing the risks associated with vehicle operation.

Fig. 1 Actual examples of near-miss scenes captured by ERs. The orange circle in each image indicates a near-miss traffic incident. The left example shows a near-miss between the car and another car coming from the right side. The right example shows a near-miss between the car and a bicycle on a crosswalk.

However, most ER data doesn't include near-miss incidents ("no near-miss"). One report [5] claimed that about 70% of ER data contains no near-miss incident. This is because the acceleration limits used to trigger the ER can be exceeded by rough roads and abrupt driving inputs. Moreover, actual safe-driving education organizers expect the ER data to be tagged and sorted according the type of incident (e.g. pedestrian and bicycle) because they want to extract the best possible videos as safe-driving education material for each incident type. Unfortunately, manually identifying and labelling all near-miss incidents from the large amount of ER data available is too time consuming, expensive, and error prone. Therefore, the automating the process is essential to reducing the cost of safe-driving education and strengthening the effective use of ER data. The objective of this paper is to automatically detect the presence of near-miss incidents and then accurately identify near-miss type.

To achieve this objective, the straightforward approach is to build a multi-class classification model. ER data is multi-modal data consisting of video and sensor readings, and it is considered necessary to use all the data in combination for identifying near-miss incidents. The state of own vehicle and its surroundings is mainly determined from sensor readings and video. Both are key information for determining whether an ER data segment contains a near-miss or not. Thanks to advances in deep neural networks (DNNs), we can now handle such data by convolutional neural networks

Manuscript received January 18, 2021.
Manuscript revised July 30, 2021.
Manuscript publicized November 1, 2021.

[†]The authors are with NTT Human Informatics Laboratories, NTT Corporation, Yokosuka-shi, 239–0847 Japan.

*This paper is an extended version of PAKDD2020 "Identifying Near-Miss Traffic Incidents in Event Recorder Data" [1].

a) E-mail: shuhei.yamamoto.ea@hco.ntt.co.jp

DOI: 10.1587/transinf.2021EDP7017

(CNNs) [6] as well as recurrent neural networks (RNNs) [7]. Passing the image frame data through a CNN will yield feature vectors, and the feature vectors of image frames and sensor streams can be integrated using a full connect neural network; the resulting time-series data can be modelled by an RNN. Although this approach can detect near-miss incidents (i.e., determine the presence or absence of a near-miss event), it is not accurate in terms of classifying incidents the according to its type. There are two reasons for this failure.

Issue 1: The near-miss detection task doesn't require detailed information of the obstacle captured by the front video because it is sufficient that just some kind of obstacle is detected. This involves using a CNN to extract basic visual features. However, the task of classifying the near-miss incidents requires an understanding of the kind of object and its position relation to the car. Simple CNNs can't extract visual features with sufficient detail.

Issue 2: The task of identifying near-miss incidents can be treated as a two-level hierarchy classification task. First, each ER segment is classified into near-miss or no near-miss. Second, the near-miss object in each ER segment is identified. However, general multi-class classification frameworks don't provide such a hierarchical architecture, and instead attempt to solve the two classification tasks simultaneously (i.e. treat the task as a one-level classification task). This makes the task more complex which degrades classification accuracy.

To resolve these two issues, this paper proposes a classification method that combines a supervised DNN to process object detection results with multi-task learning. The proposed method has three main components. The first component, the Temporal Encoding Layer, generates a feature vector by encoding frame images, sensor streams, and object detection results as time-series data. The second component, the Grid Embedding Layer, creates a feature vector by embedding object detection results into a grid space by determining the positions of each object relative to the car. The third component, the Multi-task Layer, splits the main task into two sub-tasks to classify near-miss type. We conduct experiments on an actual ER dataset to evaluate the effectiveness of the proposal. Our result shows that the proposed method can well handle ER data with improved performance.

This paper is an extended version of our previous paper [1]. The enhancements are as follows. First, we conducted new experiments to clarify classification performance versus the number of information sources in Temporal Encoding Layer and at different hyper-parameters of Grid Embedding Layer and Multi-task Layer (Sect. 5.2). Second, we added a case study with actual ER data using several frame images and sensor streams (Sect. 5.3). Finally, we showed its effectiveness in the aspect of understanding the estimation results output by the proposed method (Sect. 5.4).

2. Related Works

Several studies have focused on near-miss traffic incident detection (i.e., determine the presence or absence of a near-miss event) from dashboard camera (dashcam) data. Suzuki et al. [8] estimate the risk level for each frame image in front video by using CNN, which is a highly effective DNN architectures. Their model demonstrated improved accuracy in near-miss detection by introducing pedestrian detection task as sub-function. Ke et al. [9] detect near-miss scenes using pedestrian detection from the front video captured by a vehicle-mounted camera; the distance between the car and the pedestrian is used to calculate the risk level of each frame. Kim et al. [10] analyze front video of car crashes captured during car simulator trials on variety of roads for clarifying traffic characteristics of dangerous scene. While their model detect near-miss scenes using front video, they do not consider the classification of the near-miss incidents.

Dashcam data has been used for various tasks other than near-miss detection. By extracting driver operations from dashcam data, Yokoyama et al. [11] use feature engineering to detect the drivers with dangerous driving styles. Front video is a significant part of autonomous vehicle driving technology. To permit autonomous control of vehicle movement, Jain et al. [12] predict driving movements such as straight, left/right turn, lane change, and stop based on front video information using in-vehicle cameras; their prediction model analyzes the features of the driver's face. To avoid traffic accidents, Chan et al. [13] proposed a method that anticipates accidents among vehicles. Our work differs from theirs as regards the goals and model proposed.

Our approach is motivated by the success achieved by using DNNs to analyze video data. The DNN components of CNN and RNN are widely used for human activity recognition. Baccouche et al. [14] proposed a standard approach to human activity recognition based on DNN. Their method uses CNN to extract a set of human movement features from each frame image and RNN to model their temporal transitions. Sharma et al. [15] introduced a visual attention mechanism based on DNN for extracting characteristic regions in each frame image; they used it to encode feature vectors extracted by CNN. Simonyan et al. [16] proposed a spatio-temporal approach that uses both optical flow and normal images with the intention of capturing the movements of objects present in videos. Our experiments, shown in Sect. 5, evaluates the effectiveness of human activity recognition schemes for identifying near-miss incidents.

3. Preliminary

3.1 Data Format

Each ER segment consists of a sequence of frame images combined with the data streams output by several sensors. Sequence length is taken to be the number of frames in the ER sequence, T. The sensor data at each time-step is a

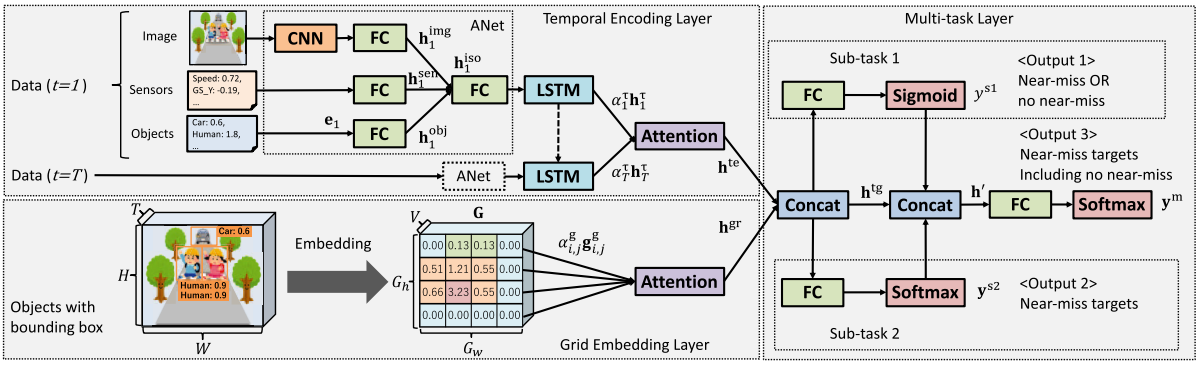

Fig. 2 Overview of our proposed DNN model.

vector consisting of several dimensions such as longitudinal/lateral acceleration and speed. We normalized the sensor data in each dimension to z-score because the dimensions have different value scales.

3.2 Object Detection

To correctly identify near-miss type, our approach uses the object detection results of image $\{I_t\}_{t=1}^{T}$. For this we employ YOLO [17], which is one of the most effective DNN-based object detection algorithms. The object detection result of image I_t consists of N_t objects. Each detected object, n, consists of the triple $\{\mathbf{o}_{t,n}, \mathbf{l}_{t,n}, \mathbf{p}_{t,n}\}$. The one-hot vector $\mathbf{o}_{t,n} = \{o_{t,n,v}\}_{v=1}^{V}$ is the object type where V is the number of object types, and the bounding box vector, $\mathbf{l}_{t,n} = \{x_{t,n}^{\text{lef}}, y_{t,n}^{\text{top}}, x_{t,n}^{\text{rig}}, y_{t,n}^{\text{bot}}\}$, specifies the object's coordinates (left, top, right and bottom) in the image; the detection probability vector $\mathbf{p}_{t,n} = \{p_{t,n,v}\}_{v=1}^{V}$.

3.3 Annotation Label and Its Re-Organization

The application of supervised machine learning is assumed to yield the correct label for near-miss target $\mathbf{y}^{\text{m}} \in \mathbb{R}^{C}$, which is one-hot vector consisting of the number of label types C. We extract two additional kinds of correct labels by re-organizing the near-miss target label $\mathbf{y}^{\text{m}}$. The first additional label, y^{s1}, identifies *near-miss* ($y^{\text{s1}} = 1$) or *no near-miss* ($y^{\text{s1}} = 0$). The second one, one-hot vector $\mathbf{y}^{\text{s2}} \in \mathbb{R}^{C-1}$, identifies the near-miss incidents for each ER sequence other than those identified as *no near-miss*.

4. Proposed Method

In this section, we describe the proposed method; it uses DNN to classify the occurrence targets of "near-miss". The proposed method is composed of three main components (Fig. 2). The first component, the Temporal Encoding Layer, creates a feature vector by encoding frame image, sensor streams, and object detection results and uses RNN to handle their temporal transitions. The second component, the Grid Embedding Layer, uses DNN to create a feature vector by encoding feature scores; the object detection results, bounding box information of each object, are embedded into a grid space. The third component, the Multi-task Layer, uses these feature vectors to perform sub-tasks and then uses their results to complete the main task. We describe these components in Sects. 4.1, 4.2, and 4.3, respectively.

4.1 Temporal Encoding Layer (TEL)

The objective of this layer is to generate a feature vector by considering the temporal transitions present in the time-series data.

Image encoder: To obtain holistic features such as the surrounding environment from front video, we encode each video image into a feature vector by using CNN. Here, to extract visual features from each image, we prepare two types of GoogLeNets [18] pretraind by ImageNet [19] and Places365 [20]. The GoogLeNets of this paper encode each image I_t into two feature vectors. Next, these feature vectors are encoded by a full connect neural network (FC) into a feature vector with dimension of U. The feature vector extracted by this process from frame number t is denoted as $\mathbf{h}_t^{\text{img}}$.

Sensor encoder: To obtain features that describe the car status, we use FC to encode the sensor data into a feature vector with U dimensions. The feature vector extracted by this process from frame number t is denoted as $\mathbf{h}_t^{\text{sen}}$.

Object encoder: To extract in detail features such as obstacles and traffic signs present in the front video, we use the object detection results after translating them into a simple vector representation. Here, we focus on the appearance degree of each object and generate vector $\mathbf{e}_t$, which refers to the number of object types, V. The score is calculated by $\mathbf{e}_t = \sum_{n=1}^{N_t} \mathbf{o}_{t,n} \cdot \mathbf{p}_{t,n}$. If several identical objects are detected in an image, in order to enhance the appearance degree of the object, the score is calculated by summing object detection probability $\mathbf{p}_{t,n}$. Next, the generated feature vector $\mathbf{e}_t$ is encoded into a feature vector with U dimensions by FC; this yields, for frame number t, $\mathbf{h}_t^{\text{obj}}$.

Time-series modeling: This unit concatenates the three feature vectors ($\mathbf{h}_t^{\text{iso}} = [\mathbf{h}_t^{\text{img}}; \mathbf{h}_t^{\text{sen}}; \mathbf{h}_t^{\text{obj}}]$) and encodes the results into a feature vector with U dimensions using by FC. Next, we use the LSTM unit to model the fea-

ture vectors in each time-step [21] and derive here a new feature vector that is more direct as it assesses the feature vectors in all time-steps by fusing with the soft attention mechanism [22]. Denoting the sequence of feature vectors obtained by the LSTM unit as $\{\mathbf{h}_t^\tau\}_{t=1}^T$, the soft attention mechanism calculates a new feature vector, $\mathbf{h}^{\mathrm{te}}$, as follows: $\mathbf{h}^{\mathrm{te}} = \sum_{t=1}^T \alpha_t^\tau \mathbf{h}_t^\tau$, $\alpha_t^\tau = \mathrm{softmax}(\mathbf{u}_t^{\mathrm{T}} \mathbf{u}^\tau)$, $\mathbf{u}_t = \tanh(\mathbf{W}^\tau \mathbf{h}_t^\tau + \mathbf{b}^\tau)$, where $\mathbf{W}^\tau \in \mathbb{R}^{U \times U}$, $\mathbf{b}^\tau \in \mathbb{R}^U$, $\mathbf{u}^\tau \in \mathbb{R}^U$ are the model parameters of DNN.

4.2 Grid Embedding Layer (GEL)

Grid embedding: The objective of this layer is to derive a feature vector that can be used to identify near-miss targets; it does so by considering the bounding box information of each object in each frame image. In this paper, we propose a grid embedding method for utilizing bounding box information; we focus on the position of each object in the image and consider the position relationship between the car and each object. This method prepares grid space $\mathbf{G} \in \mathbb{R}^{G_h \times G_w \times V}$ by setting appropriate vertical and horizontal grid dimensions (G_h and G_w); it then embeds the objects into grid space $\mathbf{G}$. The embedded grid feature matrix $\mathbf{G}$ is generated by Algorithm 1. An example of the grid embedding flow is shown in Fig. 3.

As the embedding score for each cell, we employ the 2D area ratio r because we prioritize the distance between the car and each object. We think that the area ratio can represent the distance between the car and each object in the image because the area ratio of an object is inversely proportional to its distance from the car, i.e., objects close to the car have larger area ratios than far objects.

Encoding grid features: We can obtain grid features $\mathbf{g}_{i,j}$ by the above processes. Not all cells are important in the task of identifying near-miss incidents because the image captured strongly depends on the setting position of the ER. For example, as shown in Fig. 1, the car's bonnet occupies significantly different parts of the image if the ER's direction and position are changed. Moreover, such cells don't contribute to achieving our goal because objects will not appear there. Therefore, it is not appropriate to directly use grid importance.

In this paper, we employ the soft attention mechanism to calculate a new feature vector $\mathbf{h}^{\mathrm{gr}}$ as follows: $\mathbf{h}^{\mathrm{gr}} = \sum_{i=1}^{G_h} \sum_{j=1}^{G_w} \alpha_{i,j}^{\mathrm{g}} \mathbf{g}_{i,j}$, $\alpha_{i,j}^{\mathrm{g}} = \mathrm{softmax}(\mathbf{u}_{i,j}^{\mathrm{T}} \mathbf{u}^{\mathrm{g}})$, $\mathbf{u}_{i,j} = \tanh(\mathbf{W}^{\mathrm{g}} \mathbf{g}_{i,j} + \mathbf{b}^{\mathrm{g}})$, where $\mathbf{W}^{\mathrm{g}} \in \mathbb{R}^{V \times U}$, $\mathbf{b}^{\mathrm{g}} \in \mathbb{R}^U$, $\mathbf{u}^{\mathrm{g}} \in \mathbb{R}^U$ are the DNN model parameters. These formulas mean that attention weight $\alpha_{i,j}^{\mathrm{g}}$ is dynamically estimated from grid feature $\mathbf{g}_{i,j}$ as grid importance, and the feature vector $\mathbf{h}^{\mathrm{g}}$ is calculated based on attention weights and grid features.

4.3 Multi-Task Layer (MTL)

The objective of this layer is to identify the near-miss target using the two feature vectors obtained in Sects. 4.1 and 4.2, respectively. First, we concatenate the feature vectors $\mathbf{h}^{\mathrm{tg}} = [\mathbf{h}^{\mathrm{te}}; \mathbf{h}^{\mathrm{gr}}]$. Here, we utilize a multi-task learning framework

ALGORITHM 1: Grid Embedding

Input: $\{\{\mathbf{o}_{t,n}\}_{n=1}^{N_t}\}_{t=1}^T$, $\{\{\mathbf{l}_{t,n}\}_{n=1}^{N_t}\}_{t=1}^T$, H, W, G_h, G_w
Output: $\mathbf{G}$
Initialize: $\mathbf{G} \in \mathbb{R}^{G_h \times G_w \times V} \leftarrow 0$;
$(S_w, S_h) \leftarrow (W/G_w, H/G_h)$;
for $t = 1$ *to* T **do**
  for $n = 1$ *to* N_t **do**
    $(x1, y1) \leftarrow (\lceil x_{t,n}^{\mathrm{lef}}/S_w \rceil, \lceil y_{t,n}^{\mathrm{top}}/S_h \rceil)$;
    $(x2, y2) \leftarrow (\lceil x_{t,n}^{\mathrm{rig}}/S_w \rceil, \lceil y_{t,n}^{\mathrm{bot}}/S_h \rceil)$;
    $r \leftarrow \{(x_{t,n}^{\mathrm{rig}} - x_{t,n}^{\mathrm{lef}}) \times (y_{t,n}^{\mathrm{bot}} - y_{t,n}^{\mathrm{top}})\}/(W \times H)$;
    for $i = y1$ *to* $y2$ **do**
      for $j = x1$ *to* $x2$ **do**
        $\mathbf{g}_{i,j} \leftarrow \mathbf{g}_{i,j} + \mathbf{o}_{t,n} \cdot r$

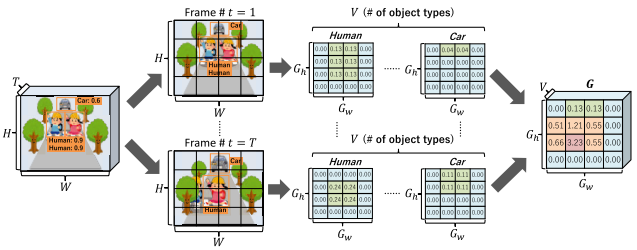

Fig. 3 A worked example of grid embedding flow.

by setting two simple sub-tasks as part of the main task.

The first sub-task determines the presence or absence of a near-miss event for each ER sequence. We encode $\mathbf{h}^{\mathrm{tg}}$ to scalar value $\hat{y}^{\mathrm{s1}}$, which is the output of this sub-task, by using FC and sigmoid function and calculating cross entropy error L^{s1} between the correct label y^{s1} and result $\hat{y}^{\mathrm{s1}}$ as follows: $L^{\mathrm{s1}} = -\frac{1}{D} \sum_d^D \{y^{\mathrm{s1}} \log \hat{y}^{\mathrm{s1}} + (1 - y^{\mathrm{s1}}) \log (1 - \hat{y}^{\mathrm{s1}})\}$, where D and d are the number of training data and the index used in scanning the training data, respectively; d is used to link y^{s1} and $\hat{y}^{\mathrm{s1}}$, but is omitted in this paper.

The second sub-task identifies the near-miss incidents for each ER sequence other than those identified as *no near-miss*. We encode $\mathbf{h}^{\mathrm{tg}}$ into vector $\hat{\mathbf{y}}^{\mathrm{s2}}$, the result of this sub-task, by FC and softmax function, and then calculate cross entropy error L^{s2} between the correct label $\mathbf{y}^{\mathrm{s2}}$ and result $\hat{\mathbf{y}}^{\mathrm{s2}}$ as follows: $L^{\mathrm{s2}} = -\frac{1}{D} \sum_d^D \sum_k^{C-1} y_k^{\mathrm{s2}} \log \hat{y}_k^{\mathrm{s2}}$.

We concatenate the results using the form $\mathbf{h}' = [\mathbf{h}^{\mathrm{tg}}; \hat{y}^{\mathrm{s1}}; \hat{\mathbf{y}}^{\mathrm{s2}}]$. We can now consider the results of these simple sub-tasks. We encode $\mathbf{h}'$ into vector $\hat{\mathbf{y}}^{\mathrm{m}}$ which represents the result of the main task, by FC and softmax function and calculate cross entropy error L^{m} between the correct label $\mathbf{y}^{\mathrm{m}}$ and result $\hat{\mathbf{y}}^{\mathrm{m}}$ as follows: $L^{\mathrm{m}} = -\frac{1}{D} \sum_d^D \sum_k^C y_k^{\mathrm{m}} \log \hat{y}_k^{\mathrm{m}}$.

We optimize the objective function $L = L^{\mathrm{m}} + \beta \cdot (L^{\mathrm{s1}} + L^{\mathrm{s2}})$ which includes the errors of these three tasks. β denotes the hyper-parameter used for controlling sub-task errors. The label yielded by the main task is given by extracting the index with maximum score from the result $\hat{\mathbf{y}}^{\mathrm{m}}$.

The general aim of multi-task learning is to leverage useful information contained in multiple related tasks to improve the generalization performance. Learning multiple tasks jointly can lead to significant performance improve-

ments compared with learning them individually as has been several related works [23], [24]. For example, [23] jointly learns representations of words, entities, and meaning representations via multi-task learning. [24] shows the effectiveness of this approach with regard to various natural language processing tasks including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. Following the success of these multi-task learning approaches, the innovation of our multi-task learning is to learn a classifier specific to each sub-task; we extract effective features and obtain new feature vectors for performing each sub-task. All features including frame image, sensor streams, and object detection results can be useful for determining whether each video contains a near-miss incident or not. However, once we know that a video contains a near-miss, object detection results constitute the most helpful information for determining the near-miss targets because they allow us to understand the kind of objects around the car. While single task learning must learn such features implicitly, our multi-task learning can learn them explicitly as isolated features.

5. Experimental Evaluations

5.1 Dataset and Parameter Setting

The experimental evaluation uses the Near-Miss Incident Database provided by the Smart Mobility Research Center of Tokyo University of Agriculture and Technology in Japan. The dataset is a collection of data captured by ERs mounted in Japanese taxis. Each ER data sequence is 15 seconds long; 10 seconds before the trigger and 5 seconds after the trigger. Each sequence was manually assigned one of five risk levels {*high, middle, low, bit, no near-miss*} and six near-miss incident types {*car, bicycle, motorcycle, pedestrian, self, other*} by experts. The experiment focused on five near-miss incidents {*car, bicycle, motorcycle, pedestrian, self*[†]}. 700 sequences were randomly extracted for each near-miss incident type. 700 sequences tagged *no near-miss* were also randomly extracted. Therefore, the experiment examined 4,200 sequences with 6 labels ($C = 6$). We randomly split the dataset into 2,940 (70%) sequences as training data and 1,260 (30%) as test data.

Each sequence was recorded at 30 frames per second and so consisted of 450 frames. In this paper, we sampled $T = 30$ frames at intervals of 15 frames. Each image had resolution of $W = 640$ and $H = 400$ in RGB format. The original images were processed by YOLO for object detection. This yielded $V = 69$ object types. For visual feature extraction, linearly transformed images (224×224 byte resolution) were processed by two GoogLeNets. For the sensor data, we extracted three sensor streams: speed and longitudinal/lateral acceleration.

For the DNN in the proposed method, we set the number of hidden units in each FC to $U = 256$, and the output vector after each FC is non-linearly transformed by the ReLu function [25] with Dropout $p = 0.7$ [26]. We optimized the DNN according to Adam [27] based on the objective function of L gradient as calculated by the back propagation method. Here, we set the mini-batch size to 50 and the back propagation iteration number to 100. With regard to GoogLeNet, we used the Caffe [28] model pretrained by ImageNet and Places365, and updated the parameters in the output layer by fine-tuning.

[†]*self* refers to a dangerous or illegal movement involving only the car

Table 1 Classification performance versus number of information sources in Temporal Encoding Layer and SVM. "V", "S", and "O" mean "Video", "Sensor", and "Objects", respectively.

Sources			Ours			SVM		
V	S	O	Pre.	Rec.	F1.	Pre.	Rec.	F1.
✓			43.47	42.14	42.22	40.19	42.06	40.73
	✓		41.60	42.62	41.60	40.66	40.71	40.39
		✓	57.75	55.95	55.53	51.29	48.41	48.25
✓	✓		49.09	49.52	49.02	43.94	43.41	43.54
✓		✓	63.38	63.17	63.11	56.59	55.16	54.12
	✓	✓	62.25	61.90	61.94	54.91	54.76	54.27
✓	✓	✓	**64.10**	**64.29**	**64.10**	57.64	58.25	57.64

5.2 Results

We conducted four experiments. Experiment 1 and 2 address issue 1, while experiment 3 addresses issue 2. In addition, experiment 4 is intended to verify the effectiveness of combining the three proposed components. To examine the effectiveness of the proposed method in identifying near-miss incidents, we use four evaluation metrics: accuracy, precision, recall, and F1-score [29].

Experiment 1: Table 1 shows the classification performance versus the number of information sources (front video, sensor streams, and objects) as processed by TEL. To focus on the improvement in classification performance by the addition of information sources, we did not apply GEL or MTL. To classify the near-miss target without using MTL, we employ a full connect neural network (FC) as the output layer. FC decodes feature ($\mathbf{h}^{\mathrm{te}}$) to a vector with the number of labels ($C = 6$). After that, we extract the label having maximum value in the vector as prediction label. Also, to confirm the effectiveness of feature encoding and the temporal transitions modeling by CNN and RNN, we make comparisons using Support Vector Machine (SVM) with different information sources (features) SVM is implemented by LIBSVM. The best hyper-parameters of SVM such as kernel type, cost parameter, and RBF kernel's γ were selected by grid-search. In order to use the ER sequence as SVM input, we transformed each information source into a vector space and concatenated them over all frames. In the case of a single information source, the highest evaluation values for all metrics was achieved with the use of detected objects. For the case of two information sources, the best performance was achieved by using sensor streams and detected objects. This confirms the effectiveness of using detected objects as an information source.

On the other hand, although the method of using front video and sensor streams is the straightforward approach when using DNN as explained in Sect. 1, its evaluation values are lower than those of the proposed method using detected objects only. Excepting this case, we can confirm that performance generally improves with the addition of information sources. For each combination of features (for each line in Table 1), the proposal always yields better evaluation scores than SVM. This indicates the effectiveness of TEL using CNN and RNN, which is one of the components of the proposed method.

Experiment 2: Table 2 (a) shows the classification performance for three GEL grid sizes (G_h, G_w). To focus on the impact of grid size on classification performance, we used three information sources in TEL and did not use MTL. To classify the near-miss target without using MTL, we employ a full connect neural network (FC) as output layer. The FC decodes feature ($\mathbf{h}^{\mathrm{tg}}$) to a vector with the number of labels ($C = 6$). After that, we extract the label having maximum value in the vector as prediction label. We selected the grid sizes that maintained the aspect ratio of the original image $(400 : 640) = (5 : 8)$. Also, row $(-, -)$ in the table shows the evaluation values without GEL. Regardless of grid size, using GEL yielded higher evaluation values, so it does improve the classification performance. Moreover, we can confirm that the performance increases with the grid resolution. This result suggests that GEL can extract detailed features from the object detection results. The following sections use the grid size of $(G_h, G_w) = (8, 10)$. However, we confirmed that the classification performance did not increase even if the grid resolution was increased from $(8, 10)$ to $(16, 20)$. We consider that there are two reasons for this. The first is that high resolution makes training difficult by increasing the number of attention parameters. To be specific, the number of attention parameters $\alpha^{\mathrm{g}}_{i,j}$ is quadrupled if grid resolution $(8, 10)$ is increased to $(16, 20)$. This suggests that high resolution grids need more complex training than low resolution ones. The second is that high resolution grids make it harder to utilize the co-occurrence relation of objects in each cell feature $\mathbf{g}_{i,j}$. In the example of the near-miss shown in Fig. 9, GEL can identify the near-miss target of *motorcycle* if person and motorcycle occur in the same cell feature $\mathbf{g}_{i,j}$. However, high resolution grids weaken such co-occurrence relations. Therefore, our consideration is that grid resolution triggers a trade-off problem: high resolution grids (e.g. (16,20)) can capture the detail position of each object, but make it difficult to handle the position relation among objects. For these reasons, GEL could not enhance F1-score even with a high resolution grid (16,20). In order to resolve this problem, we will consider to handle the spatial relationship among objects by capturing the distance between cells in future work.

Experiment 3: Table 2 (b) shows the classification performance for three β values {0.1, 1.0, 10.0} in MTL. To focus on the ability of MTL to improve the classification performance and assess the effect of varying β, we used three information sources in TEL and did not use GEL. Note that the entries on the row marked "-" indicate the evaluation values attained without MTL. The results show that, regardless of the β value, using MTL yields higher performance. Therefore, MTL does improve the classification performance. Also, we can confirm that the β value of 10.0 yields lower F1-score than the other values. This suggests that the estimation accuracy of the main task is degraded if sub-task error is excessively weighted. The following sections use $\beta = 0.1$ because it achieved the highest F1-score.

Table 2 F1-score at different hyper-parameters of the proposed.

(a) Different grid sizes

G_h	G_w	**F1-score**
-	-	64.10
5	8	64.66
8	10	**65.19**
16	20	**65.19**

(b) Different β values

β	**F1-score**
-	64.10
0.1	**65.56**
1.0	64.97
10.0	64.55

Table 3 Classification performance of each method. "V", "S", and "O" mean "Video", "Sensor", and "Objects", respectively.

Method	**V**	**S**	**O**	**Pre.**	**Rec.**	**F1.**
DNN	✓	✓		49.09	49.52	49.02
SVM	✓	✓	✓	57.64	58.25	57.64
IDT [30]	✓			43.20	45.37	44.09
ST-CNN [16]	✓			49.25	51.03	48.99
DSA [13]	✓		✓	60.39	58.36	59.63
Proposed(V)	✓			44.55	43.13	43.39
Proposed(V,O)	✓		✓	63.52	64.98	64.01
Proposed	✓	✓	✓	**65.75**	**65.79**	**65.68**

Experiment 4: We show the classification performance with the proposed and five baseline methods in Table 3. The baselines are as follows.

DNN: Straightforward approach using DNN (i.e., TEL without objects).

SVM: SVM using three different information sources, which performs best in Experiment 1.

IDT [30]: It was proposed for recognizing human activity in video and is one of the SOTA methods for extracting video features; IDT identifies several visual key points and uses the trajectories of key points to characterize each video. We set the trajectory length to 15 and and the sampling stride to 5. Each video is then converted into a K-dimensional feature vector by K-means clustering all videos (we set K to 200). We use the IDT-based features to train the SVM classifier. The best hyper-parameters of SVM are selected by same manner to Experiment 1.

ST-CNN [16]: It was proposed for human activity recognition in video and is another SOTA method. The method combines two types of DNNs; one is the spatial convolution network for capturing scenes and objects depicted in the video, and the other is a temporal convolution network for capturing motions between frames. ST-CNN calculates average scores for these two feature vectors. Their label is given by extracting the index with maximum score from the result. We set the number of stacked images to 10.

DSA [13]: It was proposed for anticipating traffic acci-

dents among vehicles from front video. The method extracts two visual features; one is the object feature for capturing object movement between continuous frames, and the other is a horistic feature for each frame image (i.e., $\mathbf{h}_t^{\text{img}}$). We extract DSA-based features from front video and train a DNN composed from temporal encoder (LSTM) and classifier layer. We set the number of candidate objects to 10.

In addition to these methods, we prepared three variants of the proposed method. The first and second methods are Proposed(V) and Proposed(V,O), each of which inputs video only and video and objects to allow comparison, in fair conditions, to ST-CNN/IDT and DSA, respectively. The final one is Proposed, which is the full model using video, sensor, and objects.

For all evaluation metrics, the proposed method achieved the highest values among the compared methods. The results indicate the effectiveness of our approach in terms of the near-miss traffic identification task for ER data.

Note that we conducted the χ^2 test based on cross tabulation (joint frequency distribution of cases/tests) with two categorical variables (i.e., proposed and each baseline); each variable can be correct or incorrect. The results confirmed that the proposed method is significantly better than the baselines (p-value < 0.01).

Figure 4 shows the precision, recall, and F1-score in each class for proposed, SVM, and ST-CNN. Of particular interest, for the four labels of *car*, *bicycle*, *motorcycle*, and *pedestrian*, the precision and recall scores were higher than the two other methods. Also, we can confirm that the proposed method achieved the highest score in all labels of F1-score. Figure 5 uses a confusion matrix to show the detailed classification results of Proposed, SVM, and ST-CNN. In this figure, the true labels and predicted labels are plotted on the horizontal and vertical axes, respectively. The number in each cell shows the number of tests with each label. The proposed method accurately identified more objects than the other two methods, except for *no near-miss* and *self* labels, which confirms the superior performance of the proposed method.

When we focus on the results in fair condition between the proposed and baselines, we confirmed that Proposed(V,O) yielded higher evaluation scores than DSA. The main difference between the proposed method and DSA is that the proposed method can consider the position relationship between the car and each object by using GEL. This result suggests that the task of identifying near-miss targets is important to capture each object's position. Regarding to the results attained when inputting video only, Proposed(V) had lower evaluation score than ST-CNN. Although ST-CNN can process pixel-level movements among sequential images as a feature through optical flow, the proposed method processes holistic features of each image by CNN independently. Due to this difference, the proposed method cannot obtain features effective for identifying near-miss targets from video only. However, when we use the proposed

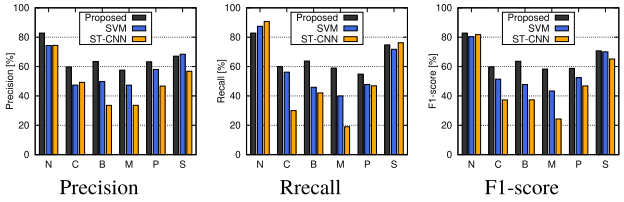

Fig. 4 Precision, recall, and F1-score values in each class for Proposed, SVM, and ST-CNN methods. X-axis plots the labels "N", "C", "B", "M", "P", and "S" mean "No near-miss", "Car", "Bicycle", "Motorcycle", "Pedestrian", and "Self", respectively.

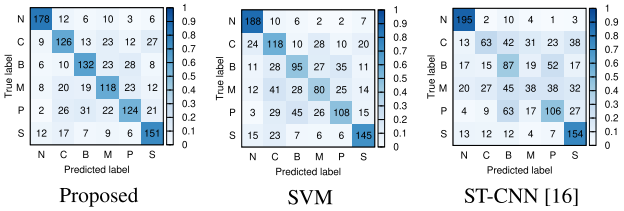

Fig. 5 Confusion matrix for classification results of Proposed, SVM, and ST-CNN methods. "N", "C", "B", "M", "P", and "S" mean "No near-miss", "Car", "Bicycle", "Motorcycle", "Pedestrian", and "Self", respectively.

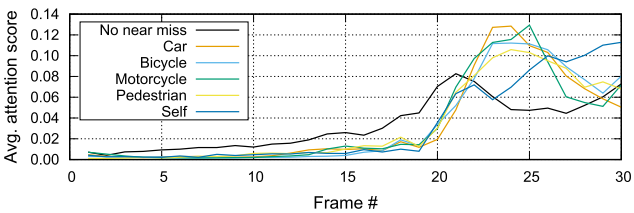

Fig. 6 Averaged attention score for each label in TEL. The vertical and horizontal axes are averaged α_t^τ over test data and frame number, respectively.

method in practical situations, we can use object detection for video images and handle detected objects by TEL and GEL. In this case, the proposed method attained higher scores than ST-CNN (i.e., Proposed(V,O) vs. ST-CNN in Table 3). Therefore, we consider that the proposed method is more effective than ST-CNN in the task examined.

5.3 Qualitative Analysis

The proposed method uses soft attention for temporal and grid space processing in TEL and GEL, respectively. By calculating mean values of each soft attention of α_t^τ and $\alpha_{i,j}^{\text{g}}$ for the correct labels in the test data, we can compare the time and space attributes emphasized by the proposed method.

The mean attention scores α_t^τ calculated for each correct label are shown in Fig. 6. The vertical and horizontal axes are averaged attention scores α_t^τ over test data and frame number t. The trigger frame number is $t = 20$. The scores of the near-miss targets of *car*, *bicycle*, *motorcycle*, and *pedestrian* peaked at around frame number $t = 25$. On the other hand, the *self* label attained highest attention score toward the last frame, $t = 30$, while *no near-miss* attained its peak score at frame number $t = 21$. As demonstrated by

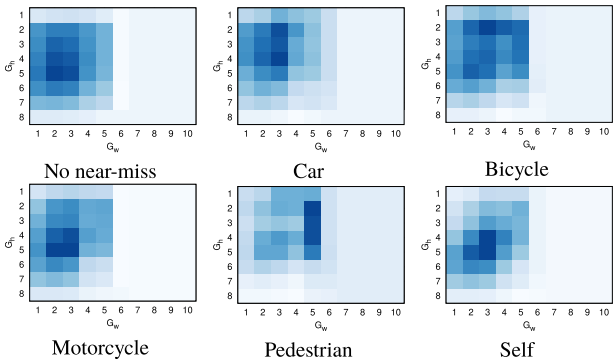

Fig. 7 Averaged attention score for each label in GEL. The color intensity represents the mean attention score $\alpha^{\mathrm{g}}_{i,j}$ value in each cell.

Fig. 8 This example shows a car (on the left) cutting in front of the driver. Detected objects are shown by bounding box, and the grid attention scores $\alpha^{\mathrm{g}}_{i,j}$ yielded by the proposed method are visualized by pale red tint in each cell. The proposed method uses the attention scores to emphasizes the dangers.

these results, the labels of *self* and *no near-miss* have different characteristics from the other labels; the four other labels demonstrate a similar tendency in terms of α^{τ}_{t}.

The mean attention scores $\alpha^{\mathrm{g}}_{i,j}$ calculated for each correct label are shown in Fig. 7. In this figure, the color intensity represents the mean attention score value in each cell. Cells on the left side of all figures are higher than those on the right. We think this is because vehicles and bicycles drive on the left side in Japan. Cells in the low center region have lower values as this region is often occupied by the car's bonnet. The *pedestrian* label has high attention scores in the vertical column of center cells. This result suggests that pedestrians frequently appeared in this region. We consider that GEL contributes the improvement of estimation performance by considering grid importance when processing ER data.

Our proposed method well supports safe-driving education. One its greatest advantages lies in is risk prediction training [2]. This involves drivers watching ER data containing near-miss traffic incidents and predicting the causes of the near-miss incidents. Figure 8 shows an example of a visualization tool that supports risk prediction training. This tool encourages drivers to focus on the precursors of near-miss events. In the frame image of Fig. 8, objects detected by the tool are shown by bounding boxes, and attention scores $\alpha^{\mathrm{g}}_{i,j}$ are visualized by the red tint in each cell. In this example, a near-miss event occurred because the car on the left turned right too sharply. The proposed method can estimate and highlight dangerous areas/objects for drivers as shown in this example. We believe that such information will greatly enhance the effectiveness of safe-driving education by more intuitively indicating what traffic targets should be focused on while driving.

We confirm the proposal's performance on actual ER data using several frame images and sensor streams. The example given in Fig. 9 shows a near-miss incident involving a *motorcycle*. The proposed method correctly determined the label, while the baseline method output the wrong label of *pedestrian*. In this example, the car stopped temporarily ($t \leq 15$), and restarted after the motorcycle crossed the intersection ($16 \leq t \leq 19$); the car braked suddenly because

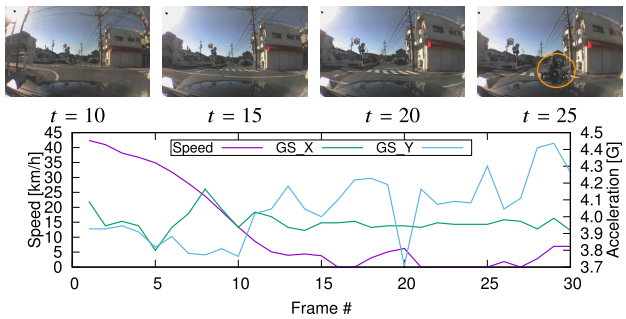

Fig. 9 Example. The near-miss target of this example is *motorcycle*, which is indicated by the orange circle at $t = 25$ image. The proposed method correctly assigned *motorcycle* as the label of this event. The baseline method incorrectly assigned *pedestrian* to the same example. GS_X and GS_Y are lateral and longitudinal accelerations, respectively.

of the motorcycle ($t = 20$). In this example, a motorcycle, which is the near-miss target, appeared in the front video at frame number $t = 25$, a few frames after the trigger time $t = 20$. Identifying *motorcycle* as a near-miss target is difficult for two reasons. The first is that object-based methods such as DSA classify *pedestrians* with high probability because YOLO detects person and motorcycle simultaneously. The second is that video processing methods such as ST-CNN classify *car* or *bike* because their movement is similar to *motorcycle*. However, the proposed method can handle multi-modal information of image features, sensor data, and detected objects, and so can correctly classify the example to *motorcycle*. Also, we think that the task of identifying near-miss incidents requires an analysis of not only the trigger time frame but also all frames. This is also suggested from an analysis of the soft attention scores shown in Fig. 6. The proposed method can correctly determine labels because it considers the object detection results output by TEL and GEL.

6. Conclusion

This paper proposed a classification method that can well utilize in a coherent manner the data provided by front

video, sensor streams, and object detection results, to accurately label near-miss events in the data captured by ERs (dashcams). The proposed method has three components. Temporal Encoding Layer; feature encoding for multi-modal and time-series data. Grid Embedding Layer; feature embedding to place detected objects into the grid space set relative to the vehicle. Multi-task Layer; multi-task learning utilizing sub-tasks developed from the main task. An experiment using actual ER data confirmed the performance improvements attained by the proposed components.

We intend to develop a semi-supervised model to handle small amounts of training data and extend the model to support the multi-labeling of events. Also, to achieve higher performance, we will study new approach for encoding detected objects by utilizing Graph Neural Network [31].

References

[1] S. Yamamoto, T. Kurashima, and H. Toda, "Identifying near-miss traffic incidents in event recorder data," PAKDD, vol.12085, pp.717–728, 2020.
[2] M. Tada, H. Noma, A. Utsumi, M. Segawa, M. Okada, and K. Renge, "Elderly driver retraining using automatic evaluation system of safe driving skill," IET Intelligent Transport Systems, vol.8, no.3, pp.266–272, 2014.
[3] samsara, "Samsara for fleets," 2016. Retrieved Nov. 2, 2018 from https://www.samsara.com/pdf/docs/samsara-for-fleets.pdf.
[4] Pioneer corporation, "Pioneer builds a map-based accident prediction platform and intelligent pilot, an adas solution for on-the-road cars," 2016. Retrieved Nov. 2, 2018 from https://global.pioneer/en/news/press/2016/pdf/1125-1.pdf.
[5] M. Kamata, M. Fujita, M. Shino, M. Nagai, Y. Michitsuji, and K. Maeda, "Research on incident analysis using drive recorder part 1: Toward database construction," FISITA World Congress, pp.22–27, 2006.
[6] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biol. Cybern., vol.36, no.4, pp.193–202, 1980.
[7] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," Interspeech 2010, pp.1045–1048, 2010.
[8] T. Suzuki, Y. Aoki, and H. Kataoka, "Pedestrian near-miss analysis on vehicle-mounted driving recorders," MVA, pp.416–419, 2017.
[9] R. Ke, J. Lutin, J. Spears, and Y. Wang, "A cost-effective framework for automated vehicle-pedestrian near-miss detection through onboard monocular vision," CVPRW, pp.898–905, 2017.
[10] H. Kim, K. Lee, G. Hwang, and C. Suh, "Crash to Not Crash: Learn to identify dangerous vehicles using a simulator," Proc. AAAI Conference on Artificial Intelligence, vol.33, pp.978–985, 2019.
[11] D. Yokoyama, M. Toyoda, and M. Kitsuregawa, "Understanding drivers' safety by fusing large scale vehicle recorder dataset and heterogeneous circumstantial data," PAKDD, vol.10235, pp.734–746, 2017.
[12] A. Jain, H.S. Koppula, B. Raghavan, S. Soh, and A. Saxena, "Recurrent neural networks for driver activity anticipation via sensory-fusion architecture," ICRA, pp.3118–3125, 2016.
[13] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, "Anticipating accidents in dashcam videos," ACCV, vol.10114, pp.136–153, 2017.
[14] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," Human Behavior Understanding, vol.7065, pp.29–39, 2011.
[15] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition using visual attention," ICLR workshop, 2016.
[16] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," NIPS, pp.568–576, 2014.
[17] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," CVPR, pp.6517–6525, 2017.
[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," CVPR, pp.1–9, 2015.
[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vision., vol.115, no.3, pp.211–252, 2015.
[20] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," IEEE Trans. Pattern Anal. Mach. Intell., pp.1452–1464, 2017.
[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol.9, no.8, pp.1735–1780, 1997.
[22] Z. Yang, D. Yang, C. Dyer, X. He, A.J. Smola, and E. Hovy, "Hierarchical attention networks for document classification," HLT-NAACL, pp.1480–1489, 2016.
[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," JMLR, vol.12, pp.2493–2537, 2011.
[24] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," AISTATS, pp.127–135, 2012.
[25] V. Nair and G.E. Hinton, "Rectified linear units improve restricted boltzmann machines," ICML, pp.807–814, 2010.
[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," JMLR, vol.15, pp.1929–1958, 2014.
[27] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," ICLR, 2015.
[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," Proc. 22nd ACM international conference on Multimedia, pp.675–678, 2014.
[29] C.D. Manning, P. Raghavan, and H. Schutze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[30] H. Wang and C. Schmid, "Action recognition with improved trajectories," ICCV, pp.3551–3558, 2013.
[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," International Conference on Learning Representations, 2019.

Shuhei Yamamoto researcher, NTT Human Informatics Laboratories, NTT Corporation. He receivedthe M.S. and Ph.D in informatics from University of Tsukuba, Japan, in 2014 and 2016, respectively. His current research interests data mining and machine learning. He is a member of the Information Processing Society of Japan (IPSJ).

Takeshi Kurashima distinguished Researcher, NTT Human Informatics Laboratories, NTT Corporation. He received the B.S. degree from Doshisha University, Japan, in 2004, and the M.S. and Ph.D. degrees in Informatics from Kyoto University, Japan, in 2006 and 2014, respectively. He is currently a Research Engineer at NTT Corporation, Japan. He was a visiting scholar at Stanford University from 2016 to 2017. His current research interests include data mining, machine learning, and recommender systems. He is a member of IEICE.

Hiroyuki Toda senior Research Engineer, Supervisor, NTT Human Informatics Laboratories, NTT corporation. He received a B.E, and M.E. in materials science from Nagoya University in 1997 and 1999, and a Ph.D. in computer science from University of Tsukuba in 2007. He joined NTT in 1999. His current research interests include information retrieval, and data mining. He is a member of the Information Processing Society of Japan (IPSJ), the Japanese Society for Artificial Intelligence (JSAI), the Database Society of Japan (DBSJ), and the Association for Computing Machinery (ACM).