Identifying Near-Miss Traffic Incidents in Event Recorder Data

Shuhei Yamamoto^(✉), Takeshi Kurashima, and Hiroyuki Toda

NTT Service Evolution Laboratories, NTT Corporation, 1-1 Hikarinooka,
Yokosuka-shi, Kanagawa-ken 239-0847, Japan
{shuhei.yamamoto.ea,takeshi.kurashima.uf,hiroyuki.toda.xb}@hco.ntt.co.jp

Abstract. 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.

Keywords: Deep neural network · Event recorder · Near-miss traffic incident · Multi-modal data · Time-series data

1 Introduction

Recently, the event recorder¹ 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

¹ <https://en.wikipedia.org/wiki/Dashcam>.

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. (Color figure online)

drivers watch actual ER footage of near-miss traffic incidents [17]. 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 cameras². 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 data³. 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.

However, most ER data doesn’t include near-miss incidents (“**no near-miss**”). One report [6] 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.

² <https://www.samsara.com/pdf/docs/samsara-for-fleets.pdf>.

³ <https://global.pioneer/en/news/press/2016/pdf/1125-1.pdf>.

Thanks to advances in deep neural networks (DNNs), we can now handle such data by convolutional neural networks (CNNs) [3] as well as recurrent neural networks (RNNs) [8]. 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 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.

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 (dash-cam) data. Suzuki et al. [15] 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. 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. [20] 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. [5] 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. 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. Sharma et al. [12] 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. [13] 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

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 vector consisting of several dimensions such as longitudinal/lateral acceleration and speed. We normalize the sensor data in each dimension to $\mathcal{N}(0, 1)$ because the dimensions have different value scales.

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 [10], 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 $\mathbf{p}_{t,n} = \{p_{t,n,v}\}_{v=1}^V$.

Annotation Label: The application of supervised machine learning is assumed to yield the correct label for near-miss target $\mathbf{y}^m \in \mathbb{R}^C$, 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 $\mathbf{y}^m$. The first additional label, y^{s1} , identifies *near-miss* ($y^{s1} = 1$) or *no near-miss* ($y^{s1} = 0$). The second one, one-hot vector $\mathbf{y}^{s2} \in \mathbb{R}^{C-1}$, identifies the near-miss incidents for each ER sequence other than those identified as *no near-miss*.

Fig. 2. Overview of our proposed DNN model.

4 Proposed Method

The proposed method is composed of three main components (Fig. 2). We describe these components in Sects. 4.1, 4.2, and 4.3.

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 [16] pretrained by ImageNet [11] and Places365 [21]. 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 , $\mathbf{h}_t^{\text{obj}}$.

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**
 $(x1, y1, x2, y2) \leftarrow (\lceil x_{t,n}^{\text{lef}}/S_w \rceil, \lceil y_{t,n}^{\text{top}}/S_h \rceil, \lceil x_{t,n}^{\text{rig}}/S_w \rceil, \lceil y_{t,n}^{\text{bot}}/S_h \rceil)$;
 $r \leftarrow \{(x_{t,n}^{\text{rig}} - x_{t,n}^{\text{lef}}) \times (y_{t,n}^{\text{bot}} - y_{t,n}^{\text{top}})\}/(W \times H)$;
 for $i = y1$ **to** $y2$ **do**
 for $j = x1$ **to** $x2$ **do**
 $\mathbf{g}_{i,j} \leftarrow \mathbf{g}_{i,j} + \mathbf{o}_{t,n} \cdot r$

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 feature vectors in each time-step [4] 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 [19]. Denoting the sequence of feature vectors obtained by the LSTM unit as $\{\mathbf{h}_t^T\}_{t=1}^T$, the soft attention mechanism calculates a new feature vector, $\mathbf{h}^{\text{te}}$, as follows: $\mathbf{h}^{\text{te}} = \sum_{t=1}^T \alpha_t^T \mathbf{h}_t^T$, $\alpha_t^T = \text{softmax}(\mathbf{u}_t^T \mathbf{u}^T)$, $\mathbf{u}_t = \tanh(\mathbf{W}^T \mathbf{h}_t^T + \mathbf{b}^T)$, where $\mathbf{W}^T \in \mathbb{R}^{U \times U}$, $\mathbf{b}^T \in \mathbb{R}^U$, $\mathbf{u}^T \in \mathbb{R}^U$ 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.

Fig. 3. A worked example of grid embedding flow.

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 feature vector $\mathbf{h}^g$ as follows: $\mathbf{h}^g = \sum_{i=1}^{G_h} \sum_{j=1}^{G_w} \alpha_{i,j}^g \mathbf{g}_{i,j}$, $\alpha_{i,j}^g = \text{softmax}(\mathbf{u}_{i,j}^T \mathbf{u}^g)$, $\mathbf{u}_{i,j} = \tanh(\mathbf{W}^g \mathbf{g}_{i,j} + \mathbf{b}^g)$, where $\mathbf{W}^g \in \mathbb{R}^{V \times U}$, $\mathbf{b}^g \in \mathbb{R}^U$, $\mathbf{u}^g \in \mathbb{R}^U$ are the DNN model parameters. These formulas mean that attention weight $\alpha_{i,j}^g$ is dynamically estimated from grid feature $\mathbf{g}_{i,j}$ as grid importance, and the feature vector $\mathbf{h}^g$ 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 $\mathbf{h}^{\text{tg}} = [\mathbf{h}^{\text{te}}; \mathbf{h}^g]$. Here, we utilize a multi-task learning framework 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 $\mathbf{h}^{\text{tg}}$ to scalar value $\hat{y}^{s1}$, 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 $\hat{y}^{s1}$ as follows: $L^{s1} = -\frac{1}{D} \sum_d \{y^{s1} \log \hat{y}^{s1} + (1 - y^{s1}) \log (1 - \hat{y}^{s1})\}$, 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 $\hat{y}^{s1}$, 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 $\mathbf{h}^{\text{tg}}$ into vector $\hat{\mathbf{y}}^{s2}$, the result of this sub-task, by FC and softmax function, and then calculate cross entropy error L^{s2} between the correct label $\mathbf{y}^{s2}$ and result $\hat{\mathbf{y}}^{s2}$ as follows: $L^{s2} = -\frac{1}{D} \sum_d \sum_k^{C-1} y_k^{s2} \log \hat{y}_k^{s2}$.

We concatenate the results using the form $\mathbf{h}' = [\mathbf{h}^{\text{tg}}; \hat{y}^{s1}; \hat{\mathbf{y}}^{s2}]$. We can now consider the results of these simple sub-tasks. We encode $\mathbf{h}'$ into vector $\hat{\mathbf{y}}^m$ which represents the result of the main task, by FC and softmax function and calculate

cross entropy error L^m between the correct label $\mathbf{y}^m$ and result $\hat{\mathbf{y}}^m$ as follows: $L^m = -\frac{1}{D} \sum_d \sum_k^C y_k^m \log \hat{y}_k^m$.

We optimize the objective function $L = L^m + \beta \cdot (L^{s1} + L^{s2})$ 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 $\hat{\mathbf{y}}^m$.

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 improvements compared with learning them individually as has been several related works [1, 2]. For example, [2] jointly learns representations of words, entities, and meaning representations via multi-task learning. [1] 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 Technology⁴ in Japan. The dataset is a collection of data captured by ERs mounted in Japanese taxis. Each ER data sequence is 15 s long; 10 s before the trigger and 5 s 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 $\{car, bicycle, motorcycle, pedestrian, self\}$ ⁵. 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.

⁴ <http://web.tuat.ac.jp/~smrc/research.html>.

⁵ *Self* refers to a dangerous or illegal movement involving only the car.

Table 1. Classification performance of each method

Method	Video	Sensor	Object	Precision	Recall	F1-score
DNN	✓	✓		49.09	49.52	49.02
SVM	✓	✓	✓	57.64	58.25	57.64
IDT [18]	✓			43.20	45.37	44.09
ST-CNN [13]	✓			49.25	51.03	48.99
Proposed	✓	✓	✓	65.75	65.79	65.68

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 [9] with Dropout $p = 0.7^6$ [14].

5.2 Results

To examine the effectiveness of the proposed method in identifying near-miss incidents, we use three evaluation metrics: precision, recall, and F1-score [7]. We show the classification performance with the proposed and four baseline methods in Table 1. The baselines are as follows.

- DNN:** Straightforward approach using DNN (i.e., TEL without objects).
- SVM:** SVM using three information sources (video, sensor, and object). 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.
- IDT [18]:** 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. 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.
- ST-CNN [13]:** It was proposed for recognizing human activity in video and is another SOTA method. The method combines two types of CNNs; one is a spatial CNN for capturing scenes and objects depicted in the video, and the other is a temporal CNN for capturing motions between frames. **ST-CNN** calculates average scores for these two feature vectors.

⁶ p is retention probability in Dropout.

Fig. 4. 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.

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 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.

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^T and $\alpha_{i,j}^G$ 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^T calculated for each correct label are shown in Fig. 5. The vertical and horizontal axes are averaged attention scores α_t^T 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 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_t^T .

The mean attention scores $\alpha_{i,j}^G$ calculated for each correct label are shown in Fig. 6. 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.

Fig. 5. Averaged attention score for each label in TEL. The vertical and horizontal axes are averaged α_t^T over test data and frame number, respectively.

Fig. 6. Averaged attention score for each label in GEL. The color intensity represents the mean attention score $\alpha_{i,j}^G$ value in each cell.

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.

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 (dash-cams). 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.

References

1. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: Joint learning of words and meaning representations for open-text semantic parsing. In: AISTATS, pp. 127–135 (2012)
2. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. JMLR **12**, 2493–2537 (2011)

3. Fukushima, K.: Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* **36**, 193–202 (1980)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997)
5. Jain, A., Koppula, H.S., Raghavan, B., Soh, S., Saxena, A.: Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In: *ICRA*, pp. 3118–3125 (2016)
6. Kamata, M., Fujita, M., Shino, M., Nagai, M., Michitsuji, Y., Maeda, K.: Research on incident analysis using drive recorder part 1: toward database construction. In: *FISITA World Congress*, pp. 22–27 (2006)
7. Kondrak, G.: *N*-gram similarity and distance. In: Consens, M., Navarro, G. (eds.) *SPIRE 2005*. LNCS, vol. 3772, pp. 115–126. Springer, Heidelberg (2005). https://doi.org/10.1007/11575832_13
8. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In: *INTERSPEECH*, pp. 1045–1048 (2010)
9. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: *ICML*, pp. 807–814 (2010)
10. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: *CVPR*, pp. 6517–6525 (2017)
11. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**(3), 211–252 (2015). <https://doi.org/10.1007/s11263-015-0816-y>
12. Sharma, S., Kiros, R., Salakhutdinov, R.: Action recognition using visual attention. In: *ICLR Workshop* (2016)
13. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: *NIPS*, pp. 568–576 (2014)
14. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *JMLR* **15**, 1929–1958 (2014)
15. Suzuki, T., Aoki, Y., Kataoka, H.: Pedestrian near-miss analysis on vehicle-mounted driving recorders. In: *MVA*, pp. 416–419 (2017)
16. Szegedy, C., et al.: Going deeper with convolutions. In: *CVPR*, pp. 1–9 (2015)
17. Tada, M., Noma, H., Utsumi, A., Segawa, M., Okada, M., Renge, K.: Elderly driver retraining using automatic evaluation system of safe driving skill. *IET Intell. Transp. Syst.* **8**(6), 266–272 (2014)
18. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: *ICCV*, pp. 3551–3558 (2013)
19. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A.J., Hovy, E.H.: Hierarchical attention networks for document classification. In: *HLT-NAACL* (2016)
20. Yokoyama, D., Toyoda, M., Kitsuregawa, M.: Understanding drivers' safety by fusing large scale vehicle recorder dataset and heterogeneous circumstantial data. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) *PAKDD 2017*. LNCS (LNAI), vol. 10235, pp. 734–746. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57529-2_57
21. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: a 10 million image database for scene recognition. In: *IEEE TPAMI*, pp. 1452–1464 (2017)