Denoising Recurrent Neural Networks for Classifying Crash-Related Events

Sungjoon Park, *Member, IEEE*, Yeon Seonwoo, *Member, IEEE*, Jiseon Kim, *Member, IEEE*, Jooyeon Kim, *Member, IEEE*, and Alice Oh, *Member, IEEE*

***Abstract*— With detailed sensor and visual data from automobiles, a data-driven model can learn to classify crash-related events during a drive. We propose a neural network model accepting time-series vehicle sensor data and forward-facing videos as input for learning classification of crash-related events and varying types of such events. To elaborate, a novel recurrent neural network structure is introduced, namely, denoising gated recurrent unit with decay, in order to deal with time-series automobile sensor data with missing value and noises. Our model detects crash and near-crash events based on a large set of time-series data collected from naturalistic driving behavior. Furthermore, the model classifies those events involving pedestrians, a vehicle in front, or a vehicle on either side. The effectiveness of our model is evaluated with more than two thousand 30-s clips from naturalistic driving behavior data. The results show that the model, including sensory encoder with denoising gated recurrent unit with decay, visual encoder, and attention mechanism, outperforms gated recurrent unit with decay, gated CNN, and other baselines not only in event classification and but also in event-type classification.**

***Index Terms*— Recurrent neural networks, missing data imputation, denoising sensor inputs, driving events.**

I. Introduction

ROAD traffic crash is still one of the main cause for death, injury and disability. World Health Organization predicts that road traffic injuries will be the fifth leading cause of death by 2030 [1]. In this context, one way to reduce death from such tragedy is to report an accident as soon as possible. If there is a real-time automatic collision detection algorithm, this could be linked to an automatic reporting system to call the police and emergency rescue in urgent situation to save valuable human lives.

Most automobiles are equipped with sensors capturing various states including velocity, longitude, latitude, accelerator and brake pedal states. If these sensor measurements are to be recorded with timestamp in real-time, the collected data could be a good source to detect critical and potentially dangerous driving events. Recent developments in machine learning can help analyzing these time-series sensor recordings. We present a novel Recurrent Neural Network (RNN) model to detect dangerous driving events from automobile sensor data and front-facing video data.

Previous approaches for detecting prominent events from driving data include smartphone sensor data with a thresholding classifier [2] or decision trees [3]. Other approach uses visual data collected from surveillance video cameras [4]. However, there has been little research of detecting dangerous driving events from a combination of time-series data from multiple sensors installed in cars and videos from in-vehicle forward-facing cameras. In this paper, a neural network architecture which leverages both sources of time-series data to classify crash-related events is proposed.

Recent advances in neural network models have led a wide application of the RNN for modeling sequential data [5]–[10]. RNN has been applied in the field of driving behavior studies for detection of driver confusion status with Long Short Term Memory (LSTM), a variant of the RNN [11], and prediction of driver action with the bidirectional recurrent neural network [12]. However, RNN and its variants including LSTM [13] and the Gated Recurrent Unit (GRU) [14] do not consider missing values nor intrinsic noise in the data which are unavoidable in data from sensors in noisy, naturalistic driving situations. Thus we design a novel variant of the recurrent neural network cell to handle such data characteristics. In previous research in the health care domain, the Gated Recurrent Unit with Decay (GRUD) was proposed to handle missing data [15]. We improve upon the GRUD by integrating the denoising process into the model to construct the Denoising Gated Recurrent Unit with Decay (DGRUD).

In this paper, our model, DGRUD, is shown to be effective for classifying driving events with evaluation from naturalistic driving data (details of the data are described in Section IV). To show the improved accuracy of classification, we compare our model with basic GRU, GRUD and other comparison models in evaluating the accuracy of types of crash classification and near-crash involving a front vehicle, a side vehicle, or a pedestrian. DGRUD outperforms both RNN (GRU and GRUD) and CNN based models in classification tasks.

Our main contributions are as follows:

- Proposing a neural network architecture that accepts input of time-series vehicle sensor data and visual features collected from forward-facing videos for learning to classify crash-related events and their types.
- Developing a novel recurrent neural network model to effectively cope with noisy time-series data with missing

Manuscript received December 22, 2017; revised July 6, 2018, December 17, 2018, and April 29, 2019; accepted May 17, 2019. Date of publication June 18, 2019; date of current version June 29, 2020. This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government [MSIP; Ministry of Science, ICT (Information and Communication Technology) and Future Planning] under Grant 2016R1A2B4016048 and in part by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Korean Government [MSIP; Ministry of Science, ICT (Information and Communication Technology) and Future Planning] under Grant 2017M3C4A7065962. The Associate Editor for this paper was D. Fernandez-Llorca. *(Corresponding author: Alice Oh.)*

The authors are with the School of Computing, Korea Advanced Institute of Science and Technology, Daejeon 305-701, South Korea (e-mail: sungjoon.park@kaist.ac.kr; yeon.seonwoo@kaist.ac.kr; jiseon_kim@kaist.ac.kr; jooyeon.kim@kaist.ac.kr; alice.oh@kaist.edu).

Digital Object Identifier 10.1109/TITS.2019.2921722

values, in order to build an end-to-end model that reduces heavy pre-processing steps.

Our paper is organized as follows. Section II describes previous work on learning driving events and data pre-processing techniques of missing data imputation and denoising method. Section III describes the overall design of our neural network model, and introduces a novel RNN cell structure for time-series data. Section IV reports the classification performance of our model. Section V analyzes the evaluation results based on the outcomes of experiments. Finally, Section VI presents future directions of this research.

II. Related Work

In this section, previous work in two related categories is described. One is discussion on research of learning driving events and the other is description for methods of missing data imputation and denoising techniques.

1) Driving Event Detection: Previous research deal the detection driving events with machine learning algorithms based on various data sources such as video, audio, and event relevant features.

Vehicle forward-facing video is a good source for detecting driving events. Lane changes can be captured by a visual rhythm method which records the pixel data along the scan line in every video frame to capture the temporal information change [16]. Furthermore, the overall driving environment can be detected as well through a neural network based sensory, perceptual, and conceptual analyzer [17]. Surveillance video can be another data source of event detection. Abnormal vehicle events can be automatically captured by a semi-supervised mixture of a Gaussian Hidden Markov Model [18], or unsupervised learning method [19]. Furthermore, an abnormal event such as speed violation can be detected in real-time monitoring of those videos [20].

Meanwhile, driver features, vehicle characteristics, and environmental features such as time, weather, light and surface conditions are used to predict driving events in intersections with multilayer perceptrons (MLP) [21]. Also, driving patterns in highways through similar features including road, vehicle, weather condition, and traffic attributes through artificial neural networks [22]. Some studies focus on detecting audio events in public transport vehicles by using Gaussian mixture models and support vector machines (SVM) [23].

Since this paper is focusing on the automatic detection of crash-related events, vehicle collision detection is reviewed in detail. Several previous approaches applied machine learning algorithms and data mining techniques to detect collisions [24].

MLP is a basic form of the neural network which is frequently used to classify road accident patterns based on relevant features including driver attributes (gender, age, location, and region) [25] or traffic-related features [26]. Decision trees are another widely used method for the classification problem. Accident, road, and environmental features are used as inputs for these tree algorithms to predict collisions [27]–[29]. Other classification algorithms such as K-nearest neighbors (KNN) are also used to classify accidents [30], [31]. On the other hand, unsupervised techniques such as k-means clustering algorithm and autoregressive models (ARM) are used to discover an association rule between input variables and accidents as outcomes [32].

In brief, most previous work in collision detection used relevant features of the events including vehicle, road, driver, and other environmental features as training data. This class of approaches is not easily applicable to immediate collision detection because these features can only be collected after determining the event is a collision. To detect collisions in real-time, the detection algorithm should rely on time-series naturalistic driving data including sensors and video streams collected in real-time.

Recently, a large dataset of time-series naturalistic driving data allowed researchers to develop algorithms that can be employed in a real-time classification of driving events or collisions. Time-series vehicle sensor data and visual features are used to find what visual-cognitive functional factors of elderly drivers affect crash events by employing a Poisson regression model [33].

Specifically, smartphone sensors are used as an alternative to vehicle sensors to collect real-world driving data. A rule-based pattern matching method is applied to those data for detecting the events [2]. Furthermore, MLP is employed to detect driving events from time-series sensor data. The 'attribute vector' is constructed from the raw sensor inputs to contain statistics of the inputs, and MLP is applied over the vector and trained to classify the events, which outperforms other baseline models [34].

Also, video streams are used for event detection. To build an end-to-end model for detection, pre-trained Convolutional Neural Networks (CNN) applied with RNN was used. The network showed improved performance over the traditional methods including steps of (1) extraction, (2) representation, and (3) classification [35].

Compared to these studies, our model integrates these previous approaches of capturing events from time-series sensor data and video streams for collision detection and build an end-to-end model that accepts both types of data.

2) Missing Data Imputation: Missing data imputation has been widely applied to various application fields: predicting air quality [36], climate [37], wireless sensor data [38], cancer [39], and satellite images [40]. When missing data is observed, one of the most straightforward approaches is an available-case analysis, which is discarding data with missing values [41]. However, this may result in leaving biased samples if there are systematic patterns in missing values, so filling in missing data is preferred.

The simplest way to impute missing values is to replace them with the last observed values or to constant values such as means. However, this approach might cause a problem in case there is more than a trivial fraction of data missing, where one can use regression models to predict missing values based on the other available features [42]. A more complex approach uses machine learning techniques. Sequential regression trees for implementing multiple imputations via chained equations produces more plausible imputations, and hence more reliable

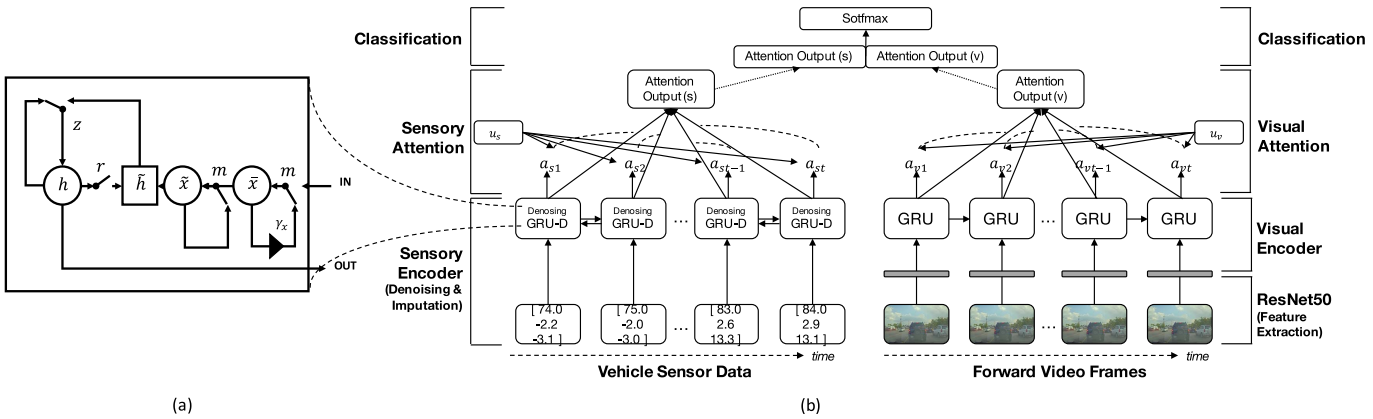


Fig. 1. Proposed neural network architecture (b) and our proposed recurrent neural network cell structure (a), DGRUD. Our neural network consists of two parts: 1) processing time-series vehicle sensor measurements and 2) processing video frames. Two parts are combined at the top of the attention layer, and then a softmax layer is added to classify driving events. Our cell has a denoising mechanism that smooths time-series input by using a weighted mean filter scheme. The weights for each time step are jointly trained with the networks.

inferences than the standard sequential regression imputation techniques [43]. Another approach is handling missing value by SVM. This SVM regression based algorithm for filling in missing data predicts the input attribute values [44]. Also, there is a non-parametric perspective considering the uncertainty of the predicted outputs when missing values are contained [45]. As for neural networks, an ensemble of the networks for incomplete data classification is proposed [46]. The dataset containing missing value is divided into a group of complete sub-datasets for training the neural network. The proposed model can handle all the information with missing values and maintain maximum consistency of incomplete data compared with other models in classification. These approaches mainly focus on imputing missing values precisely, regardless of the task or machine learning model which should learn over the dataset with missing values.

Recently, imputation mechanism is integrated into GRU-based recurrent neural networks and performs better over comparison models which do not have the mechanism in it [15]. Based on the previous work, our model integrates missing value imputation to downstream tasks as well, in order to jointly learn the patterns for the tasks as well as how to replace missing values. This enables us to build an end-to-end model which reduces the pre-processing step of imputation.

3) Denoising Techniques: Image denoising techniques can be divided into a spatial domain (linear or non-linear filters) and transform domain (data adaptive or non-data adaptive) approach [47]. In signal processing, moving averaging, wavelet transform, fuzzy logic, and various methods are developed [48], [49]. For denoising vehicle sensor signals, wavelet transformation is used [50], [51].

As with missing data imputation techniques, denoising methods also concentrate on removing noise itself, not considering downstream tasks. Our approach combines dealing with missing and noisy data with the prediction task using DGRUD. For the denoising method, different from previous approaches, our model employs weighted mean filter smoothing over the time domain because it is simple yet effective for joint training with neural networks. Integration of denoising and missing data imputation allows reducing complicated pre-processing steps and improving the performance of downstream tasks since the parameters in the model are jointly optimized for the tasks.

III. Model

In this section, our neural network architecture for event classification is illustrated. Next, our novel RNN structure is introduced, which is effective for learning useful information from time-series vehicle sensor data.

A. Model Overview

The model consists of two parts as shown in Fig. 1(b). The left part of the subfigure (b) starts from the sensory encoder which accepts time-series vehicle sensor data with missing values. Specifically, Fig 1(a) shows the sensory encoder structure of the overall model (DGRUD) which imputes missing values and denoises inputs. On the other hand, the right part has a visual encoder that processes visual features of the forward-facing videos. Each part has an attention layer that improves performance of encoders as well as detection for part of the video or time-series data relevant to certain driving event. Finally, attention layer's outputs are concatenated, and used as inputs to the last softmax layer to compute the probability for classification.

1) Processing Sensor Data: The left part of the model in Fig. 1 (b) takes time-series vehicle sensor data as input features. Since the data contains missing values, sensory encoder should be able to deal with the missing values. The proposed DGRUD allows imputing missing values to smooth away noise in the sensor data. Our model, DGRUD, is appropriate for learning useful information from time-series naturalistic driving data. Details are introduced in the next subsection.

2) Processing Visual Features: Next, as shown in the right half of Fig. 1 (b), visual features are also used to detect driving events because some events require information collected

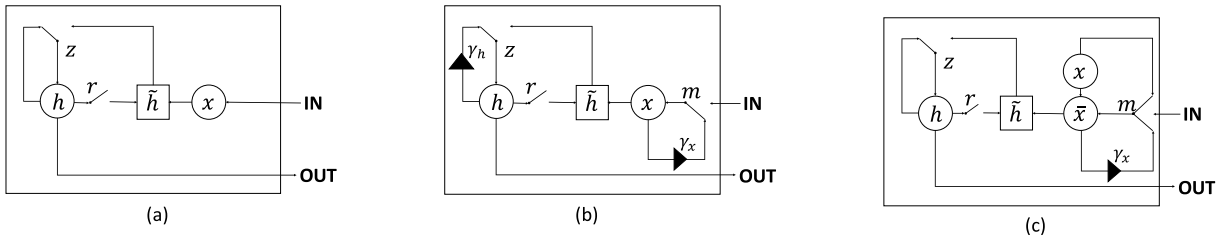


Fig. 2. Visualization of (a) GRU, (b) GRUD, (c) DGRUD. GRU has two gating mechanisms to store information while processing sequential inputs. GRUD is an extended version of GRU, capable of imputing missing values in input with decay. Our model, DGRUD, has denoising filter with trainable weights.

from the forward-facing video camera. Every image of the video frames is compressed by the pre-trained ResNet50 [52]. By feeding the re-sized frames to the network, each input frame is compressed to a 2,048 dimensional feature vector. Each feature vector is then used as an input to a vanilla GRU cell [14].

3) Attention Layer: Attention mechanism is widely used with recurrent neural networks. In general, this allows models to focus more on a specific part of the sequence which helps the model to better learn the objective. This mechanism not only enhances the performance of the model, but also provides cues of network's prediction [53]–[55]. Then, attention mechanism [55] is added which computes the attention weights for every input time step as in Eq. 2-3:

$$u_t = tanh(W_s h_t + b_s) \tag{1}$$

$$a_t = \frac{exp(u_t' c_s)}{\Sigma_t exp(u_t' c_s)} \tag{2}$$

$$v = \Sigma_t a_t h_t \tag{3}$$

The hidden state for every time step is projected into u_t (Eq. 2), then the degree of focusing on the timestep across the sequence is measured by multiplying context vector c_s and u_t, and it is normalized through the softmax function (Eq. 3). These normalized weights are known as attention weights. The weights are applied to the input sequence of this layer, h_t, to compute the output of the layer v as a weighted sum of the sequence (Eq. 3). The layer is attached to the sensory encoder and the visual encoder.

4) Softmax Layer: A softmax layer is stacked to classify events based on the concatenated outputs of each attention layer v_{sv}.

$$p = \frac{exp(W_c v_{sv} + b_c)}{\Sigma exp(W_c v_{sv} + b_c)} \tag{4}$$

W_c and b_c are trainable parameters of the softmax layer, and p includes probabilities of belonging to each class. To train the model, negative log-likelihood of the correct labels is minimized.

B. Denoising Gated Recurrent Unit With Decay (DGRUD)

We propose a novel RNN structure, Denoising GRUD (DGRUD), which is appropriate for learning useful information from the time-series sensor data containing noisy features. In addition, the cell handles missing values that occur in cases of malfunctioning of the sensor or systematic missing during the data collection. We introduce our model by highlighting the difference from the baseline models. As shown in Fig. 2, DGRUD first replaces missing values by gate m, then smooths inputs by computing the weighted mean, $\bar{x}$. This processed input is fed as input to GRU operations.

1) Gated Recurrent Unit With Decay (GRU): GRU is a widely used RNN structure [14] that is simpler than LSTM cell [13] and yet shows comparable performance. GRU has two gating mechanisms:

$$z_t = sigm(W_z x_t + U_z h_{t-1} + b_z) \tag{5}$$

$$r_t = sigm(W_r x_t + U_r h_{t-1} + b_r) \tag{6}$$

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \tag{7}$$

The reset gate r_t decides the extent of previous hidden state $h_{t\text{-}1}$ during computation of the candidate state $\tilde{h}_t$. The update gate z_t controls the ratio between the previous hidden state and the candidate state to determine the current state h_t.

2) Gated Recurrent Unit With Decay (GRUD): GRUD [15] imputes missing values while the parameters of the cells are trained. Decay rates are added to the vanilla GRU that reflects inputs reliable only for a while. The rate is defined as follows:

$$\gamma_t = exp(-max(0, W_\gamma \delta_t + b_\gamma)) \tag{8}$$

where W_γ and b_γ are trainable parameters and δ_t is the amount of time passed from the most recent value observed for each features. Based on the rate, the cell imputes missing values as below:

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d)\gamma_{x_t}^d x_{t'}^d + (1 - m_t^d)(1 - \gamma_{x_t}^d)\bar{x}^d \tag{9}$$

where the gate m_t^d controls whether the value is imputed through the cell, containing binary values which indicates whether the input feature d is observed at current timestep t. $\gamma_{x_t}^d$ denotes the decay rate of feature d in input x at timestep t. This regulates the ratio between 1) the last observed value $x_{t'}^d$ and 2) the global mean over the entire sequence of input features $\bar{x}^d$. The decay rate is applied to the inputs and to the hidden states as well. The other part of the cell is the same as vanilla GRU. The parameters are jointly trained with the other GRU parameters. Intuitively, when the elapsed time of the input missing is increasing, the last observed value should be decayed toward to the global mean as a default setting. Assuming the global mean as *default* is reasonable since the cell is designed to learn tasks for health care domain [15].

3) Denoising Gated Recurrent Unit With Decay (DGRUD): In order to effectively deal with the time-series vehicle sensor data with missing values *and* noise, a trainable denoising filter is added to GRUD. The main difference between GRUD and DGRUD is that DGRUD has denoising function which adapts to the downstream task. The noise can be removed by applying the mean filter, which is widely used as a linear filtering scheme [56]. Our model relies on the scheme. The reason choosing such a simple approach is to 1) keep the complexity of the cell as simple as possible, and 2) make the gradients easily computed through backpropagation.

In this cell, $\bar{x}_t^d$ is computed, which is *smoothed* version of x_t^d and $x_{t'}^d$ in GRUD, by applying weighted mean filter scheme as follows:

$$x_t^d \leftarrow m_t^d x_t^d + (1 - m_t^d) x_{t'}^d \tag{10}$$

$$\bar{x}_t^d = \frac{1}{k} \Sigma_{i=t-k}^{t} w_i^d x_i^d \tag{11}$$

where k is a hyperparameter that indicates how many timesteps should be considered to evaluate the mean. To clarify the difference between $\bar{x}_d$ in GRUD and $\bar{x}_t^d$ in our cell, $\bar{x}_d$ is global mean of the feature d over full sequence always resulting in the same value regardless of time t. On the other hand, $\bar{x}_t^d$ depends on time t because it is moving (weighted) average with window size k from $t - k$ to t. First, assuming k is given, missing values are imputed in time t with the most recent observations as Eq. 10. Note that the weight w_i^d can vary by feature d as well as a timestep t. Then, a smoothed input through weighted average $\tilde{x}_t^d$ is computed as Eq. 11. Followings are applying decay rates as in GRUD with smoothed inputs:

$$\tilde{x}_t^d \leftarrow m_t^d \bar{x}_t^d + (1 - m_t^d) \gamma_{x_t}^d \bar{x}_t^d \tag{12}$$

Assuming that the default state of the vehicle is *zero-state* which means a vehicle will eventually stop running. Based on the assumption, $\bar{x}_d$ is replaced to a zero vector so that the last term in Eq. 9 disappears. The other part of the cell is the same as GRUD, applying GRU operation with $\tilde{x}_t^d$. Note that the filter weights w with size k, indicating weights for each input in timestep t are jointly trained while learning from downstream tasks.

In summary, DGRUD consists of three steps: (1) denoising inputs by using trainable weighted mean filters, (2) imputing missing data with exponential decay, and (3) performing GRU operations. The step (1) alleviates noise in inputs through computing weighted mean of the inputs with trainable weights. Then the step (2) replaces missing values to the last smoothed observed value. The value will be decayed toward to zero if the missing value continues. Lastly, (3) GRU operation is performed after the missing values are replaced and the smoothing process is done. Fig. 1(a) depicts this process: raw inputs pass gate m to smooth raw inputs, and smoothed input $\bar{x}$ is passing the gate m again to impute missing values, and the imputed value will decay with the rate γ_x. Then the imputed input $\tilde{x}$ will be used to perform GRU operations. Note that another difference from the GRUD is that the decay for hidden state of GRU h is removed.

Lastly, DGRUD is applied bi-directionally in the overall model, which means one DGRUD scans sensor inputs forward and another DGRUD scans the inputs in backward. Each DGRUD generate hidden state h and they are concatenated to deliver information of time t to the attention layer stacked above.

IV. Empirical Experiments

A. Dataset

We obtain time-series sensor data and forward videos of driving events in SHRP2 (The second Strategic Highway Research Program) NDS (Naturalistic Driving Study) dataset. The database has information of drivers (demographic background, physical, psychological, and medical condition, etc.) vehicles (types, condition, etc.) that consented to participate in the research program. Qualified researchers can access to the participants' trips (summary, time-series records) and events records with personally identifiable information removed by the database.

The dataset includes 9,574 forward videos including crash-related events and normal driving situations aligned with time-series vehicle sensor data collected in 2014. Data contains various driving conditions (day/night, sunny/cloudy/rainy, etc.). Each data are a trip segment of about 30 seconds. The events are collected from about 1,500 drivers. Type of the sensors includes vehicle speed, acceleration in x/y-direction, gyroscope in z-direction, acceleration pedal position, steering wheel position, and GPS measurements, radar measurements recorded for every 100ms. Forward videos are stack of images recorded 14 frames per second. In addition, meta-data of events (event type, event type, narrative of events, etc.) is available. This meta-data is used as labels for supervised learning.

Note that our data contain significant amount of missing values. The proportion of the timesteps in a data which contains at least one missing observation of a sensor is 10.9% in event classification task, and 36.9% for event type classification task.

B. Tasks

Two classification tasks are performed: 1) Event classification and 2) Event type classification to learn driving events and their nature. Evaluation of performance of our neural network architecture and DGRUD is conducted through these two tasks.

1) Event Classification: The first task is classifying whether there exists an event (crash/near-crash) or not (no conflict) at any timestep in given sequential sensor and visual data. The task includes 3 classes as follows:

- *Crash:* includes explicit impact with other vehicles or obstacles. Since the moment of the accident is included in the middle of the video and sensor records, they usually fluctuate and the vehicle stops after the accident.
- *Near-crash:* includes sudden stop or turn to avoid conflict with other vehicles or obstacles. Similar to the *Crash*, but vehicle immediately recovers to normal driving state after the event.
- *No crash:* includes normal driving situations. There are various normal driving situations, such as going straight,

turning on intersection, highway driving, and waiting at a red light.

2) Event Type Classification: The second task is classifying events by their type. Like the first task, the model should detect the event type at any timestep in the given sequential sensor and visual data. The following are the five classes in the task, and large fluctuations in the observed sensor data is expected except the last *No conflict* class:

- *Conflict with a lead vehicle:* includes crash or near-crash with the vehicle ahead. The other vehicle involved in the event is clearly recorded in the center of the forward video.
- *Conflict with vehicle in adjacent lane:* includes crash or near-crash with a vehicle in the left/right lane. The other vehicle involved in the event is recorded in the right/left side of the forward video, but sometimes the vehicle cannot be observed in the video.
- *Single vehicle conflict:* includes conflicts without any other vehicle, such as crash/near-crash with non-vehicle objects including lane dividers and trees.
- *Conflict with animal/pedestrian:* includes crashes and near-crashes with animals or pedestrians.
- *No conflict:* includes normal driving situations. There are various normal driving situations, such as going straight, turning on intersection, highway driving, and waiting at a red light.

C. Pre-Processing

To train the models for each task, events with data from a set of sensors, such that at least one value is observed in a time-series instance, are included. For event classification, included events have following features: 1) vehicle speed, 2) acceleration in x/y-direction, 3) gyroscope in z-direction, 4) acceleration pedal position, 5) steering wheel position. For event type classification task, events have a smaller number of features, selecting them with the following features: 1) vehicle speed, 2) acceleration in x/y-direction, 3) gyroscope in z-direction. In order to obtain sufficient number of examples for every event types in both tasks, it was inevitable to drop 2 features in the event type classification task which has larger number of classes than that of event classification task. Specifically, there were only a few examples of "Conflict with animal/pedestrian" cases having all 5 features recorded.

Then time-series sensor data and visual features are aligned with respect to the timestep. First, the starting points of the two input sources are matched, then 10 frames are sampled out of 14 frames per second based on the frame-sensor aligning index information in the dataset, since sensor measurements are recorded every 100ms.

Lastly, the visual features from sampled video frames are extracted by using pre-trained neural networks for image recognition. ResNet50 [52] was used to compress a raw image that has approximately 360,000 features to a 2,048 feature vector, referring the vector as visual features. To compute the features, the top softmax layer is removed, and the input image is fed into the network to obtain the vector. By employing this approach, sufficient size of a mini-batch and reasonable training time could be obtained.

After pre-processing, 2,298 trip segments are selected for event classification. Each class consists of 1,766, 176, 356 events, respectively. For event type classification, 1,794 events are retained. Each class consists of 609, 188, 513, 128, 356 events.

D. Experimental Settings

The total number of events are split into train, validation, and test sets with the ratio 6:2:2 while preserving the ratio between classes. Hyperparameters are tuned on validation set, and 5-fold cross validation results are reported. All models are trained until F1 score of the validation set does not increase significantly. To decide the number of hidden units in GRU, GRUD, DGRUD, and attention layer, we try 16, 32, 64, 128 hidden units, and choose the appropriate number of hidden units based on the performance over the validation set, as well as considering the memory limits of GPUs used in experiments. The size of the hidden units of GRU processing visual features is set to 64, and that of GRUD or DGRUD in sensory part to 64 as well. Also, the attention vector size is set to 64. We also explored the size of trainable filter weights in DGRUD, 2, 5, 10, 20, and choose the number to 10. The weights in are initialized to a zero vector.

We compare the classification performance of the following three conditions in terms of data source accepted by each model:

- *Processing Sensor Data* Time-series sensor data are used to classify events. GRUD [15] is used as a baseline model, and compare performance with DGRUD. To the best of our knowledge, GRUD is state-of-the-art among models that integrate missing value imputation in the network. It showed better performance over vanilla recurrent neural networks since it effectively deals with missing values in input times-series data. Also, CNN is frequently used to capture patterns from time-series data, so vanilla CNN and Gated CNN is added with 2 convolution and max pooling layers as a comparison model. Lastly, since the missing values can be easily replaced by constant default values, such as the mean of each features or zero, our model is also compared with this method.
- *Processing Visual Features* Visual features are used to classify events. Since it is unlikely to occur missing values in video frames, i.e., in feature vectors computed from ResNet50 [52], a vanilla GRU is used for visual encoders to process visual features. To the best of our knowledge, applying recurrent neural network above the pre-trained convolutional neural network is state-of-the-art for event detection from video streams [35]. It showed better performance over traditional approaches.
- *Processing both Sensor Data and Visual Features* Models which use both sensor data and visual features to classify events. GRU is used in visual encoder part, and GRUD or DGRUD is used in sensory encoder part.

In addition, attention layers are attached to each model, and performance of models with and without the layers are compared to systematically evaluate the attention layer.

All models are trained by the Adam [57] optimizer. Learning rate is set to.001, and batch size 128. Our neural network

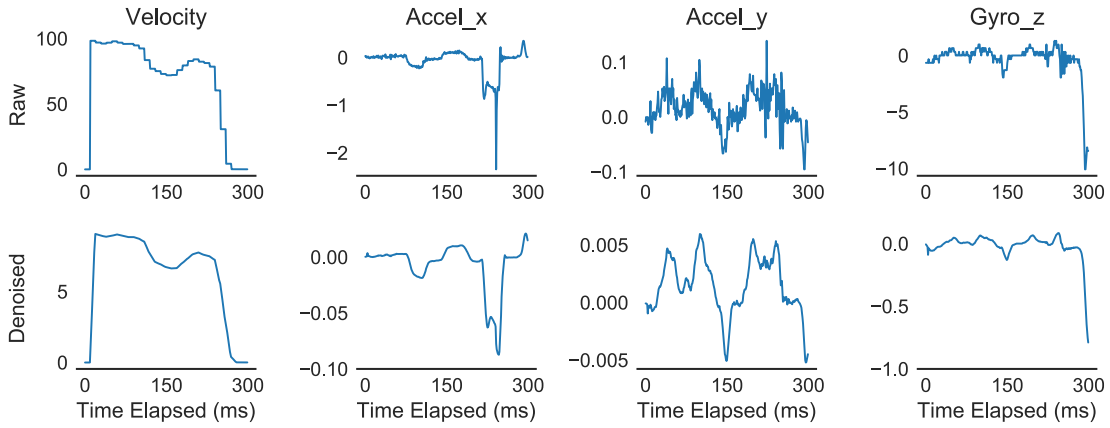


Fig. 3. Four time-series sensor data of conflict with lead vehicle event. These sensor data are used for event type classification task. X-axis indicates time, and Y-axis represents (smoothed) sensor data. Compared to the top row (raw data), bottom row is smoothed through the learned filter.

is implemented through TensorFlow, and we publicly open our implementation of DGRUD on GitHub.[1]

Both precision and recall are considered as an evaluation metric for the performance of models, so F1 score is reported as follows:

$$F1 = \frac{2 \times r \times p}{r + p} \tag{13}$$

where r is recall and p is precision. Overall macro F1 score and the score for each class are reported in detail.

V. Results

A. Event Classification

Event classification results of the models are shown in Table I. Model 1-7 have only sensory encoder which takes time-series sensor inputs. Specifically, Model 2, 3, 6, and 7 are capable of handling missing data (GRUD, DGRUD) whereas Model 1, 4, and 5 are not, so that missing data is replaced by a constant value. GRUD shows higher macro F1 score over the vanilla GRU (+.006), and it is improved when an attention layer is attached (+.039). Our model, DGRUD, outperforms GRUD with attention (+.128) that shows better performance compared to the GRUD, CNN, and gated CNN [58].

Model 8 and 9 accept only visual features to classify events through the GRU. As in the sensory encoders, these visual encoders show better performance with attention (+.037), which is similar to that of gated CNN. Note that DGRUD with attention outperforms visual encoders (+.080).

Overall, performance enhanced when the models (Model 10-13) have both sensory encoder and visual encoder. Model 10-11 use GRUD as sensory encoder, and the others use our model. Model 13 shows the highest F1 score (.799) across the competing models. The tendency that DGRUD outperforms GRUD is consistent when visual features are added as inputs to the neural networks. Except the *Crash* class, F1 score is highest on the *No Crash* (.808) and *Near-Crash* (.603) classes, resulting in higher macro F1 score of Model 13. In addition, the attention layer is helpful in improving the classification performance.

[1]https://github.com/SungjoonPark/DenoisingRNN

In general, *Crash* events are most accurately classified, with *No Conflict* and *Near-Crash* following because a crash event is obvious for the most part, but *Near-Crash* and *No Conflict* events are indistinguishable in some cases. As shown in Fig. 4, *Near-Crash* events could be predicted as *Crash* or *No Conflict* events.

B. Event Type Classification

Next, The results of event type classification are presented in Table II. The pattern of higher f1 scores when GRUD is replaced with DGRUD is similar to the event classification task. Comparing sensory encoder models, GRUD shows better F1 scores than GRU (+.042), and even better with attention (+.023). DGRUD with attention outperforms GRUD with attention (+.034). Also, the gated CNN and the visual encoders record similar level of macro F1 score of GRUD.

Models that accept sensor data and visual data show better scores in event classification task. Also, DGRUD outperforms GRUD without attention (+.084) and with attention (+.070). The F1 score of the dual encoder model (Model 13) which uses DGRUD as the sensory encoder achieves the highest macro F1 score (.621) as well as for the classes of *Conflict with Animal/Pedestrians* (.382). In case of the other classes, the dual encoder using DGRUD without attention shows the highest F1 scores.

Among the five classes, *No conflict* is most accurately classified, followed by *Single Vehicle Conflict* and *Conflict with Lead Vehicle* with a high accuracy as well. The other classes do not show high F1 scores, partially due to the small size of the training data. Note that the F1 score of *Conflict with Animal/Pedestrian* is highly improved with visual encoders since it is very difficult to classify correctly without visual data. *Conflict with vehicle in adjacent lane* is even harder to classify since it might not be captured by the *forward* videos, and the time-series sensor data can be difficult to distinguish from *Single Vehicle Conflict* and *Conflict with Lead Vehicle* classes.

Specifically, as shown in Fig. 4, about half of the *Conflict with vehicle in adjacent lane* events are classified by the model as *Conflict with Lead Vehicle*. Also, *Conflict with*

TABLE I
F1 SCORES FOR EACH CLASS OF EVENT CLASSIFICATION

Model				F1 scores			
No.	Sensory Encoder	Visual Encoder	Attention Layer	Crash	Near-Crash	No Crash	Overall (macro)
1	GRU	-	X	0.9165	0.2899	0.5659	0.5908
2	GRUD	-	X	0.9182	0.3181	0.5520	0.5961
3	GRUD	-	O	0.9336	0.3700	0.6001	0.6346
4	CNN	-	X	**0.9972**	0.2400	0.7560	0.6644
5	Gated CNN	-	X	0.9949	0.2857	0.7697	0.6835
6	DGRUD	-	X	0.9914	0.4545	0.7895	0.7442
7	DGRUD	-	O	0.9768	0.5306	0.7796	0.7623
8	-	GRU	X	0.9858	0.2371	0.7128	0.6452
9	-	GRU	O	0.9916	0.3863	0.6700	0.6826
10	GRUD	GRU	X	0.9579	0.4325	0.6633	0.6846
11	GRUD	GRU	O	0.9949	0.5390	0.7632	0.7657
12	DGRUD	GRU	X	0.9915	0.5714	0.7445	0.7692
13	DGRUD	GRU	O	0.9860	**0.6026**	**0.8080**	**0.7988**

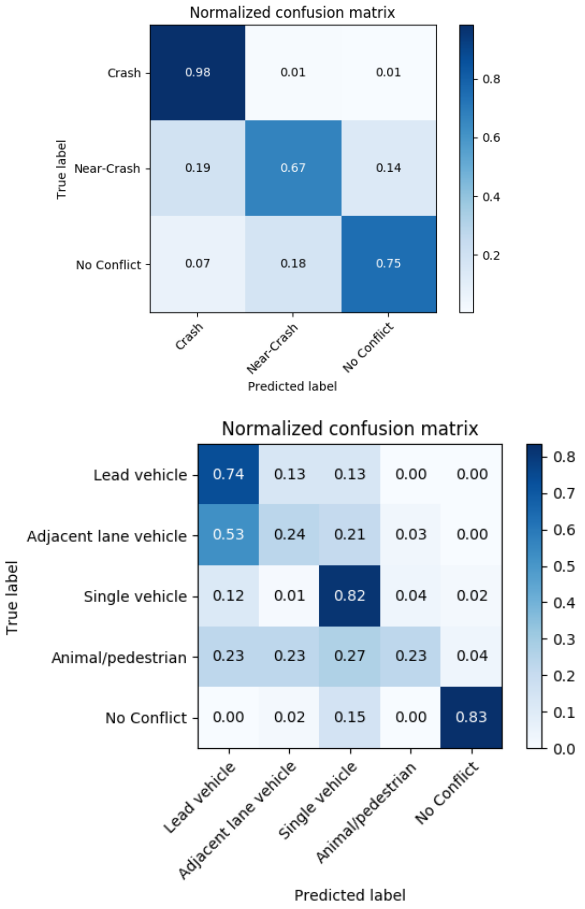


Fig. 4. Confusion matrix of DGRUD (Sensory encoder) and GRU (Visual encoder) for the event classification task (top) and the event type classification task (bottom).

Animal/Pedestrian events are predicted to be other conflict events.

C. Denoising Effects

DGRUD effectively learns a linear filter to smooth away the noise in time-series input. To show the effect, the amount of noise in sensor data is empirically estimated through beta-sigma procedure [59], since the time-series data was provided by SHRP2 without information of the vehicle sensors. The beta-sigma procedure analyze the distribution of derivatives, to look at contribution of noise terms to the value of them. The procedure assumes Gaussian noise and the degree of the noise is represented as a standard deviation of the distribution. In order to compute the effect of noise reduction in our model, estimated the noise of the raw sensor input (Eq. 10), i.e., denoising layer input is compared to the noise of the output of the denoising layer (Eq. 11).

In event classification task, estimated amount of noise in 6 raw sensor data on average was 0.2212. After training the weighted mean filter, estimated noise was reduced to 0.0594. In case of the event type classification task, the estimated noise of 4 raw sensor signal was 0.1587, which was reduced to 0.0447. Thus we observed our model reducing 73.1%, 71.8% of the noise in the raw data in each task, respectively.

Fig. 3 illustrates an example of the smoothed inputs of four time-series sensor features in the event of the conflict with the lead vehicle. These inputs are extracted during the event type classification task. For each subfigure, X-axis indicates time and Y-axis represents each sensor values. Compared to the raw data in the top row, bottom row shows smoothed data by the trained denoising layer (Eq. 11).

D. Attention Mechanism

As shown in Table I and II, attention mechanism helps to learn events and event types more precisely. In general, the overall macro F1 score is increased when the attention layer is attached. This tendency is shown not only in the sensory encoder or the visual encoder but also in dual encoders.

Also, Fig. 5 presents exemplar sensory attention outputs trained on the event classification task, to explore the role of attention layer qualitatively. In the top half of the figure, the graph represents near-crash events when the car suddenly stops to avoid collision with the vehicle ahead stopping at the intersection. Then the driver succeeds in avoiding a conflict, turning right after the lead vehicle starts moving. In this case, the attention weights slightly increase when the near-crash event starts, and then the level of attention is maintained.

TABLE II
F1 SCORES FOR EACH CLASS OF EVENT TYPE CLASSIFICATION

Model				F1 scores					
No.	Sensory Encoder	Visual Encoder	Attention Layer	Lead Vehicle	Adjacent Lane	Single Vehicle	Animal/ Pedestrian	No Conflict	Overall (macro)
1	GRU	-	X	0.6084	0.2540	0.6991	0.1316	0.4937	0.4373
2	GRUD	-	X	0.6429	0.2504	0.6889	0.1279	0.6848	0.4790
3	GRUD	-	O	0.6588	0.2681	0.6881	0.1669	0.7257	0.5015
4	CNN	-	X	0.6667	0.1212	0.6952	0.1429	0.6800	0.4612
5	Gated CNN	-	X	0.6259	0.2742	0.7045	0.068	0.9128	0.5172
6	DGRUD	-	X	0.6667	0.3023	0.7397	0.2500	0.7238	0.5365
7	DGRUD	-	O	0.6560	0.3103	0.7381	0.1771	0.7936	0.5350
8	-	GRU	X	0.5566	0.2203	0.5167	0.2149	0.6476	0.4312
9	-	GRU	O	0.5852	0.1326	0.4968	0.2466	0.7193	0.4876
10	GRUD	GRU	X	0.6316	0.3147	0.6888	0.3101	0.7151	0.5320
11	GRUD	GRU	O	0.6600	0.2947	0.6826	0.3082	0.8135	0.5518
12	DGRUD	GRU	X	**0.7461**	**0.3768**	**0.7692**	0.2703	**0.9189**	0.6163
13	DGRUD	GRU	O	0.7358	0.3681	0.7537	**0.3819**	0.8676	**0.6214**

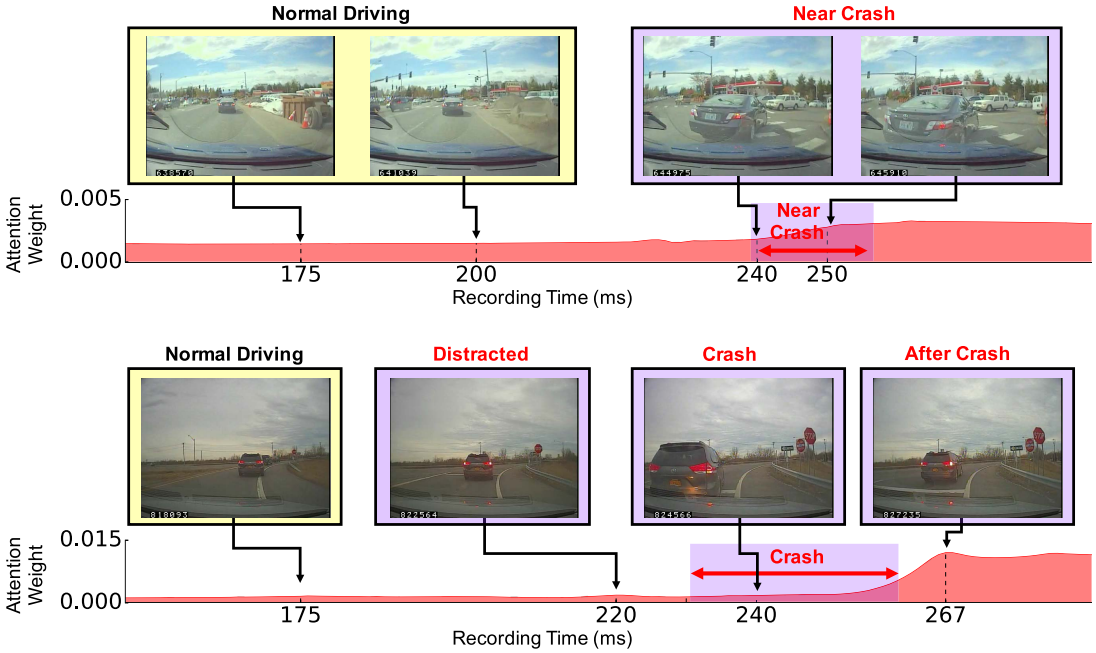


Fig. 5. Examples of sensory attention weights of near-crash event (top) and crash event (bottom). The weights show that in both cases, they start to rise as the event occurs, even increasing after the end of the event. This implies that the network monitors not only the immediate impact of the event but also post-event data to classify the events precisely.

On the other hand, the bottom half of figure 5 represents a crash event due to the distraction of the driver in the intersection. Driver fails to recognize that there was a car ahead so the car crashed into the vehicle ahead without reducing velocity. In this case, the attention weights tend to slightly fluctuate at impact, and if deceleration lasts after the crash, attention weights sharply increase. This can be interpreted to mean that crash-involved vehicles are likely to stop to clear up the accident. That is, stopping after the impact is an obvious pattern for the crash event, which helps the classifier to accurately distinguish crash events among near-crash and no crash events. The sudden fluctuation of the sensor values at impact are also observed in cases of near-crash, thus the attention weights might increase sharply after the impact if the vehicle is still decelerating after the event.

Meanwhile, attention weights in visual encoders tend to increase when the driver is getting closer to the vehicle ahead, regardless of occurrence of crash or near-crash events.

VI. DISCUSSION

In this paper, we develop a neural network model to classify driving events and type of the events. The network accepts time-series sensor inputs and visual features as inputs, and these inputs are encoded by DGRUD and GRU respectively. Each attention layer give weights across time domain, and softmax layer is attached on top of that layer to compute probability for each class. Specifically, we propose a novel RNN cell structure, DGRUD, which can impute missing data and smooth noise away in input time-series data as well. The filter weights are trainable parameters that are jointly trained

with the others in the cell which is flexible enough to be optimized in task-specific way. The weights are also viewed as a single set of layers that consists of one convolution filter across time domain and average-pooling layer with stride k over convolutions.

Our results show that by adding small number of parameters of filter weights in DGRUD, driving events and their types can be classified more accurately. Specifically, the model requires (Number of sensors $\times$ Window size k) of additional parameters over GRUD, to reduce noise in the input sensor data. In this study, only 60 (6×10) and 40 (4×10) trainable parameters are added for each of the event classification and the event type classification task, respectively. This means that removing noise helps distinguishing events because appropriate smoothing can prevent no crash events to be classified as crash events due to high level of noise while preserving the peaks generated by sudden change of vehicle status during events. Furthermore, when extracted features from images comprising forward videos are added to the network, additional performance improvement is observed. It is obvious that some events only can be recognized with time-series sensors, and others only with forward visual features.

Our model shows reasonable performance when considering the accessibility of the data in practice because smartphones and black-box video-recorders can easily collect basic sensor data and forward-facing videos. With these devices, our model can be implemented in driver assistance systems to provide automatic emergency calls with more important and detailed information immediately after crash events are detected and save lives. Also, accurate crash detection would provide automatic claim processing. For insurers, detecting crash event accurately can greatly reduce opportunities for fraud claims. Moreover, detecting the type of crash event can provide further information about the event, helping assessment for cause of a crash event. Insurers also can collect and analyze crash data to enable identifying better driving behaviors for novel insurance policies lowering costs.

In fact, precision and recall of a crash detection algorithm are crucial to achieve these positive outcomes. However, despite using sensor and visual inputs simultaneously, F1 scores of our model in *near-crash* detection, *conflict with adjacent lane vehicle*, and *with animal or pedestrian* reach only moderate level of precision and recall. As for future work, there are two ways to improve the performance on these classes. First, improvement is expected if different types of time-series information are to be added to our model, such as backward/left-side/right-side videos or additional sensor signals such as radars measuring distance to the adjacent vehicles. Second, our model could be improved by employing advanced RNN architectures. By adding a denoising layer to Factorized Recurrent Networks [60], the network would memorize long-term dependencies better on video frames and irregularly observed sensor data. Another candidate is full-capacity unitary RNNs [61] which shows better performance over LSTMs. These networks could be trained better by applying auxiliary loss functions [62].

Lastly, our model could assist safety driving when applied to driving related tasks that require time-series sensor data and forward video.

REFERENCES

[1] W. H. O. Violence, I. Prevention, and W. H. Organization, *Global Status Report on Road Safety: Time for Action*. Geneva, Switzerland: World Health Organization, 2009.

[2] C. Saiprasert, T. Pholprasit, and S. Thajchayapong, "Detection of driving events using sensory data on smartphone," *Int. J. Intell. Transp. Syst. Res.*, vol. 15, no. 1, pp. 17–28, Jul. 2015.

[3] N. Kalra and D. Bansal, "Detecting and characterizing driving events using smartphone," in *Proc. Int. Sci. Technol. Congr.*, 2014, pp. 1462–1468.

[4] H. Guan, R. Kasahara, and H. Kasahara, "Traffic light recognition and dangerous driving events detection from surveillance video of vehicle camera," *Electron. Imaging*, vol. 4, pp. 3–10, Jan. 2017.

[5] M. Fathollahi and R. Kasturi, "Autonomous driving challenge: To infer the property of a dynamic object based on its motion pattern," in *Computer Vision–ECCV*. New York, NY, USA: Springer, 2016, pp. 40–46.

[6] J. Zhang *et al.*, "Chemical substance classification using long short-term memory recurrent neural network," in *Proc. IEEE 17th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2017, pp. 1994–1997.

[7] E. Choi, A. Schuetz, W. F. Stewart, and J. Sun, "Using recurrent neural network models for early detection of heart failure onset," *J. Amer. Med. Inform. Assoc.*, vol. 24, no. 2, pp. 361–370, Aug. 2016.

[8] H. Hayashi, K. Shima, T. Shibanoki, Y. Kurita, and T. Tsuji, "Bioelectric signal classification using a recurrent probabilistic neural network with time-series discriminant component analysis," in *Proc. 35th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2013, pp. 5394–5397.

[9] J. Wang and J. Wang, "Forecasting energy market indices with recurrent neural networks: Case study of crude oil price fluctuations," *Energy*, vol. 102, pp. 365–374, May 2016.

[10] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at Uber," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1–5.

[11] C. Hori *et al.*, "Driver confusion status detection using recurrent neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2016, pp. 1–6.

[12] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of recurrent neural networks for probabilistic modeling of driver behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1289–1298, May 2017.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[15] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Sci. Rep.*, vol. 8, no. 1, p. 6085, 2018. doi: 10.1038/s41598-018-24271-9.

[16] C.-H. Yeh, J.-C. Bai, S.-C. Wang, P.-Y. Sung, R.-N. Yeh, and M. Shih, "Vision-based vehicle event detection through visual rhythm analysis," in *Proc. IEEE Int. Conf. Multimedia Expo*, Apr. 2008, pp. 309–312.

[17] C.-Y. Fang, S.-W. Chen, and C.-S. Fuh, "Automatic change detection of driving environments in a vision-based driver assistance system," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 646–657, May 2003.

[18] H. Sheng, C. Li, Q. Wei, and Z. Xiong, "Real-time detection of abnormal vehicle events with multi-feature over highway surveillance video," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 550–556.

[19] Q. Chen, Q. Qiu, Q. Wu, M. Bishop, and M. Barnell, "A confabulation model for abnormal vehicle events detection in wide-area traffic monitoring," in *Proc. IEEE Int. Inter-Disciplinary Conf. Cogn. Methods Situation Awareness Decis. Support (CogSIMA)*, Mar. 2014, pp. 216–222.

[20] S. T. Rakkesh, A. R. Weerasinghe, and R. A. C. Ranasinghe, "Simulation of real-time vehicle speed violation detection using complex event processing," in *Proc. IEEE Int. Conf. Inf. Automat. Sustainability (ICIAfS)*, Dec. 2016, pp. 1–6.

[21] D. Akin and B. Akba, "A neural network (NN) model to predict intersection crashes based upon driver, vehicle and roadway surface characteristics," *Sci. Res. Essays*, vol. 5, no. 19, pp. 2837–2847, 2010.

[22] F. R. Moghaddam, S. Afandizadeh, and M. Ziyadi, "Prediction of accident severity using artificial neural networks," *Int. J. Civil Eng.*, vol. 9, no. 1, p. 41, Mar. 2011.

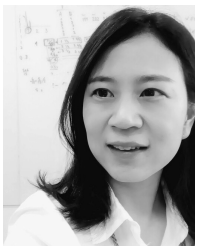
[23] J.-L. Rouas, J. Louradour, and S. Ambellouis, "Audio events detection in public transport vehicle," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 733–738.
[24] M. Gupta, V. K. Solanki, and V. K. Singh, "Analysis of datamining technique for traffic accident severity problem: A review," in *Proc. 2nd Int. Conf. Res. Intell. Comput. Eng.*, 2017, pp. 197–199.
[25] P. Baluni and Y. Raiwani, "Vehicular accident analysis using neural network," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 4, no. 9, pp. 161–164, 2014.
[26] Y. Lv, Z. Haixia, Z. Xing-lin, L. Ming, and L. Jie, "Research on accident prediction of intersection and identification method of prominent accident form based on back propagation neural network," in *Proc. ICCASM*, Oct. 2010, pp. 434–438.
[27] M. Sowmya and D. P. Ponmuthuramalingam, "Analyzing the road traffic and accidents with classification techniques," *Int. J. Comput. Trends Technol.*, vol. 5, pp. 183–188, Nov. 2013.
[28] N. Mahajan and B. P. Kaur, "Analysis of factors of road traffic accidents using enhanced decision tree algorithm," *Int. J. Comput. Appl.*, vol. 135, no. 6, pp. 1–3, 2016.
[29] S. Shanthi and R. G. Ramani, "Classification of vehicle collision patterns in road accidents using data mining algorithms," *Int. J. Comput. Appl.*, vol. 35, no. 12, pp. 30–37, 2011.
[30] E. Suganya and S. Vijayarani, "Analysis of road accidents in India using data mining classification algorithms," in *Proc. Int. Conf. Inventive Comput. Inform. (ICICI)*, Nov. 2017, pp. 1122–1126.
[31] J. R. Asor, G. M. B. Catedrilla, and J. E. Estrada, "A study on the road accidents using data investigation and visualization in Los Baños, Laguna, Philippines," in *Proc. Int. Conf. Inf. Commun. Technol. (ICOIACT)*, Mar. 2018, pp. 96–101.
[32] S. Kumar and D. Toshniwal, "A data mining approach to characterize road accident locations," *J. Modern Transp.*, vol. 24, no. 1, pp. 62–72, Mar. 2016.
[33] C. Huisingh, E. B. Levitan, M. R. Irvin, P. MacLennan, V. Wadley, and C. Owsley, "Visual sensory and visual-cognitive function and rate of crash and near-crash involvement among older drivers using naturalistic driving data," *Investigative Ophthalmol. Vis. Sci.*, vol. 58, no. 7, pp. 2959–2967, 2017.
[34] J. F. Jünior *et al.*, "Driver behavior profiling: An investigation with different smartphone sensors and machine learning," *Plos One*, vol. 12, no. 4, 2017, Art. no. e0174959.
[35] L. Ding, W. Fang, H. Luo, L. Peter, B. Zhong, and X. Ouyang, "A deep hybrid learning model to detect unsafe behavior: Integrating convolution neural networks and long short-term memory," *Autom. Construct.*, vol. 86, pp. 118–124, Feb. 2018.
[36] W. Junger and A. P. de Leon, "Imputation of missing data in time series for air pollutants," *Epidemiology*, vol. 20, no. 6, pp. 96–104, 2015.
[37] G. T. Ferrari and V. Ozaki, "Missing data imputation of climate datasets: Implications to modeling extreme drought events," *Revista Brasileira de Meteorologia*, vol. 29, no. 1, pp. 21–28, Mar. 2014.
[38] L. Pan and J. Li, "K-nearest neighbor based missing data estimation algorithm in wireless sensor networks," *Wireless Sensor Netw.*, vol. 2, no. 2, p. 115, 2010.
[39] N. Eisemann, A. Waldmann, and A. Katalinic, "Imputation of missing values of Tumour stage in population-based cancer registration," *BMC Med. Res. Methodol.*, vol. 11, no. 1, p. 129, Dec. 2011.
[40] F. Gerber, R. Furrer, G. Schaepman-Strub, R. de Jong, and M. Schaepman, "Predicting missing values in spatio-temporal satellite data," 2016, *arXiv:1605.01038*. [Online]. Available: https://arxiv.org/abs/1605.01038s
[41] C. K. Enders, *Applied Missing Data Analysis*. New York, NY, USA: Guilford Press, 2010.
[42] B. K. Vaughn, "Data analysis using regression and multilevel/hierarchical models," *J. Educ. Meas.*, vol. 45, no. 1, pp. 94–97, 2008.
[43] L. F. Burgette and P. R. Hahn, "Symmetric Bayesian multinomial probit models," Dept. Stat. Sci., Duke Univ. Durham, Durham, NC, USA, Tech. Rep. 1–20, 2010.
[44] F. Honghai, C. Guoshun, Y. Cheng, Y. Bingru, and C. Yumei, "A SVM regression based approach to filling in missing values," in *Proc. KES*, 2005, pp. 581–587.
[45] K. Pelckmans, J. D. Brabanter, J. A. Suykens, and B. De Moor, "Handling missing values in support vector machine classifiers," *Neural Netw.*, vol. 18, nos. 5–6, pp. 684–692, Aug. 2005.
[46] K. Jiang, H. Chen, and S. Yuan, "Classification for incomplete data using classifier ensembles," in *Proc. Int. Conf. Neural Netw. Brain*, Oct. 2005, pp. 559–563.
[47] M. C. Motwani, M. C. Gadiya, R. C. Motwani, and F. C. Harris, "Survey of image denoising techniques," in *Proc. GSPX*, Sep. 2004, pp. 27–30.
[48] M. Alfaouri and K. Daqrouq, "ECG signal denoising by wavelet transform thresholding," *Amer. J. Appl. Sci.*, vol. 5, no. 3, pp. 276–281, 2008.
[49] S. Joshi, R. A. Vatti, and R. V. Tornekar, "A survey on ECG signal denoising techniques," in *Proc. CSNT*, Apr. 2013, pp. 60–64.
[50] Y. Wang, W. Zhang, and H. Wang, "Application of wavelet method with nonlinear threshold control in vehicle wheel force signal denoising," in *Proc. 4th Int. Conf. Intell. Comput. Technol. Automat.*, vol. 2, Mar. 2011, pp. 957–960.
[51] H. Wang and W. Zhang, "Application of wavelet transform in vehicle wheel speed signal denoising," in *Proc. Int. Conf. Measuring Technol. Mechtron. Autom.*, vol. 3, Apr. 2009, pp. 191–194.
[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[53] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. EMNLP*, 2015, pp. 5998–6008.
[54] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[55] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. NAACL*, 2016, pp. 1480–1489.
[56] V. Govindaraj and G. Sengottaiyan, "Survey of image denoising using different filters," *Int. J. Sci., Eng. Technol. Res.*, vol. 2, no. 2, pp. 344–351, Feb. 2013.
[57] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[58] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," 2016, *arXiv:1612.08083*. [Online]. Available: https://arxiv.org/abs/1612.08083
[59] S. Czesla, T. Molle, and J. H. M. M. Schmitt, "A posteriori noise estimation in variable data sets - with applications to spectra and light curves," *Astron. Astrophys.*, vol. 609, p. A39, Jan. 2018.
[60] S. Wisdom, T. Powers, J. Hershey, J. Le Roux, and L. Atlas, "Full-capacity unitary recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4880–4888.
[61] F. Belletti, A. Beutel, S. Jain, and E. Chi, "Factorized recurrent neural architectures for longer range dependence," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1522–1530.
[62] T. H. Trinh, A. M. Dai, M.-T. Luong, and Q. V. Le, "Learning longer-term dependencies in RNNS with auxiliary losses," 2018, *arXiv:1803.00144*. [Online]. Available: https://arxiv.org/abs/1803.00144

Sungjoon Park received the B.A. and M.A. degrees in psychology from Seoul National University, Seoul, South Korea, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in machine learning and natural language processing with the Users and Information Laboratory, Department of Computing, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. His research interests include time-series data analysis, representation learning, natural language processing, and computational psychology.

Yeon Seonwoo received the B.S. degree in computer science and engineering from Sungkyunkwan University, South Korea, in 2016, and the M.S. degree in machine learning from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2017, where he is currently pursuing the Ph.D. degree in machine learning. His research interests include machine learning, stochastic process, time-series data analysis, and representation learning.

Jiseon Kim received the B.S. degree from the Division of Computer Science, Sookmyung Women's University, South Korea, in 2017. She is currently pursuing the M.S. degree in computer science with the Korea Advanced Institute of Science and Technology (KAIST). Her research interests include natural language processing and computational social science.

Jooyeon Kim received the B.E. degree in systems innovation from the University of Tokyo in 2014, and the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 2016, where he is currently pursuing the Ph.D. degree in computer science. His research interests include Bayesian machine learning, text and network analysis, and game analytics.



Alice Oh received the master's degree in language and information technologies from CMU, and the Ph.D. degree in computer science from MIT. She is currently an Associate Professor in computer science with KAIST. Her research interest includes developing and applying machine learning models for human social behavior data. She has served on various technical committees, including ACL, EMNLP, ACM CHI, ACM WSDM, ACM KDD, and AAAI.