Speech-Based Detection of In-Car Escalating Arguments to Prevent Distracted Driving

Francesco Pistolesi*, Michele Baldassini*‡, and Beatrice Lazzerini*
* Department of Information Engineering, University of Pisa - Largo L. Lazzarino 1, 56122 Pisa (Italy)
‡ Department of Information Engineering, University of Florence - Via di S. Marta 3, 50139 Florence (Italy)
francesco.pistolesi@unipi.it, michele.baldassini@ing.unipi.it, beatrice.lazzerini@unipi.it

***Abstract*—Every year, 2.5 million car crashes involve distracted drivers globally. It takes a few seconds for a car crash to happen after the driver has been distracted. Distracted driving thus poses a critical threat to road safety, needing innovative approaches for its detection and mitigation. This paper introduces a novel system to monitor in-car conversations and identify potential distractions from escalating arguments. The system analyzes Mel spectrograms generated from real-time audio signals containing in-car discussions by combining continuous voice recording and deep learning techniques. First, a *denoiser* employs a convolutional autoencoder to reduce car engine noise within the spectrograms. Then, a *classifier* uses convolutional and recurrent neural networks to determine whether the audio corresponds to a calm conversation or a quarrel based on the denoised spectrogram. The experimental results showed that the system achieved a 91.8% classification accuracy. This system addresses a previously unexplored dimension of cognitive distraction, offering valuable insights into strategies for reducing the risk of road accidents. Ongoing research is focused on accounting for other environmental noises, such as radio speakers, music, wind from open windows, and engine sounds from surrounding vehicles, which may influence classification accuracy. The system is also being extended to consider more than two occupants in the car.**

***Index Terms*—Artificial intelligence, deep learning, distracted driving, emotion recognition, intelligent transportation systems, road safety, speech.**

I. Introduction

Distracted driving is one of the gravest threats on today's roadways, imperiling countless lives and contributing significantly to vehicular accidents. This issue transcends the traditional notion of distraction, encompassing a spectrum of behaviors that divert a driver's attention from the primary task of safe vehicle operation. While people's awareness of distracted driving has grown, its manifestations continue to evolve, necessitating solutions to mitigate this threat.

Three categories of distracted driving exist: visual, manual, and cognitive distractions. *Visual distractions* divert the driver's gaze away from the road, *manual distractions* involve the removal of hands from the steering wheel, and *cognitive distractions* divert the driver's mental focus from driving.

Distracted driving has a leading role in road safety. Statistics say that approximately 3,000 lives are lost yearly in the United States because of auto accidents due to distracted driving. For example, in 2021, there were 3,522 people killed and an additional 362,415 people injured in traffic crashes involving distracted drivers in the United States alone [1]. This accounts for a staggering 8% to 9% of all fatal motor vehicle collisions. It is worth noting that while cell phone usage is a prominent factor, other distractions are equally dangerous, such as eating, talking to passengers, adjusting the radio or climate controls, or adjusting other vehicle controls [1].

Existing solutions to address distracted driving include machine learning-based methods that use smartphone sensors such as accelerometers and gyroscopes [2] or in-vehicle phone localization schemes to determine the locations of smartphones inside a moving car [3]. Various solutions based on deep learning led to high levels of accuracy but often demand substantial computational resources and may not always meet real-time detection requirements [4], [5]. For example, lightweight convolutional neural networks might trade off accuracy in certain scenarios [6]. Temporal-spatial double-line deep learning networks with causal AND-OR graphs showed promising results in continuous recognition [7]. Still, they can be complex to implement and require fine-tuning for specific applications. Also, a fine-tuned vision transformer led to high accuracy but could demand substantial computational resources [8]. Reinforcement learning methods were proposed to address pedestrian-related distractions [9].

Wearable-based systems were also used to prevent distracted driving. For example, those using Bluetooth technology have shown to be energy-efficient but may still require user cooperation for training [10]. Arousal estimation through physiological signals was performed based on the electrodermal activity (EDA) [11], [12], perinasal perspiration signal [13], and electroencephalography (EEG) signals [14], [15]. Although interesting, these approaches may necessitate additional hardware for implementation or cumbersome wearable devices.

Computer vision characterized various alternative solutions [16]. Interesting examples include tracking-based human action recognition schemes [17], knowledge-distillation-based frameworks [18], and vision-based systems to detect Eyes Off the Road (EOR) [19]. Although accurate, these methods require camera placement, setup, and continuous video recording of drivers. Also, the increase in accuracy is generally correlated with an increase in the number of parameters. This is a problem for real-world applications because of the

"This work was co-funded by the European Union as part of the *MEDEA* European project (Grant No. 101101236), and supported in part by the Italian Ministry of University and Research (MUR) in the frameworks of the *CrossLab* and *FoReLab* projects (Departments of Excellence), and by the PNRR-M4C2-Investimento 1.3, Partenariato Esteso PE00000013-FAIR—Future Artificial Intelligence Research-Spoke 1 Human-Centered AI."

limitations of in-car computing equipment.

In general, existing technologies have made significant strides in addressing distracted driving. Still, they need additional hardware, high computational demand, and focus on specific distraction scenarios. Also, most existing methods focus on visual or manual distraction, whereas a few have explored cognitive distraction [20].

Among the causes of cognitive distraction, heated discussions inside the vehicle are crucial. These discussions may or may not involve the driver. For example, the driver may engage in a heated argument with a passenger, other passengers may argue among themselves, or the driver may have a heated conversation on the phone, although using hands-free devices and keeping the hands on the steering wheel. All of these situations distract the driver and increase the risk of accidents. This type of cognitive distraction is common and needs to be explored in the literature.

This paper describes a preliminary version of a novel system addressing distracted driving that detects in-car arguments using audio sensors and deep learning. The system can detect heightened tension by monitoring vocal stress levels and arguments among vehicle occupants. The system continuously analyzes the in-vehicle audio, computing the Mel spectrogram of 5-second-long audio segments. Then, a convolutional autoencoder removes the car engine noise, and a further module based on a convolutional neural network (CNN) and a gate recurrent unit (GRU) classifies the audio segment as *calm conversation* or *heated conversation*. In the era of Big Data, where the volume and complexity of information continue to grow exponentially, our research addresses a pressing real-world problem by using machine learning techniques to analyze vast amounts of in-car audio data.

The paper is organized as follows: Section II gives a background on audio analysis and deep learning; Section III presents the dataset; Section IV describes the system; Section V presents the experiments and their results; Section VI draws the conclusions.

II. Background

A. Audio files

An audio file, denoted as $x[n]$, represents sound as a discrete-time signal, where n is the discrete-time index. It quantifies the sound's amplitude at each time step and is sampled at F_s, determining the samples per second.

A waveform illustrates sound by depicting how air pressure (y-axis) changes over time (x-axis) as a continuous curve. The continuous-time audio signal, $x(t)$, can be expressed as:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \operatorname{sinc}\left(\pi(F_s t - n)\right). \tag{1}$$

where $\operatorname{sinc}(x) = \sin(x)/x$.

B. Spectrogram in Decibels

A spectrogram visually displays sound energy distribution across frequencies over time. It is derived from an audio signal's Short-Time Fourier Transform (STFT), which dissects the signal into overlapping segments, computes their Fourier transforms, and combines the results. Formally:

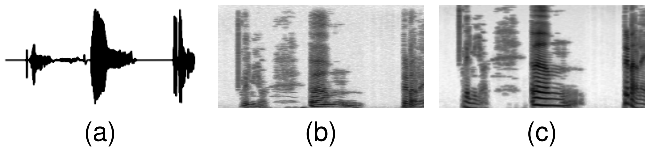

Fig. 1. An audio signal and its waveform (a), spectrogram (b), and Mel spectrogram (c).

$$X(m,\omega) = \sum_{n=0}^{N-1} x[n] \cdot w[n-mR] \cdot e^{-j\omega n} \tag{2}$$

where $X(m,\omega)$ is the STFT, $x[n]$ is the audio signal, $w[n]$ is the window function, N is window size, R is overlap, and $e^{-j\omega n}$ is a complex exponential term.

Creating a spectrogram requires two steps on the magnitude values $X(m,\omega)$. First, we take the natural logarithm for dynamic range compression. Second, we scale these values in decibels (dB) using a factor of 10. This results in the spectrogram in dB, denoted as $S_{\text{dB}}(m,\omega)$:

$$S_{\text{dB}}(m,\omega) = 10\log_{10}\left(|X(m,\omega)|^2\right). \tag{3}$$

where m represents time windows, and ω denotes frequency bins.

C. Mel Spectrogram

Mel's spectrogram, denoted as $M(f_{\text{mel}}, t)$, is a representation of an audio signal's spectral content in the Mel-frequency scale in a way that aligns better with human auditory perception. Computing the Mel spectrogram requires a standard spectrogram, $S_{\text{dB}}(f,t)$. Then, linear frequencies (f) are converted to the Mel-frequency scale $f_{\text{mel}} = 2595 \cdot \log_{10}(1 + \frac{f}{700})$. This conversion maps frequencies in a way that is closer to human auditory perception. A set of triangular filters are then applied to ensure logarithmic spacing in the linear frequency scale. Each filter $H_k(f_{\text{mel}})$ is defined by a center frequency. Center frequencies are evenly spaced in the Mel scale. For each time window (t) in the spectrogram, these Mel filters are applied to the magnitude spectrogram $S_{\text{dB}}(f,t)$ by summing the products of the filter responses and the magnitude spectrogram values:

$$M_k(t) = \sum_{f_{\text{mel}}} S_{\text{dB}}(f_{\text{mel}}, t) \cdot H_k(f_{\text{mel}}) \tag{4}$$

The outcome, $M(f_{\text{mel}}, t)$, represents the audio signal's energy distribution across Mel-frequency bands over time.

D. Deep Neural Networks

Deep Neural Networks (DNNs) are machine learning models inspired by the human brain. They consist of interconnected artificial neurons organized into layers. Each neuron processes information and passes it to the next layer, thereby performing complex data transformations. DNNs are the key in many modern AI applications, ranging from image recognition to natural language understanding.

1) Convolutional Neural Networks (CNNs): A CNN is a deep learning model typically used for tasks involving grid-like input data like images. A CNN uses a series of convolutional layers that learn and extract meaningful patterns or features from the input data. These layers use small filters to perform convolutions across the data, thus generating feature maps that capture relevant information. These features become increasingly complex and abstract as the data flows through the network. Pooling layers are often used to downsample the data, reducing spatial dimensions while retaining essential details. Fully-connected layers at the end of the network enable the CNN to make predictions or classifications based on the learned features.

2) Gated Recurrent Unit (GRU): A GRU is a recurrent neural network (RNN) architecture that processes sequential data. GRUs consist of a reset gate and an update gate, and use a gating system to regulate the information flow through the network. In particular, the reset gate determines what information from the previous time step should be forgotten or reset, whereas the update gate decides the new information to store. This mechanism allows GRUs to capture long-range dependencies in sequences. GRUs are typically used in natural language processing tasks and speech recognition.

III. Dataset

This section describes the data used to train and test the system, which included two types of sound: *voice audio* (calm and heated conversations) and *car engine noise*.

A. Voice audio

Voice audio files were generated considering TV talk shows and films, which were examined to find scenes of quarrels and calm discussions. Voice audio files comprised scenes without any soundtrack. Also, when selecting voice audio files of conversations, we paid attention to including both loud and soft-spoken discussions in equal quantities. The resulting set comprised 32 and 29 voice audio files for the *quarrel* and *calm discussion*, respectively.

B. Car engine noise

The audio files containing car engine noise were collected from YouTube. In particular, various videos of car tests were watched, looking for typical engine noises audible inside a car. Car test videos were selected as they are recorded inside the car and faithfully reproduce the car engine noise that typically overlaps with speech when in a car. The set of car engine noises comprised 10 audio files lasting from 8 to 10 minutes, each recorded in a different car.

C. Data Partitioning

Voice and noise files were randomly selected to obtain 3 sets as follows:

- set *A*: 20 files containing *calm discussions*, 22 files containing *quarrels*, 6 files containing *engine noise*;
- set *B*: 5 files containing *calm discussions*, 5 files containing *quarrels*, 2 files containing *engine noise*;
- set *C*: 4 files containing *calm discussions*, 5 files containing *quarrels*, 2 files containing *engine noise*.

Each audio file only belonged to one set (*A*, *B*, or *C*). The files in set *A* were used for the training, whereas those in sets *B* and *C* were used for validation and testing, respectively.

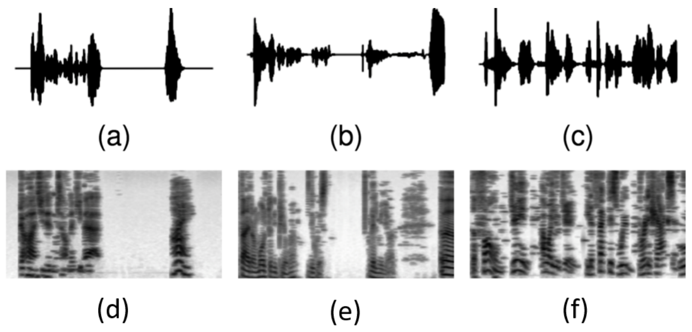

Fig. 2. Waveforms of voice audio comprising calm discussions (first row) and corresponding Mel spectrograms (second row).

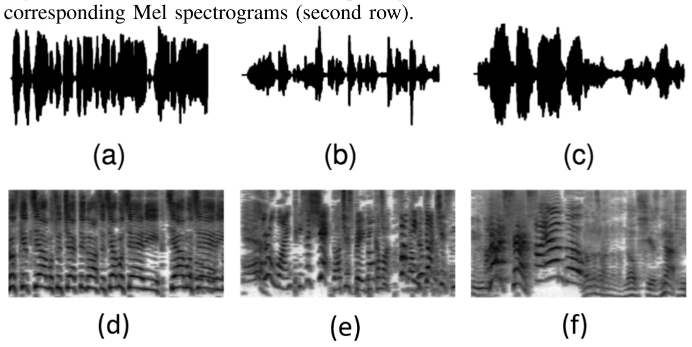

Fig. 3. Waveforms of voice audio comprising quarrels (first row) and corresponding Mel spectrograms (second row)

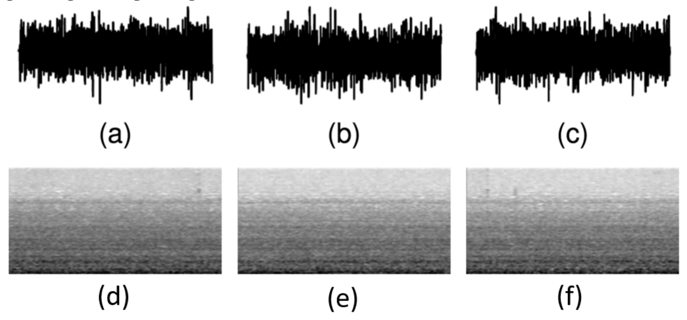

Fig. 4. Waveforms of audio comprising car engine noise (first row) and corresponding Mel spectrograms (second row)

D. Mixing audio files

As voice and noise audio files were characterized by different sampling rates (44.1 kHz and 32 kHz, respectively), those at 44.1 kHz were under-sampled at 32 kHz.

Then, both voice and noise audio files were divided into segments whose length L was determined by considering the parameters used for the spectrogram computation (window size N and overlap R) as follows:

$$L = R(\text{number\_of\_windows} - 1) + N. \tag{5}$$

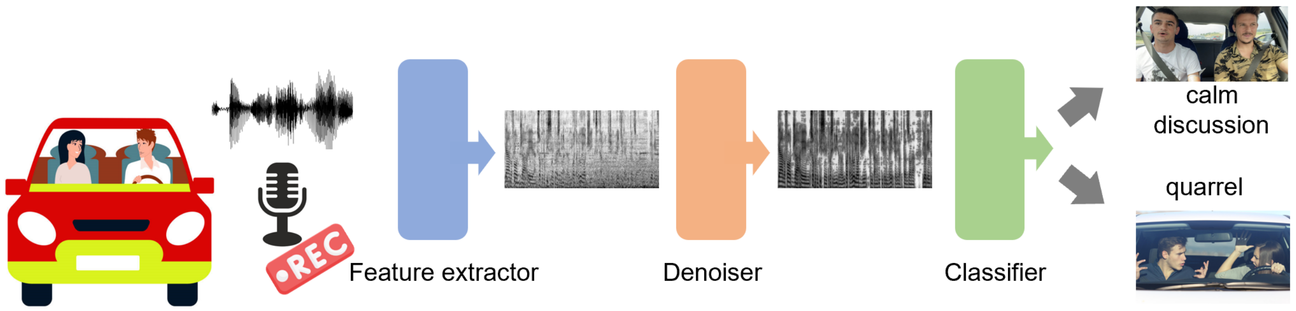

Fig. 5. System overview. The *Feature extractor* computes the Mel spectrogram of the audio segment. The *Denoiser* reduces the car engine noise in the spectrogram, and the *Classifier* takes as input the denoised spectrogram and recognizes if the spectrogram contains a calm discussion or a quarrel.

The number of windows thus contributed to determining the length of the audio segments.

All possible combinations of voice and noise segments were then generated: each combination represented an *audio segment (AS)*. When mixing an audio segment and a noise segment, the noise volume was either reduced or increased to avoid covering the voice or being too low, respectively. This was done by carefully listening to each AS, thereby finding the best compromise between the voice and noise volumes. This procedure generated three roughly balanced sets, A, B, and C, containing 258813, 5262, and 7626 ASes, respectively. Fig. 6 shows how to generate ASes from voice and noise audio files.

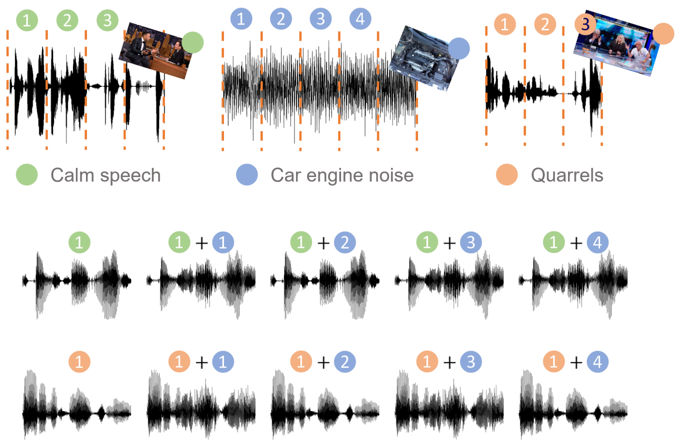

Fig. 6. Procedure to generate ASes from voice and noise audio files. First, the audio files are segmented (first row). Then, all possible combinations of voice and noise audio segments are calculated, and the corresponding audio segments are mixed (second and third rows).

IV. System

This section outlines the system and describes its modules.

A. Overview

The system is made up of three modules (see Fig. 5):

1) *Feature extractor*: computes the Mel spectrogram of an AS as described in Section IV-B);
2) *Denoiser*: reduces the noise level in the Mel spectogram (see Section IV-C);
3) *Classifier*: determines if the denoised spectrogram stems from an AS containing a calm discussion or a quarrel (see Section IV-D).

B. Feature extractor

As explained in Section II-B, computing the Mel spectrogram requires the spectrogram in Decibels, whose parameters are the window size N, along with the overlap R, i.e., the time interval (number of samples) between the beginnings of two consecutive windows. We used a window size N=2048 samples (at 32 kHz) to compromise between the information content within the windows and the computational time to calculate the corresponding Fourier transforms. This choice yielded a window lasting 2048/32 kHz = 64 ms. As speech can be considered pseudo-stationary for up to 20 ms (i.e., 640 samples) [21], we used windows spaced 512 samples apart (75% of overlap). The time interval between the beginnings of two consecutive windows was thus 16 ms: 512 samples/32 kHz. The frequencies were then converted into Decibels. An example spectrogram is shown in Fig. 7b, where the horizontal and vertical components represent the time and frequency, respectively.

To convert the spectrogram frequencies to the Mel scale, the Hz scale was first divided into 128 bins. Then, using overlapping triangular filters, each bin was transformed into a corresponding Mel scale bin. Fig. 7c shows an example Mel spectrogram whose vertical component represents the 128 Mel scale bin.

C. Denoiser

The denoiser removed the car engine noise from the Mel spectrogram using an autoencoder based on convolutional layers to capture spatial dependencies between pixels. The roles of the autoencoder's encoder and decoder are described in the following sections. Fig. 9 shows the denoiser architecture.

1) Encoder: The encoder takes in input a grayscale Mel spectrogram obtained from an AS (i.e., voice + noise) and generates a compressed representation of it. In particular, the encoder contains a series of convolutional layers, each creating a set of feature maps by applying various filters to the input image. The amount of compression can be modified by varying the size of the filters and the number of feature maps. After

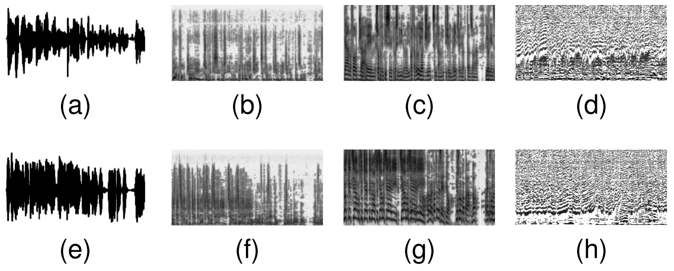

Fig. 7. Waveforms of voice audio segments of a calm discussion (a), spectrogram (b), Mel spectrogram (c), and MFCC (d). Waveforms of voice audio segments of a quarrel (e), spectrogram (f), Mel spectrogram (g), and MFCC (h).

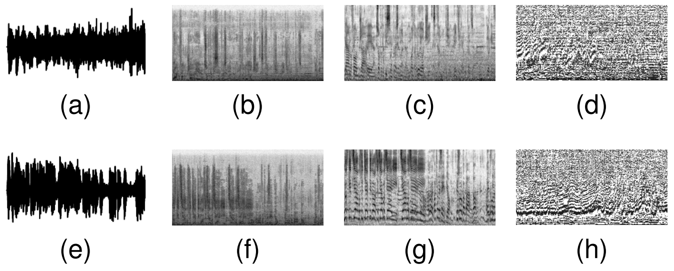

Fig. 8. Waveforms of audio segments (voice + noise) of a calm discussion (a), spectrogram (b), Mel spectrogram (c), and MFCC (d). Waveforms of voice audio segments of a quarrel (e), spectrogram (f), Mel spectrogram (g), and MFCC (h).

various experiments, we decided to halve the size of the feature maps from one layer to the next: the stride of each convolutional layer was thus set to two.

2) Decoder: The decoder takes in input the encoder representation of the Mel spectrogram and generates a denoised spectrogram by using a series of transposed convolutional layers (deconvolutional layers). Each deconvolutional layer applies a set of filters to the input feature maps, creating a set of output feature maps. The size of the filters and the number of output feature maps can be adjusted to control the level of reconstruction. To double the size of the feature maps from one layer to the next, the stride of each deconvolutional layer was set to two.

3) Denoising procedure: As can be surmised by comparing Fig. 7b to Fig. 8b, and Fig. 7c to Fig. 8c, the car engine noise significantly affects the voice sound and is hard to recognize. The autoencoder needs information about the noise to identify and remove it, thereby reconstructing a denoised spectrogram. To this aim, the autoencoder was trained using the noisy spectrogram of an AS as input, and the noise-free spectrogram generated by the voice segment of that AS as target.

D. Classifier

The classifier comprised a CNN followed by a GRU and a Multi-Layer Perceptron (MLP).

1) CNN: The CNN takes the noise-free spectrogram generated by the denoiser, and captures its spatial and temporal relationships by generating a feature map tensor $\mathbf{F} \in \mathbb{R}^{w\times h\times c}$, where w and h are the width and the height of each map, and c is the number of feature maps.

Each feature map $\mathbf{F}_i \in \mathbb{R}^{w\times h}$, was reshaped by vertically concatenating its columns one after the other. The reshaping of a feature map $\mathbf{F}_i$ thus generated a column vector $\mathbf{f}_i \in \mathbb{R}^{w\cdot h\times 1}$. The same procedure was followed for all feature maps $\mathbf{F}_1,\ldots,\mathbf{F}_c$, thus obtaining column vectors $\mathbf{f}_1,\ldots,\mathbf{f}_c$. Concatenating the k-th elements $f_1^k,\ldots,f_c^k$ of all column vectors $\mathbf{f}_1,\ldots,\mathbf{f}_c$, led to defining the k-th time step $(f_1^k,\ldots,f_c^k)$ of a multivariate time series $<(f_1^1,\ldots,f_c^1),\ldots,(f_1^{w\cdot h},\ldots,f_c^{w\cdot h})>$ made up of $w\cdot h$ timesteps, each made up of c elements (see Fig. 10).

2) GRU: The GRU consisted of two stacked recurrent layers. The first layer reduced the number of elements of the time series, whereas the second layer returned a vector

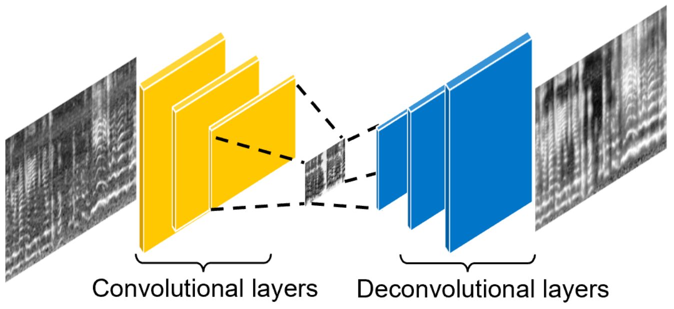

Fig. 9. Denoiser overview. The denoiser takes the Mel spectrogram comprising voice and noise and returns the denoised spectrogram.

computed after processing the last time step of the time series. The output's shape of the first layer was thus $w\cdot h\times k_1$, where k_1 was the number of hidden units of the layer. The second layer produced an output vector made up of k_2 elements, where k_2 was the number of hidden units of the second layer.

3) MLP: The MLP had two hidden layers and an output layer. The input of the MLP was the output vector generated by the GRU. The output layer consisted of one neuron with a sigmoid activation function to perform a binary classification: *calm discussion* or *quarrel*.

E. Experiments

A grid search was performed to find the best architecture for the denoiser and classifier.

1) Denoiser: The denoiser was optimized by altering the compression rate of the autoencoder (AE). In particular, the number of convolutional layers varied from 2 to 5, with filter sizes of 3×3, 5×5, and 7×7, and the number of filters equal to 16, 32, 64 and 128. The AEs were trained by repeating 10-fold cross-validation three times for each combination of hyperparameter values, each time shuffling the data.

Given a spectrogram $\mathbf{X} \in [0,1]^{n\times m}$, the quality of its reconstruction $\mathbf{Y} \in [0,1]^{n\times m}$ was assessed using the mean absolute error (MAE) and the structural similarity index measure (SSIM) [22] in the following loss function:

$$\mathcal{L} = \mathrm{MAE}(\mathbf{X},\mathbf{Y}) + \mathrm{SSIM}(\mathbf{X},\mathbf{Y}) \tag{6}$$

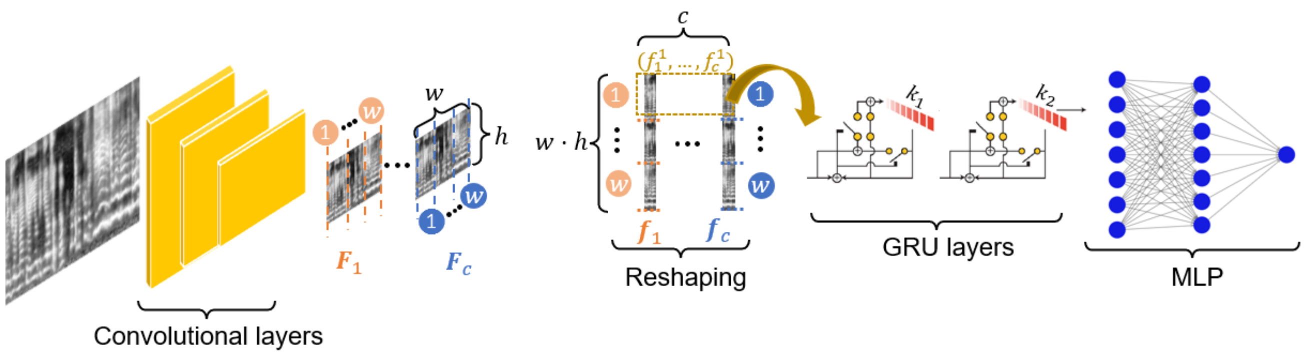

Fig. 10. Classifier overview. The CNN takes the spectrogram generated by the denoiser and computes a feature map tensor $\mathbf{F} \in \mathbb{R}^{w \times h \times c}$. All feature maps $\mathbf{F}_1, \dots, \mathbf{F}_c$ are reshaped, thus obtaining column vectors $\mathbf{f}_1, \dots, \mathbf{f}_c$. The GRU layers process the time series, generating an output vector that is then fed as input to the MLP. The MLP performs the binary classification.

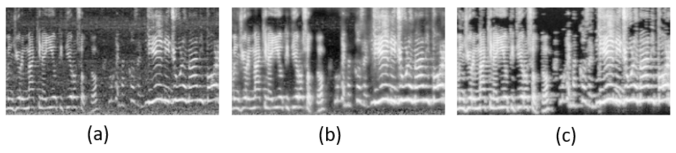

Fig. 11. The Mel spectrogram computed from an AS comprising a calm discussion (a), the Mel spectrogram denoised by the AE (b), and the noise-free Mel spectrogram computed only using the voice segment of the AS (c).

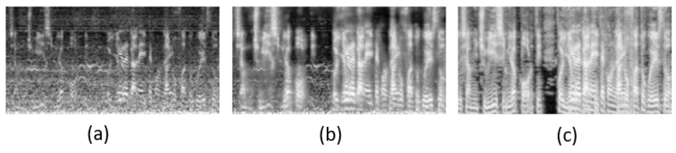

Fig. 12. The Mel spectrogram computed from an AS comprising a quarrel (a), the Mel spectrogram denoised by the AE (b), and the Mel spectrogram computed from the voice segment contained in the AS (c).

where

$$\mathrm{MAE}(\mathbf{X}, \mathbf{Y}) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \mid y_{ij} - x_{ij} \mid, \tag{7}$$

and

$$\mathrm{SSIM}(\mathbf{X}, \mathbf{Y}) = \frac{(2\mu_{\mathbf{X}}\mu_{\mathbf{Y}} + C_1) + (2\sigma_{\mathbf{XY}} + C_2)}{(\mu_{\mathbf{X}^2} + \mu_{\mathbf{Y}}^2 + C_1)(\sigma_{\mathbf{X}}^2 + \sigma_{\mathbf{Y}}^2 + C_2)}, \tag{8}$$

where $\mu_{\mathbf{X}}$ is the pixel sample mean of $\mathbf{X}$, $\mu_{\mathbf{Y}}$ the pixel sample mean of $\mathbf{Y}$, $\sigma_{\mathbf{X}}^2$ the variance of $\mathbf{X}$, $\sigma_{\mathbf{Y}}^2$ the variance of $\mathbf{Y}$, $\sigma_{\mathbf{XY}}$ the covariance of $\mathbf{X}$ and $\mathbf{Y}$, C_1 and C_2 are two variables to balance the division with a weak denominator. The values of C1 and C2 were set to 0.01 as this is a widely used value in the image processing community [22].

The lowest average loss on the test folds was considered as a criterion for selecting the best architecture. The Rectified Linear Unit (ReLU) was used as the activation function for both convolutional and deconvolutional layers as it runs much faster than other activation functions [23]. The AE obtained the best performance with an encoder comprising three convolutional layers with 16, 32, and 64 filters of size 3×3. The decoder consisted of 3 convolutional layers with 64, 32, and 16 filters of size 3×3.

2) Classifier: The grid search was performed by varying the hyperparameter values of the CNN, the number of neurons in both layers of the GRU, and the number of hidden layers and neurons in each hidden layer of the MLP. A number of convolutional layers in {2,...,6} was tested trying filter sizes of 3×3, 5×5, and 7×7. A number of neurons in the GRU layers were tested ranging from 32 to 256, with step 32.

TABLE I
MEAN LEVELS OF LOSS AND CORRESPONDING STANDARD DEVIATIONS OF THE DENOISER ON THE TEST FOLDS

AE's loss	Features		
	Spectrogram	Mel Spectrogram	MFCC
MSE	0.63 ± 0.104	0.57 ± 0.110	0.73 ± 0.094
MAE	0.57 ± 0.111	0.48 ± 0.106	0.62 ± 0.115
MSE + SSIM	0.66 ± 0.134	0.59 ± 0.082	0.69 ± 0.151
MAE + SSIM	0.41 ± 0.121	**0.33 ± 0.017**	0.52 ± 0.115

Up to two hidden layers in the MLP were tried, varying the number of neurons in each layer from 32 to 512, with step 32. The logistic sigmoid and the hyperbolic tangent functions were tested as activation and recurrent activation functions for the GRU and MLP layers. For the MLP, the ReLU activation function was also tested.

For each combination of the hyperparameters, the 10-fold cross-validation was repeated 3 times, each time shuffling data to train the models with different configurations of the folds.

The binary cross-entropy was used as a loss function to train the binary classifier. In particular, the lowest mean cross-entropy on the test folds was considered as a criterion to select the best architecture, which consisted of 3 convolutional layers with 16, 32, and 64 filters of size 3×3 for the CNN, 2 recurrent layers with 64 hidden neurons for the GRU, and 2 layers with 64 neurons for the MLP.

V. Experiments and Discussion

This section presents the experiments and discusses their results by comparing the performances of the system modules

TABLE II
MEAN LEVELS OF ACCURACY AND CORRESPONDING STANDARD DEVIATIONS ON THE TEST FOLDS OF THE TESTED CLASSIFIERS

Classifiers	**Features**					
	with denoise			without denoise		
	Spectrogram	Mel Spectrogram	MFCC	Spectrogram	Mel Spectrogram	MFCC
CNN + RNN	0.885 ± 0.184	$\mathbf{0.925 \pm 0.152}$	0.874 ± 0.191	0.842 ± 0.176	0.865 ± 0.129	0.831 ± 0.138
MobileNetV2	0.855 ± 0.111	0.871 ± 0.106	0.842 ± 0.115	0.820 ± 0.203	0.854 ± 0.183	0.814 ± 0.192
NasNetMobile	0.860 ± 0.134	0.882 ± 0.098	0.847 ± 0.178	0.834 ± 0.113	0.862 ± 0.192	0.821 ± 0.174
EfficientNetV2	0.870 ± 0.121	0.895 ± 0.167	0.904 ± 0.115	0.843 ± 0.103	0.872 ± 0.184	0.831 ± 0.145

to those of alternative denoisers (AEs) and classifiers. The hyperparameters of each model were optimized as described in Section IV-C and Section IV-D.

A. Alternative denoisers

The alternative AEs were trained using spectrograms, Mel spectrograms, and Mel-frequency cepstral coefficients (MFCCs) [24]. The MFCCs were derived from the Mel spectrogram by first converting the 128 Mel frequencies in Decibels and then computing the discrete cosine transform of the 128 frequencies. The resulting spectrum amplitudes represent the MFCCs. For example, Fig. 7d shows the MFCCs computed using the Mel spectrogram of Fig. 7c.

The performance was evaluated based on different loss functions: mean squaredMSE, MAE, MSE+SSIM, and MAE+SSIM. As Fig. 11 and 12 show, the Mel spectrogram can be reconstructed faithfully. Instead, when reducing noise from spectrograms in Decibels, there was a loss of information related to speech. Also, reducing noise from MFCCs led to MFCCs with poor information regarding speech. Table I shows the loss value of each AE trained using the various features.

B. Alternative classifiers

Alternative classifiers were obtained by exchanging the convolutional base with those of some widely used pre-trained networks. The weights of these networks were set by training the network using the *Imagenet* dataset. The memory required and inference time were considered as criteria to minimize in order to choose the pre-trained models. The models selected were MobileNetV2 [25], NASNetMobile [26], and EfficientNetB0 [27], which require 14, 23, and 29 MB of memory, respectively. These models also have quick inference times. In particular, without using a GPU, EfficientNetB0 has the longest inference time (46 ms); NASNetMobile and MobileNetV2 take 27 ms and 26 ms, respectively.

Transfer learning helped reuse the knowledge learned from the pre-trained models to distinguish audio segments containing calm discussions from those containing quarrels. In particular, while freezing the convolutional base of each pre-trained model, the MLP (i.e., the final part of the classifier) was replaced with an MLP made up of two hidden layers. The number of neurons in each hidden layer varied from 32 to 128, with step 32. The hyperparameters of each model were optimized by three executions of the 10-fold cross-validation, each time shuffling data.

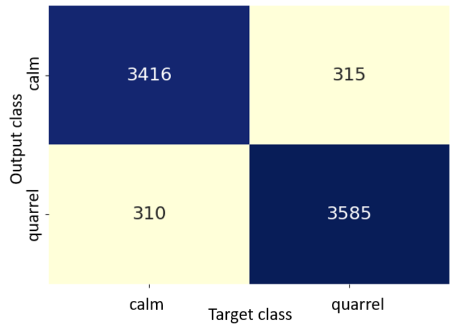

Fig. 13. Confusion matrix obtained testing the classifier on set *C*.

After training the MLP, the last convolutional layers were unfrozen to extract more characteristic features to recognize quarrels. In particular, up to three convolutional layers were gradually unfrozen (one at a time) for each pre-trained model, and the model was retrained.

Transfer learning was performed by keeping a small learning rate equal to $1 \cdot e^{-6}$ to avoid significant variations to the network weights that would erase the learned knowledge. The best hyperparameter configuration for each pre-trained model was as follows:

- MobileNetV2: 96 neurons in the first MLP's hidden layer, 64 neurons in the second MLP's hidden layer, and the last two convolutional layers unfrozen;
- NASNetMobile: 64 neurons in the first MLP's hidden layer, 64 neurons in the second MLP's hidden layer, and the last convolutional layer unfrozen;
- EfficientNetB0: 128 neurons in the first MLP's hidden layer, 96 neurons in the second MLP's hidden layer, and the last three convolutional layers unfrozen.

The experiments were repeated, this time training the models using different inputs: the spectrogram, Mel spectrogram, and MFCCs. Table II shows each classifier's mean accuracy and standard deviation.

C. Discussion

The model that achieved the highest accuracy on the validation set (set *B*) was finally tested on the test set (set *C*). As explained in Section III-C, sets *B* and *C* did not share any audio file. Testing the system on set *C* was crucial to verify its generalization capability, i.e., to ensure that it had not become accustomed to the voices in the files that generated the audio segments used for training.

During the final test, the system classifier achieved an accuracy of 91.8%. The F1 score was also calculated to evaluate the model's predictive ability inside each class. The F1 score is defined as $F_1 = 2(p \cdot r)/(p + r)$, where p and r are the precision and recall, respectively. In particular, the *precision* is the ratio of true positives to all positive samples, including those incorrectly classified as positive; the *recall* is the ratio of true positives to the samples that should have been classified as positive. The F1 scores were 90% and 88% for the calm discussion and quarrels classes, respectively. Fig. 13 shows the confusion matrix, whose main diagonal contains the count of samples correctly classified for each class.

As the matrix shows, the system achieved a high accuracy. In particular, the classifier misclassified 315 samples of calm discussion ($\sim$8.3%) and 310 samples of quarrels ($\sim$8%). The misclassification rate was thus approximately the same for both classes. This suggests that future system versions could be integrated into modern Advanced Driver Assistance Systems (ADAS) to alert the driver or automatically reduce the vehicle's speed in the case of heated discussions inside the car. This could help considerably reduce the risk of accidents due to cognitive distractions and save lives.

VI. Conclusions

This paper has presented a preliminary version of a system that detects in-car heated conversations that may distract a driver based on sound analysis and deep learning. The system achieved promising levels of accuracy up to 91.81% in distinguishing in-car calm conversations from quarrels.

Although this system version only considers engine noise as a disturbance source for vocal audio, we are generating a much larger dataset that considers multiple noise sources that are typical when inside a car, including the presence of more than two passengers.

The proposed system could be integrated into modern advanced driver assistance systems (ADAS) to quickly detect cognitive distractions caused by the driver's involvement in heated discussions, significantly reducing the risk of accidents.

Acknowledgment

The authors wish to thank Ettore Ricci for his contribution to creating the dataset.

References

[1] N. C. for Statistics and Analysis, "Distracted driving in 2021," May 2023.

[2] K. Ben Ahmed *et al.*, "Leveraging smartphone sensors to detect distracted driving activities," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3303–3312, 2019.

[3] C.-Y. Chen and K. G. Shin, "In-vehicle phone localization for prevention of distracted driving," *IEEE Transactions on Mobile Computing*, vol. 22, no. 6, pp. 3365–3379, 2023.

[4] B. Qin *et al.*, "Distracted driver detection based on a cnn with decreasing filter size," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6922–6933, 2022.

[5] C. Ou and F. Karray, "Enhancing driver distraction recognition using generative adversarial networks," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 3, pp. 385–396, 2020.

[6] P. Li *et al.*, "Driver distraction detection using octave-like convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8823–8833, 2022.

[7] P. Ping *et al.*, "Distracted driving detection based on the fusion of deep learning and causal reasoning," *Information Fusion*, vol. 89, pp. 121–142, 2023.

[8] H. Chen *et al.*, "Distracted driving recognition using vision transformer for human-machine co-driving," in *2021 5th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2021, pp. 1–7.

[9] G. P. Rosati Papini *et al.*, "A reinforcement learning approach for enacting cautious behaviours in autonomous driving system: Safe speed choice in the interaction with distracted pedestrians," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8805–8822, 2022.

[10] T. Mewborne *et al.*, "Distracted driving detection utilizing wearable-based bluetooth," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2022, pp. 485–490.

[11] L. Gomero *et al.*, "Evaluation of electrodermal activity during distracted driving," in *2022 IEEE 3rd International Conference on Human-Machine Systems (ICHMS)*, 2022, pp. 1–1.

[12] F. Pistolesi *et al.*, "A smartphone app to collect emotion-labeled signals in the wild using a body sensor network," in *2022 IEEE International Symposium on Multimedia (ISM)*, 2022, pp. 159–160.

[13] I. Pavlidis *et al.*, "Biofeedback arrests sympathetic and behavioral effects in distracted driving," *IEEE Transactions on Affective Computing*, vol. 12, no. 2, pp. 453–465, 2021.

[14] G. Li *et al.*, "A temporal–spatial deep learning approach for driver distraction detection based on eeg signals," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2665–2677, 2022.

[15] M. Baldassini *et al.*, "Detecting happiness from 14-channel binary-valued eeg charts via deep learning," in *2022 IEEE International Symposium on Multimedia (ISM)*, 2022, pp. 66–73.

[16] J. Wang *et al.*, "A survey on driver behavior analysis from in-vehicle cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 186–10 209, 2022.

[17] T. Billah *et al.*, "Recognizing distractions for assistive driving by tracking body parts," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 1048–1062, 2019.

[18] D. Liu *et al.*, "Toward extremely lightweight distracted driver recognition with distillation-based neural architecture search and knowledge transfer," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 764–777, 2023.

[19] F. Vicente *et al.*, "Driver gaze tracking and eyes off the road detection system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2014–2027, 2015.

[20] A. Misra *et al.*, "Detection of driver cognitive distraction using machine learning methods," *IEEE Access*, vol. 11, pp. 18 000–18 012, 2023.

[21] I. McLoughlin, *Applied Speech and Audio Processing*. Cambridge University Press, 2009.

[22] Z. Wang *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[23] A. Abraham *et al.*, *Intelligent Systems Design and Applications: 19th International Conference on Intelligent Systems Design and Applications (ISDA 2019) held December 3-5, 2019*, ser. Advances in Intelligent Systems and Computing. Springer International Publishing, 2020.

[24] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[25] M. Sandler *et al.*, "MobileNetV2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018.

[26] B. Zoph *et al.*, "Learning transferable architectures for scalable image recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018.

[27] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114.