Unsafe maneuver classification from dashcam video and GPS/IMU sensors using Spatio-Temporal Attention Selector

Matteo Simoncini*[12], Douglas Coimbra de Andrade[1], Leonardo Taccari[1], Samuele Salti[3], Luca Kubin†, Fabio Schoen[3], Francesco Sambo[1]

***Abstract*—In this paper, we propose a novel deep learning architecture to classify unsafe driving maneuvers from dashcam and IMU data. Such architecture processes the output of an object detection algorithm in combination with raw video frames and GPS/IMU data. At the core of the architecture there is a novel Spatio-Temporal Attention Selector (STAS) module, which (1) extracts features describing the evolution of each object in the scene over time and (2) leverages multi-head dot product attention to select the relevant ones, *i.e.*, the dangerous ones or the ones in danger, to perform classification. We also introduce a simple but effective methodology to increase the benefit of fine-tuning the backbone network. Our method is shown to achieve higher performance than other approaches in the literature applying attention over single frames.**

***Index Terms*—Unsafe maneuver classification, road scene understanding, dashcam, GPS, IMU, deep learning, attention, XAI**

I. Introduction

THE unsafe maneuver classification task was firstly introduced in [1], with the aim of classifying a safety-critical event (*i.e.*, crashes and near crashes) recorded from a dashcam, according to the unsafe maneuver leading to the dangerous situation. This task is interesting for multiple reasons. First, while road deaths are a major global burden, vehicle safety systems have shown to actively contribute to the reductions of the number of deaths and serious injuries [2]. Thus, research in this field aiming at getting a better understanding of safety-critical events is crucial to mitigate the problem. Second, in contrast with other works in the literature looking at a single aspect of these events, this research considers a broader set of maneuvers, performed both by the ego-vehicle and by other vehicles, multiple-vehicle maneuvers or single-vehicle ones (*e.g.*, loss of vehicle control, vehicle over the edge of the road). These maneuvers were grouped into ten classes, grounded on a Naturalistic Driving Study (NDS) dataset [3] and, thus, are representative of the real distribution of unsafe events. Specifically, the ten classes of unsafe maneuvers introduced in [1] and considered in this work are reported in Table I.

TABLE I: List of unsafe maneuvers

Class	Description
`SL`	*Subject lane change*. The subject performs an improper lane change, potentially from an adjacent lane, an acceleration or deceleration lane or from a parallel parking spot, drawing dangerously close to another vehicle in another lane, being it in front of the vehicle, behind the vehicle and/or with potential sideswipe threat. Alternatively, the subject invaded the lane of a car coming in the opposite direction.
`ST`	*Subject turn*. The subject performs an improper turn, potentially at an intersection, from a driveway or from a perpendicular parking spot, invading the lane or space of another vehicle proceeding in the same or opposite direction of the vehicle.
`NSL`	*Non-subject lane change*. As `SL` but with another vehicle being the one performing the unsafe maneuver.
`NST`	*Non-subject turn*. As `ST` but with another vehicle being the one performing the unsafe maneuver.
`SB`	*Subject brakes*. The subject vehicle brakes to avoid the collision with another vehicle in the same lane and going in the same direction, potentially performing an evasive maneuver.
`SOE`	*Subject over edge*. The subject vehicle runs over the edge of the road or collides with road boundaries.
`SLC`	*Subject lost control*. The subject vehicle loses control due road condition, excessive speed or other causes.
`SO`	*Subject other maneuver*. Other unsafe maneuvers performed by the subject vehicle.
`NSO`	*Non-subject other maneuver*. As `SO` but with another vehicle being the one performing the unsafe maneuver.
`O`	*Other*. Collision or near collision with animals, pedestrian, pedal-cyclist or other objects.

In this paper, we propose an extension of the approach of [1], where the authors considered video information and data from a GPS/IMU module (acceleration, angular velocity and speed) acquired from a dashcam mounted inside the vehicle and proposed a two-stream architecture based on convolutions. We propose two major improvements: first, we integrate the output of an object detection algorithm in the pipeline, to provide the network with explicit information about the entities on the road and let it learn high-level representations of the interactions between them. Second, we leverage attention [4], [5] to let the network focus on the relevant objects in the scene, *i.e.*, the one involved in the unsafe maneuver, and on the relevant temporal segments. While the usage of attention is motivated by the extremely good results it has obtained in various fields of the scientific literature [4]–[14], it also has an interesting by-product: attention forces the network to *focus* on a particular portion of the input and to have the resulting output that mainly depends on that portion. Thus, it provides an explanation on the reason behind a given prediction by learning features with an associated semantic meaning [15] that allow the network to intrinsically explain itself [16]. This

[1] Verizon Connect Research, Florence, Italy
[2] DINFO, Università degli Studi di Firenze, Italy
[3] DISI, University of Bologna, Italy
†Work done while in Verizon Connect Research
*Email: `matteo.simoncini@unifi.it`

Digital Object Identifier 10.1109/TITS.2022.3142672
1558-0016

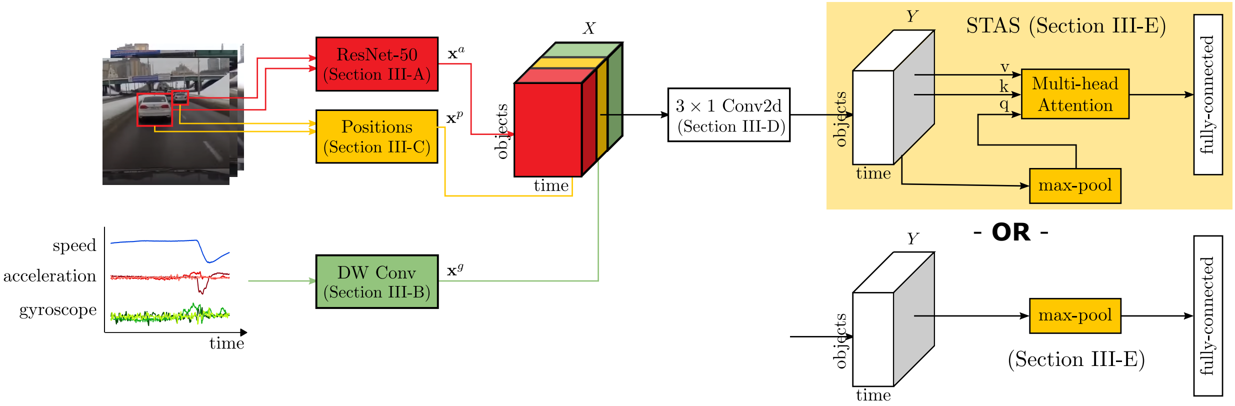


Fig. 1: An overview of the proposed architecture. In red, the appearance features extracted from the object detector output (Section III-A), in yellow the features relative to the boxes positions (Section III-C), in green the GPS/IMU features (Section III-B), in white the per-object feature extracted (Section III-D). Such features are either pooled with a max pooling layer or fed to the STAS module, composed of a max pooling layer as query vector and multi-head attention (Section III-E).

aspect in particular is known in the scientific literature as eXplainable Artificial Intelligent (XAI) [15]–[17] and it is of crucial importance when the output of the model is presented to or evaluated by a human (in the so-called human-agent systems), where a motivation behind the prediction might be necessary to convince of the correctness of the prediction or to understand the reasons behind a misclassification [18].

In the road-safety domain, and, specifically, to address the accident anticipation problem, previous works in the literature attempted to integrate attention with the output of an object detection algorithm, using Dynamic Spatial Attention (DSA) [7], [8]. Such module applies attention to all the detected objects of a single frame, in order to have the network focus on the most relevant object at a given moment in time, and then uses a recurrent module to extract the temporal dependencies between the learned features. In contrast, our aim is to use attention to select the most relevant object and the most relevant temporal segment to correctly classify the maneuver, by explicitly extracting features describing the evolution of each object in a given set of frames and then selecting the most relevant one to perform the prediction. We refer to this approach as Spatio-Temporal Attention Selector (STAS).

Moreover, and as an alternative to the use of the attention mechanism, we propose to leverage a max-pooling layer to extract relevant information from the object features. While such layer is more opaque than attention, *i.e.*, it is harder to understand the reasons behind a prediction, it showed slightly superior performance compared to the attention-based ones in our experiments, suggesting that model explainability comes with a cost. To summarize, the key contributions of this paper are the following:

- We propose an architecture that combines the objects appearance and positions from the video and the GPS/IMU streams to tackle the unsafe maneuver classification problem from dashcam data, that includes maneuvers by both the ego vehicle and other vehicles in the scene.
- We introduce the STAS module, to allow the network to focus on the relevant objects (*i.e.*, the dangerous or in-danger) in the scene and on the relevant frames of the input video to address the classification task, as well as providing explanations of the network decisions.
- We introduce several methodological novelties and technical improvements over similar works in the relevant literature, such as the usage of an RoI pooling layer, widely used in the CV community, a novel backbone fine-tuning strategy and a novel way of extracting GPS/IMU features, that are beneficial for the network efficiency and overall performance.
- We perform a broad experimentation phase, showing the superiority of the proposed approach over baseline methods, and provide an in-depth ablation study.

II. Related Works

To the best of our knowledge, the only paper addressing unsafe maneuver classification is [1]. In this section we discuss the most relevant papers in the literature, focusing on closely related tasks using object detection and the usage of the attention mechanism in the computer vision literature.

Accident / Driving maneuver detection In the context of accident detection, [19] addressed the problem in an unsupervised way, first by detecting the object in the scene for each frame and then by training a network to predict the object positions in the next frame. A misalignment between the predicted position and the actual one is considered a safety-critical event. [20] designed a system based on object detection and Random Forest to classify safety-critical events into crashes, near-crashes and safe events. Similarly, [21] addressed forward collision warning by first detecting the forward vehicle and estimating the distance from the ego-vehicle, and then by combining this information with an abnormal driver behaviour detector. These approaches, however, does not consider GPS/IMU features, that have shown to give a great boost in performance [1]. In the context of ego-vehicle maneuver detection, [22] considered both video and GPS/IMU as inputs. Video frames were fed to a pre-trained VGG network while handcrafted features were extracted from the GPS/IMU data. The two streams were then fed to an LSTM model. The authors proposed to process only frames sampled on a uniform spatial basis (*i.e.* a frame per meter) instead of a temporal one, which they proved to be beneficial for ego-maneuver detection. With this approach, however, maneuvers performed by other vehicles while the subject is not moving cannot be detected. [23] proposed a method to classify subject maneuvers from videos, using a pretrained model to extract depth from video and the camera motion information (and, thus, the trajectory performed by the subject). Then, Dynamic Time Warping is used to perform the classification. However, in our case, we're not interested in detecting the subject maneuver alone, but rather in classifying the maneuver with respect to its context. In the context of other vehicle maneuvers detection, [24] designed a framework based on the detection of road scene objects and applied tracking and motion detection.

Attention mechanism. The attention mechanism was first introduced in [4] in the Neural Machine Translation (NMT) literature, with the aim of giving the decoder the ability to dynamically *focus* on parts of the input sentence that are relevant to predict a target word, instead of being forced to encode the source sentence into a fixed-length vector. This is achieved by projecting each word in an embedding space and computing a similarity measure between words, represented as a fully-connected operation. [5] generalized the mechanism proposed in [4] (also referred to as *soft attention*) by computing the similarity measure using the dot product operation and by computing a set of embedding and attention operations, introducing the *multi-head dot product attention*. Later on, the idea of letting the network focus on a specific part of the data was applied in other fields of research, *e.g.*, computer vision, mainly in two ways. [9] proposed to use the attention mechanism in an encoder-decoder architecture for video captioning, having the network focus on features extracted from a specific part of the image. Other works generalized the approach by considering the features from the output of an object detector [10], [11]. More recently, the attention mechanism has been used to increase network representation capability, focusing on important features and suppressing unnecessary ones. [13] introduced the Convolutional Block Attention Module (CBAM), composed of a channel attention module and a spatial attention one. Starting from a 2D feature tensor, the first module uses a combination of max-pooling and average-pooling operations over the spatial dimension, to get a descriptor that is applied back to the input tensor via element wise multiplication; the second one performs a similar operation over the channel dimension. Similarly, [14] proposed the Squeeze-and-Excitation (SE) block, that computes a pooling operation over the spatial dimension (squeeze), applies a channel-wise feed-forward operation to learn inter-channel dependencies and uses the output to scale the input tensor (excitation). Finally, attention weights have been also used to achieve model explainability [9], [25].

Accident anticipation. The usage of attention for accident anticipation from dashcam videos was first proposed in [7]. They used an object detection algorithm to extract the objects in the scene and computed the features of a pre-trained convolutional neural network on Places-365 [26] on their locations. Then, they introduced the DSA system in combination with an LSTM and a custom loss to predict the car crash as early as possible. Performance was evaluated on the novel DAD dataset that, however, is mostly composed of accidents not involving the ego-vehicle. [8] improved the previous architecture by using Quasi-Recurrent Neural Network (QRNN) and an adaptive custom loss. Also, they used a fine-tuned backbone on per-frame risk factor classification and background classification task. They evaluated their performance on the broader NIDB dataset, which is mostly composed of ego-vehicle accidents.

III. Methodology

In order to address the unsafe maneuver classification task, we propose an architecture that leverages the video information and the GPS/IMU data. We first use an object detector to extract the position and types of all the objects from each frame and extract appearance and positional features for each object, as described in Section III-A, and use a tracking algorithm to link the same real object in different frames, described in Section III-C. Then, the object features are enriched with the aligned GPS/IMU features, described in Section III-B. We apply a set of convolutional operations to each object, in order to extract high-level descriptors for each of them and to reduce the temporal dimensionality of the data, described in Section III-D. Finally, we consider two setups, either performing a simple max-pooling over the objects and temporal segments or to use the STAS module, which leverages a multi-head attention layer to select the most relevant object and temporal segment, described in Section III-E.

A. Object detection and feature pooling

We use a Faster-RCNN [27] with ResNet-101 [28] backbone as object detector on each frame, and extracted object positions and classes. At this point, we would like to associate an appearance vector to each object. One could consider the output of the detector backbone or consider other backbones, that could be pre-trained on other tasks [1], [7], [22] or fine-tuned [8]. In this second case, it is common to extract the crop of the image for each object and run it through the backbone [7], [8]. This might result, however, in prohibitive inference times. For instance, to compute the features of a single frame with ten objects detected, one would need to compute ten backbone forward passes. During training, it is possible to speed up the process by storing locally the backbone outputs. However, when there are a lot of detections for each frame, this might also results in high storage costs. In contrast,inspired by [12] and as widely shown effective in the Computer Vision community, we are using a RoI pooling layer [29] that allow us to compute the object features with a single backbone forward pass both during training and inference.

B. GPS/IMU module

The data coming from the GPS/IMU sensors is used as input to the GPS/IMU module, as shown in Figure 1. As in [1], such module has two roles: to align the different sensors sampling rates and the video timestamps and to extract high-level features from the raw sensors. We first resample the signals via interpolation, so that they have the same number of samples, and this number is a multiple $\theta = 3$ of the number

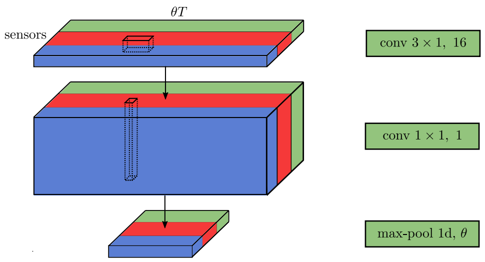


Fig. 2: Schematic representation of the GPS/IMU module leveraging depthwise separable convolution applied to each sensor. Different colors represent different sensors.

of video timestamps. Then, we apply a set of convolutional operations and, finally, we use a max pooling operation of size θ, to align the sensors and the video stream.

In contrast with [1], though, we do not use 1D convolutions over the temporal dimension, but rather apply the same convolutional operation to process each signal independently, with the idea to learn filters to be applied to a generic signal and to extract features describing its temporal evolution, preserving the individual signal semantic meaning. To accomplish this, we use 2D depthwise separable convolutions [30], [31] as it was proposed in the audio processing literature [6], [32]. Thus, starting from an input tensor of shape $\theta T \times s$, with T number of frames and s total number of sensor signals, we fist add an extra dimension, changing the shape of the input tensor to $1 \times \theta T \times s$, with the first dimension representing the number of channels. Then, we apply a 2D convolution with kernel size $k = 3 \times 1$ and with $f^s = 16$ output channels (*i.e.*, filters). The output tensor, using padding over the temporal dimension to maintain the same spatial extent, has shape $f^s \times \theta T \times s$ Then, we apply a second 2D convolution with kernel size $k = 1 \times 1$ and with 1 output channel, that get then removed to go back to a tensor of shape $\theta T \times s$. While one could stack multiple of these operations, to learn richer information as in [30], [31], in this paper we are using a single pair of convolutions, so that each element of the output sensor retains the temporal receptive field, *i.e.*, is computed only on the sensor information around the single frame. A schema of the GPS/IMU module is reported in Figure 2.

C. Object preprocessing

In Section III-A we showed how to extract a set of positions and appearance feature from each object of the scene and for each frame. Let $\mathbf{o_t} = \{o_{t,1}, \ldots, o_{t,N_t}\}$ be the set of objects detected in frame $t \in \{1 \ldots T\}$, with N_t the total number of objects detected in frame t and $o_{t,i} = (a_{t,i}, p_{t,i})$ where $a_{t,i}$ and $p_{t,i}$ are respectively the appearance features and position of the i-th object detected in frame t. Without loss of generality, let us assume $o_{t,i}$ to be relative to the same real object (*e.g.*, the same vehicle) for each frame t, with $(a_{t,i}, p_{t,i})$ vectors of zeros if the i-th object is not present or not detected in the frame t. Also, instead of considering the maximum number of detections N_t for each frame t, we can think of having a fixed number of detection N_{objs} for each frame, considering as zeros the extra objects for each videos and discarding the exceeding ones. In this setup, we can express the detected object as a matrix O of size $T \times N_{objs}$ with $o_{t,i} = (a_{t,i}, p_{t,i})$. As a heuristic to decide which objects to keep among the detected ones, we consider the top N_{objs} objects according to detection total volume, *i.e.*, sum of the detected area for each object $o_{t,i}$ for each frame t, and we kept the N_{objs} objects with the largest volumes. This simple rule turned out effective in our experiments, as the relevant objects are close to the subject vehicle for a large number of frames and, thus, are among the objects with the largest volume.

In order to build the matrix O, it is necessary to link the same real-world object in two consecutive frames, $o_{t,i}$ and $o_{t+1,i}$, using, for instance, a tracking algorithm on the detected objects. In this paper, we utilize a greedy tracking algorithm that uses only object positions, detections confidence and class information, that have shown to be effective in [20]. In particular, starting from frame $t = 0$, we assign a unique tracking *id* to each object $o_{t,i}$ with confidence $c_{t,i} \geq 0.6$. Then, iteratively for each following frame t:

- we compute the matching between the detections in frame $t-1$ and the ones in frame t and assign to each matched object the same *id*;
- we assign a new unique *id* to all unmatched objects in frame t with confidence $c_{t,i} \geq 0.6$;
- we discard all the remaining detections in frame t.

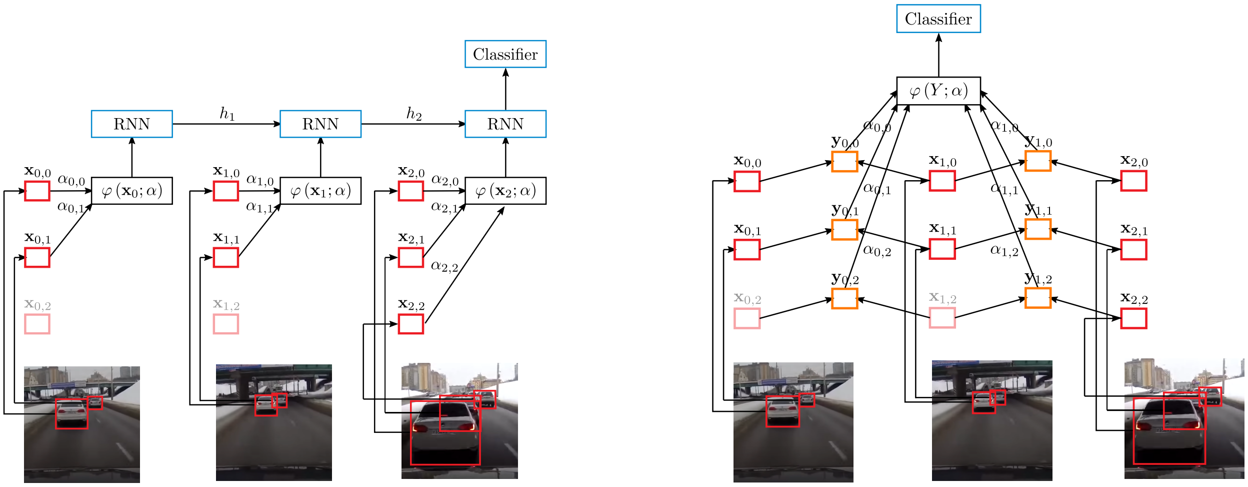
The matching is again a greedy algorithm that first generates a set of candidate detection pairs of the same class and with ≥ 0.2, then iterates over such set assigning a matching for the pairs with the highest IOU values, removing the matched detections from the candidate set and iterating, so that each object in frame $t-1$ could be a match for at most one object in frame t (maximum bipartite matching).

D. Object feature extraction

Starting from matrix O and in order to perform the classification, we build a matrix X of shape $T \times N_{objs}$, where each element $\mathbf{x}_{t,i}$ is the concatenation of three feature vectors

$$\mathbf{x}_{t,i} = \left[\ \mathbf{x}^a_{t,i} \mid \mathbf{x}^p_{t,i} \mid \mathbf{x}^g_{t,i}\ \right]. \tag{1}$$

The feature vector $\mathbf{x}^a_{t,i}$ is relative to the appearance of the object and is obtained feeding the output of the RoI pooling layer of the i-th object and of frame t to a bottlenck layer, *i.e.*, a linear layer followed by a 1D batch normalization layer and a ReLU activation, in order to reduce the dimensionality of the data and make it comparable with the other stream, as it was proven beneficial in [1]. The feature vector $\mathbf{x}^p_{t,i}$ is relative to the position and class of the detection. It contains position of the top left corner of the box, normalized in $[0, 1]$, the normalized width and height of the box, the confidence of the detection and a one-hot encoded vector indicating the class of the object. Finally, $\mathbf{x}^g_{t,i}$ is the output of the GPS/IMU module for frame t, as described in Section III-B, replicated for each object. In addition, we add a flag indicating whether the box i has been detected in frame t or not. In the latter case, $\mathbf{x}^a_{t,i}$, $\mathbf{x}^p_{t,i}$ and $\mathbf{x}^g_{t,i}$ are zeros. Finally, we add an extra, always present *dummy* object to the object matrix, with the GPS/IMU information $\mathbf{x}^g_{t,i}$ only. Such object will cope with the fact that a video might not have objects other than the subject vehicle equipped with the dashcam, and will forward GPS/IMU information to the final classifier.



(a) The DSA module as proposed in [7]. The feature vector $\mathbf{x}_{t,i}$ are extracted from each frame (in red) and the attention mechanism is used to produce the attention map $\varphi(\mathbf{x}_i;\alpha)$ and the attention weights $\alpha_{t,i}$. Then, the map is fed to a RNN.

(b) The STAS module proposed in this paper. The feature vector $\mathbf{x}_{t,i}$ are extracted for each object and for each frame (in red). Then, multiple feature vectors of the same object are aggregated, into a feature vector $\mathbf{y}_{\tilde{t},i}$ representing the evolution of the object (in orange, in the example two objects were considered). Finally, the attention map $\varphi(Y;\alpha)$ and the attention weights $\alpha_{\tilde{t},i}$ are computed on each object and on each temporal segment.

Fig. 3: Comparision between the DSA module and the STAS module. The figures are displaying single-head attention.

In order to extract features that link together the objects in two consecutive frames $\mathbf{x}_{t,i}$ and $\mathbf{x}_{t+1,i}$, some works in the literature propose to use a DSA module [7], [8]. Such module uses the attention mechanism to select the relevant object at each time step that are then feed to a recurrent layer (*e.g.* an LSTM), as showed in Figure 3a. The latter two elements are, thus, ideally performing the connection between the features $\mathbf{x}_{t,i}$ and $\mathbf{x}_{t+1,i}$, only if the attention weights $\alpha_{t,i}$ and $\alpha_{t+1,i}$ on the same object are high. Even if this is the case, such relation is exploited in the very last stages of the network, *i.e.*, in the recurrent connection and, thus, it might be difficult to leverage. However, it is worth noticing that such approach does not require explicit associations, *i.e.*, tracking, between objects $o_{t,i}$ and $o_{t+1,i}$, as the DSA module considers all the objects of a single frame in isolation and the output of the attention layer is independent from the ordering of the inputs.

In contrast, we believe that extracting object connections in the early stages of the architecture is crucial, as it allows the network to extract features that consider the evolution over time of a single object, *e.g.* an object getting bigger and bigger or an object moving from left to right. For this reason, after building the matrix X and the feature vectors $\mathbf{x}_{t,i}$, we can extract features $\mathbf{y}_{\tilde{t},i}$ related to the evolution of the objects in the scene. This is done by applying the same set of convolutional operations to each object. In order to accomplish this in an efficient way, as reported in Table II, we used 2D convolutional operations of size 3×1, thus insisting on three frames and on a single object. We stacked four convolutions with a growing number of filters $f = 64, 128, 256, 512$, always followed by 2D batch normalization, ReLU activation and 2D max-pooling operations of size 2×1, in order to reduce the temporal dimension while increasing the number of filters. The result is a matrix Y of shape $\widetilde{T} \times N_{objs}$ where each element $\mathbf{y}_{\tilde{t},i}$ represents the evolution of an object i in a temporal segment $\tilde{t}$, with $\tilde{t} \in 1, \ldots, \widetilde{T}$ indices of the reduced temporal dimension.

E. Object selection module

We would like to select the most relevant feature vectors $\mathbf{y}_{\tilde{t},i}$, to perform the classification. In this paper, we propose to use a multi-head attention layer [5]. Such layer, starting from three set of vectors, namely keys, queries and values, performs a weighted sum of the values, where the weight assigned to each value is computed by a similarity function between the query and the corresponding key. We are considering the set of vectors $\{\mathbf{y}_{\tilde{t},i}\}$ as keys and values and the 2D global max-pooling of such vectors as query. The rationale behind this choice is that, ideally, the network will generate features that have higher activations in correspondence to relevant objects. Then, by pooling along the channel axis we would obtain a vector that is representative of the relevant object activations: this has been shown to be effective in highlighting informative regions [13] and to be good for the classification task [1]. By using such vector as a query, we are training the network so that the attention weights are higher in correspondence to the object features that are the most similar to the pooled vector, thus, where the object activations are higher and to the relevant objects. Formally, we are defining the key K, query Q and value V matrices as

$$
\begin{gathered}
Q = \Big[MaxPool(\mathbf{y}_{0,0}, \ldots, \mathbf{y}_{\widetilde{T},N_{objs}}) \Big] \\
K = \begin{bmatrix} \mathbf{y}_{0,0} \\ \cdots \\ \mathbf{y}_{\widetilde{T},N_{objs}} \end{bmatrix} \quad V = \begin{bmatrix} \mathbf{y}_{0,0} \\ \cdots \\ \mathbf{y}_{\widetilde{T},N_{objs}} \end{bmatrix}
\end{gathered} \tag{2}
$$

Then, we use the multi-head attention layer as described in [5], with the projection size $d_{model} = 64$ and $h = 8$ number of heads. For each head h, such layer first projects the input vectors a reduced embedding space of size d_{model}, then computes a set of attention weights $\{\alpha^h_{\tilde{t},i}\}$, *i.e.*, a similarity

TABLE II: Summarization of object feature extraction module. The tensor dimensions in the output size column are respectively features, objects and frames, with $|\mathbf{x}^a_{t,i}| = 128$, $|\mathbf{x}^g_{t,i}| = 7$, $|\mathbf{x}^p_{t,i}| = 20$, $T = 135$ and with $N_{objs} = 25$.

Layer	**Output Size**	**Structure**
Input $(\mathbf{x}_{t,i})$	$155 \times 26 \times 135$	–
Conv2d	$64 \times 26 \times 135$	1×3, 64, pad 0×1
BatchNorm2d	$64 \times 26 \times 135$	–
MaxPool2d	$64 \times 26 \times 67$	1×2
Conv2d	$128 \times 26 \times 67$	1×3, 128, pad 0×1
BatchNorm2d	$128 \times 26 \times 67$	–
MaxPool2d	$128 \times 26 \times 33$	1×2
Conv2d	$256 \times 26 \times 33$	1×3, 256, pad 0×1
BatchNorm2d	$256 \times 26 \times 33$	–
MaxPool2d	$256 \times 26 \times 16$	1×2
Conv2d	$512 \times 26 \times 16$	1×3, 512, pad 0×1
BatchNorm2d	$512 \times 26 \times 16$	–
MaxPool2d $(\mathbf{y}_{t,i})$	$512 \times 26 \times 8$	1×2
MultiHeadAttention	512	heads $= 8$, dropout $= 0.1$
Fully-connected	10	–

measure based on dot product between each key and the query, and use them to perform a weighted combination over the values. The resulting vectors $\varphi^h(Y; \alpha^h)$ for each head are then combined together using concatenation and a linear operation into a single vector $\varphi(Y; \alpha)$ that is used for the classification, while the weights α^h are averaged into a single vector α that is used for explainability. The usage of the max pooling as query vector has been exploited in other field and application such as [6], [13], [14], but never to perform the selection of relevant objects as proposed in the paper, thus, we are naming this layer Spatio-Temporal Attention Selector (STAS).

As an alternative, we propose to use such max-pooled vector directly for the classification, as in [1]. Note that, by doing so, the features used for the classification are pooled from any feature vector of Y instead of being forced, by the attention layer, to belong mostly to a single feature vector. Relaxing such constraint might improve performance at the expense of explainability.

F. Backbone fine-tuning

While using a pretrained backbone on Places-365 [26] to extract the object appearance feature showed good results in similar task [1], [7], [22], to fine-tune the backbone would almost certainly improve the performance. However, when working with videos, it is hard to fine-tune the backbone directly on the task, especially if the video are long or have an high *fps*, as it generally requires larger datasets and more GPU memory than available. Thus, many works propose to fine-tune their backbone on auxiliary tasks. In [8], the authors propose to use a per-frame risk-factor classification. In [25], the authors trains their backbone to predict the vehicle steering and acceleration. In this paper, we fine-tune the backbone on the same unsafe maneuver classification task but with a smaller version of the video with lower frame-rate and duration.

We considered video segments of 32 frames at 5 *fps*, *i.e.* 6.4 seconds, randomly choosing such segments under the constraint that at least 75% of the video should be contained in the segment or all the frames should be event frames. This approach was possible only knowing the beginning and the end of the safety-critical event in each video. Then, we used an architecture like the one in Figure 4. We first fed the video to the backbone and applied a 2D global average pooling layer. After these operations, the output tensor has shape $F \times C$, with $F = 32$ numbers of frames and C backbone output channels. In our specific case, we considered ResNet-50, having $C = 2048$. Then, in line with the residual structure of the backbone, we considered $N = 4$ 1D adaptations of the Bottleneck layers proposed in [28] of size C, each of which applies a point-wise 1D convolution with $C/4 = 512$ filters, a 1D convolution of size 3 and $C/4 = 512$ filters and point-wise 1D convolution with $C = 2048$ filters, warping everything with a residual connection between the input of the three convolutions and the output. In addition to this, we would want to reduce gradually the temporal dimension. Thus, the first point-wise operation of each block has a stride $s = 2$ and each residual connection is equipped with an additional point-wise operation, to adjust the number of channels accordingly. Finally, a 1D global max pooling layer is applied over the reduced temporal dimension and the resulting vector is used for classification.

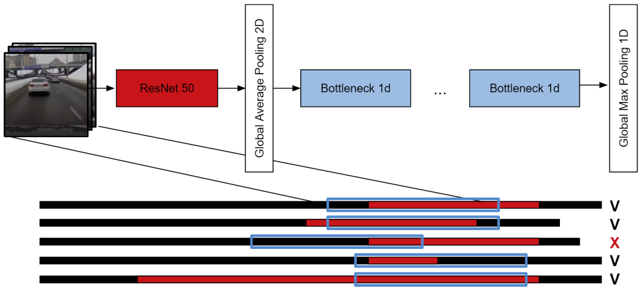


Fig. 4: The architecture used to fine-tune the backbone. Below, a set of examples used for fine-tuning, satisfying (V) or not satisfying (X) the constraint. In blue, the selected segment, in red, the safety-critical event, in black the full video.

IV. Experimental results

A. Dataset and implementation details

All the experiments were conducted on the SHRP2 NDS dataset [3], a collection of more than 8800 safety-critical events, gathered by more than 3300 drivers between 2010 and 2013. The dataset is composed of videos at 15 fps with resolution 480 × 356, while the GPS-related sensors have a sampling frequency of 1 Hz and the IMU of 10 Hz. We preprocessed the dataset as described in [1] and used the same annotations. To cope with outliers and sensor drift, we first rescaled each sensor to have zero mean in each example and then used a robust scaling strategy, scaling the $25th$ and $75th$ percentiles of each sensor in the range $[-0.5, 0.5]$. The samples with missing GPS data were discarded. For all the experiments, as in [1], we re-sampled the videos to 356×256 and considered a random crop of 314×224 during training and a central crop of the same size during testing, adjusting the extracted boxes accordingly. Also, as data augmentation, we used color jittering for the videos and colored gaussian noise for the GPS/IMU data. Finally, we run all the experiments in the *clip around the event* setup, described in [1]. In this

TABLE III: Per-class results and inference time of the proposed models, without appearance (top row) and with appearance (bottom row). † the inference time of the object detection algorithm is not included.

Model	**Inference**	**Average Precision (AP)** SL	ST	NSL	NST	SB	SOE	SLC	SO	NSO	O	**mAP**
Two Stream (Sensors only) [1]	11 ms	0.35	0.55	0.57	0.30	0.80	**0.94**	0.59	0.65	0.59	0.23	0.556
DSA + LSTM [7], [8]	121 ms †	0.15	0.52	0.62	0.43	0.84	0.90	0.60	0.46	0.37	0.67	0.555
Ours + STAS (w/o appearance)	20 ms †	0.33	0.54	0.75	0.59	0.90	0.89	0.48	0.48	**0.68**	0.68	0.633
Ours + MaxPool (w/o appearance)	20 ms †	**0.38**	0.49	0.81	0.66	0.92	0.91	0.64	0.42	0.60	0.61	0.646
Two-Stream [1]	917 ms	0.28	0.54	0.61	0.60	0.91	0.92	0.60	**0.68**	0.62	0.59	0.635
Two-Stream (Proposed fine-tuning)	917 ms	**0.38**	0.49	0.81	0.66	0.92	0.91	0.64	0.42	0.60	0.61	0.690
Ours + STAS (Proposed fine-tuning)	952 ms †	0.29	0.55	0.78	0.64	**0.96**	0.91	**0.82**	0.61	0.64	0.68	0.700
Ours + MaxPool (Proposed fine-tuning)	952 ms †	0.34	**0.72**	**0.87**	**0.78**	0.95	0.89	0.73	0.45	0.66	**0.72**	**0.712**

scenario, only a relatively small clip containing mostly only the safety-critical event is selected, to highlight the capability of the model to distinguish the various maneuvers. Starting from the event start and event end frames, we considered the $2/3$ of the event as reference and selected 45 frames after and 90 frames before. During training, we added a random offset of ± 10 frames to the reference frame. This gave us a fixed input segment with a total of 135 frames for each event. In a practical context, one may think to obtain such segment running a coarse event detection algorithm, *e.g.*, a trigger based on IMU data, and use the sensor peak as reference point.

All the experiments were conducted using Adam optimizer [33] with weight decay of 10^{-4}. We used an initial learning rate of 10^{-3}, decreased by a factor of 0.5 after 50, 70, 90 and 110 epochs. This applies to all the experiments, but the one with the LSTM, which has a longer convergence time, where we decreased the learning rate after 100, 150, 200 and 250 epochs. We also clipped the norm of the gradients to 1.0 to prevent exploding gradients. As for the backbone fine-tuning, we initialized the backbone with the pre-trained weights of Places-365 and the 1D convolutions with random weights. We used Adam with a weight decay of 10^{-4} and a learning rate of 10^{-3}, decreased to 10^{-4} and 10^{-5} after 50 and 70 epochs.

B. Comparison with state-of-the-art

We consider two variants of the proposed approach.

- *Ours + STAS* is the architecture described in Section III, with the extracted features $\mathbf{y}_{\tilde{t},i}$ fed to the STAS module.
- *Ours + MaxPool* is the same as above, but with the extracted features $\mathbf{y}_{\tilde{t},i}$ fed to a global max pooling layer.

We compare the proposed approaches with

- *Two-Stream*, *i.e.*, the two-stream architecture as proposed in [1], using both the features extracted from the full frame and the GPS/IMU data.
- *DSA + LSTM*, *i.e.*, the porting to the unsafe maneuver classification problems of DSA-based architectures proposed for accident anticipation [7], [8]. In particular, the structure of the network is the same proposed in this paper up to the extraction of the object features $\mathbf{x}_{t,i}$. Then, we compute the attention map on each object feature per frame and fed the result to a LSTM. In contrast with their approach, we use the multi-head dot product attention, since it has shown superior results compared to soft attention [5]. Furthermore, we used $\mathbf{x}_{t,i}$ as object features instead of the one proposed in the original papers, which were describing similar aspects but in a slightly different way, *e.g.* using IDT to describe object motion, in order to be able to assess the impact of the different attention modules under similar conditions. Finally, we used standard cross entropy loss instead of the accident anticipation losses proposed in the papers.

As in [1], we used the mean average precision (*mAP*) for evaluation, that is equivalent to computing the mean area under the precision-recall curve for each class. This metric accounts for both precision and recall and is robust to class imbalance.

The comparison with the state-of-the-art is reported in Table III, along with the inference time and per-class average precision (AP). In the top row, we report several results not considering the appearance features, *i.e.*, having $\mathbf{x}_{t,i} = \left[\mathbf{x}^{p}_{t,i} \mid \mathbf{x}^{g}_{t,i}\right]$: this is a valuable set-up to test, since it can be a lightweight component to add to a pipeline processing dashcam videos and already performing object detection. First, we can observe how the proposed approaches outperform the baselines we evaluate and, in particular, the DSA-based approaches from [7], [8]. It is worth highlighting, however, that the sensor only version from [1] is not using the object positional features. In the bottom row of Table III, instead, we report the results considering also the appearance features. Again, we can observe that the proposed approaches outperforms the Two Stream baseline, either considering the original backbone or the fine-tuned version proposed in this paper.

C. Ablation study

The majority of the experiments were made without using the appearance features and are reported in Table IV. First, we can see how the usage of the depthwise separable convolutions (DW), without mixing sensor information in the early stages, significantly improves performance over the standard 1D convolutions (C1D): mAP is 0.576 when using C1D (first row) and it raises to 0.633 (+0.057) by switching to DW (third row). This might suggest that mixing up sensors in the early stages would promote high informative sensors (*e.g.* the flow-direction accelerometer) and penalize low-informative ones, that could, however, be crucial in distinguishing some corner cases. Second, we evaluated the effect of the possible values of N_{objs} on the final prediction metric. We found the optimal value to be $N_{objs} = 25$, however, the usage of a higher number of objects is not significantly detrimental. As reference, we reported also the results with $N_{obj} = 0$, *i.e.*, considering only the dummy object, to observe the effect of adding the objects information. It is worth to highlight that the

TABLE IV: Tests without using appearance features.

Model	Sensors		N_{objs}	Object selection			mAP
	DW	C1D		DSA	STAS	MaxPool	
Ours + STAS (w/o appearance, C1D)		✓	25		✓		0.576
Ours + STAS (w/o appearance)	✓		0		✓		0.525
Ours + STAS (w/o appearance)	✓		25		✓		**0.633**
Ours + STAS (w/o appearance)	✓		100		✓		0.614
Ours + STAS (w/o appearance)	✓		250		✓		0.602
Ours + MaxPool (w/o appearance)	✓		25			✓	**0.646**
Ours + STAS (w/o appearance, w/o GPS/IMU)			25		✓		0.458
Ours + MaxPool (w/o appearance, w/o GPS/IMU)			25			✓	0.459

TABLE V: Tests using appearance features

Model	Sensors	N_{objs}	Appearance			Object selection		mAP
	DW		Places-365	ImageNet	Fine-tuned	STAS	MaxPool	
Ours + STAS (Places-365)	✓	25	✓			✓		0.656
Ours + STAS (ImageNet)	✓	25		✓		✓		0.657
Ours + STAS (Proposed fine-tuning)	✓	25			✓	✓		0.700
Ours + MaxPool (Proposed fine-tuning)	✓	25			✓		✓	**0.712**
Ours + STAS (Proposed fine-tuning, w/o GPS/IMU)		25			✓	✓		0.476
Ours + MaxPool (Proposed fine-tuning, w/o GPS/IMU)		25			✓		✓	0.506

setup with $N_{objs} = 25$ is outperforming $N_{objs} = 100$ which is outperforming $N_{objs} = 250$: it shows how the heuristic we used to limit the number of object is not discarding the relevant ones. Third, we observe how the usage of the max-pooling layer instead of the STAS module, *i.e.*, not forcing the network to pick a single object, is slightly beneficial for the performance, at the expense of the explainability. It is worth to highlight, however, that the two results are not too distant from each other, as the difference in terms of mAP is small (0.013). Finally, we tested the performance of the model when removing the GPS/IMU features completely, *i.e.*, considering $\mathbf{x}_{t,i} = \mathbf{x}^{p}_{t,i}$, as such data might not be available in some vision-only application scenarios. We observed a severe drop of roughly 0.18 of in the overall *mAP*, showing the importance of such type of data, if available, to tackle the unsafe maneuver classification and confirming the findings of [1].

The experiments in Table V show that, also when considering the appearance features, the architecture with max-pooling layer outperforms the STAS module by a relatively small margin in terms of mAP (0.012), again at the expense of explainability. Furthermore, we can see that the backbone fine-tuning we propose improves mAP significantly in both the Two Stream baseline and the proposed methodology, showing the effectiveness of the method. Moreover, again, the experiments without the GPS/IMU data, *i.e.*, having $\mathbf{x}_{t,i} = \left[\mathbf{x}^{p}_{t,i} \mid \mathbf{x}^{a}_{t,i}\right]$ show that such type of data are crucial to solve this task, as by removing them the performance drops of more than 0.2 in *mAP*. Finally, we can observe how each experiment outperforms the appearance-less counterpart of Table IV, clearly showing the effectiveness of this type of features.

In Table III, we report the per-class AP. Although there is no clear winner, again in the setup without the appearance features the proposed approaches are in general outperforming the baselines in the various classes and the approaches with the appearance are outperforming their counterparts. SB and SOE are the classes with the highest AP, probably because such maneuvers are fairly well distinguishable from the rest and are the majority classes in the dataset. On the other hand, most of the models show a relatively low AP on the minority classes (*i.e.* SL, ST, SO and NSO). Table III also reports the inference times of the various models. We can observe how the inference time of the proposed methods is close to the Two Stream baseline one, thanks to the ROI pooling layer that allows the computation of the objects' appearance features in a single pass. It is worth to highlight, however, that the reported times does not include the object detection algorithm inference time. We used Faster-RCNN with ResNet-101 with an inference time of 72ms per image, thus requiring 9.7s for a full 135 frames video, as we prioritized accuracy over speed in the detection generation process. Nevertheless, requiring object detection as a prerequisite does not add any extra overhead if the proposed architecture is deployed in an application already performing object detection. Finally, please note how the object detection algorithm used at inference time could be, in principle, different, *e.g.*, faster, from the training one, although we did no experimentation in this regard.

V. RESULTS VISUALIZATION

In this Section, we provide qualitative results on the object being selected by the proposed method, since labels on the relevant object in each scene were not available in our dataset.

We consider the *Ours + STAS* model with appearance features and fine-tuned backbone, with the goal of visualizing the object being selected for the classification. This can be accomplished by just looking at the attention weights, since the output is a weighted combination of the object features. The results are the heatmaps in Figures 5, 6, 7, 8, which have been independently normalized into $[0, 1]$ for visualization purposes. Note that each heatmap has the objects on the vertical axis and the reduced temporal dimension on the horizontal axis. Also, the top row is always relative to the dummy object that, for

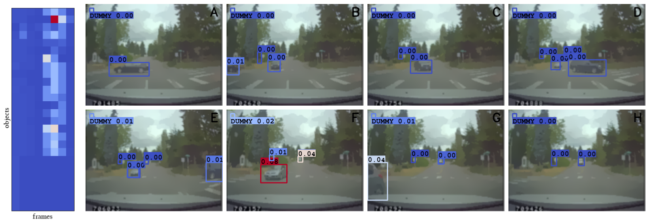


Fig. 5: Subject turn (ST) event

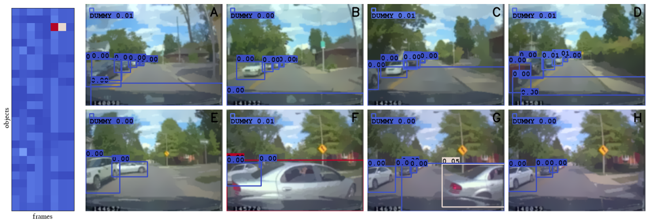


Fig. 6: Non-subject turn (NST) event

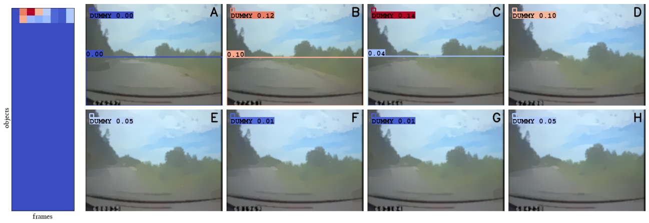


Fig. 7: Subject over edge (SOE) event with no objects

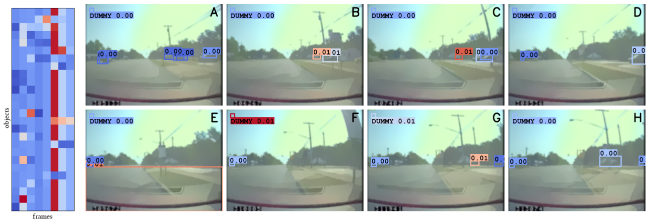


Fig. 8: Subject over edge (SOE) event with not relevant objects

visualization reasons, has been reported also in the top left corner of each frame.

Figure 5 shows a Subject Turn event in which the subject enters an intersection and starts to perform a turning maneuver ignoring the incoming vehicle in the opposite direction, being forced to interrupt the maneuver to not hit the other vehicle. The model is mostly looking at only the incoming vehicle features to perform the prediction in the moment in which the subject is forced to interrupt the maneuver, correctly identifying the relevant object, *i.e.* the one in danger, at the right moment.

Figure 6 shows a Non-subject Turn event in which the subject is travelling on a single lane road approaching an intersection, when another vehicle enters and go through such intersection ignoring the subject, that is forced to hard brake. Again, the model is looking mostly at the right object, *i.e.* the car cutting into the subject vehicle path, at the relevant moment.

Figure 7 shows a Subject Over Edge event in which the subject is going over the edge of the road. There is no object in the scene, but a detection of the subject vehicle hood detected as vehicle by the object detection algorithm. In this particular scenario, the sensors features are used to perform the prediction, extracted mainly from the dummy object. In contrast, Figure 8 shows a Subject Over Edge event with multiple detected vehicles, which are not relevant to perform the classification. The network is assigning a high weight to most of them. We believe that this outcome is due to the fact that the feature vector of each object $\mathbf{x}_{t,i}$ is composed also by the sensors features $\mathbf{x}^{g}_{t,i}$, which are crucial for the prediction of this kind of event. Thus, likely, the resulting $\mathbf{y}_{\tilde{t},i}$ features will be mostly a combination of the sensor feature and be pretty much the same for all the object, *i.e.*, they will be independent from the object appearance and position. As a result, the attention mechanism, which is based on a similarity measure, will find all the features to have an equal contribution. One can extend this concept to a context where the same object is detected multiple times. Then, the appearance and position feature, and thus the resulting $\mathbf{y}_{\tilde{t},i}$ would likely look similar and the STAS module would assign similar weights to each detection, again detecting multiple relevant objects. We found this to be a limitation of the proposed approach, which can however probably easily overcome by assigning low relevance to all the objects in the scenes when the largest attention weights are spread all over the matrix, *i.e.*, when there is no single clear peak in the weights.

VI. Conclusions

We presented an architecture that combines object appearance and position, the video stream and the sensors stream to tackle the unsafe maneuver classification problem. We discussed two variants of this architecture: the first one uses the max-pooled vector of the extracted object features directly for the classification, while the second one leverage the newly introduced STAS module to also identify the relevant object in the video, offering network explainability as an interesting by product. We observe that while the first architecture is better in terms of mAP on the unsafe maneuver classification task, the gap between the two is small, suggesting that explainability comes with a cost on overall performance, but that such cost can be acceptable, if explainability is desired. Furthermore, we presented several methodological and practical novelties over the relevant works in the literature: first, the usage of depthwise convolutions to process sensors data, in order to maintain the semantic of each sensor in deeper stages of the network; second, the usage of an RoI pooling layer to extract object appearance features in an efficient way both during training and inference; third, we introduced the concept of extracting features describing the evolution of a single object over time, by using a tracking algorithm, and using such features for classification; fourth, we presented a new backbone fine-tuning strategy tailored to the unsafe maneuver classification task that could however potentially be extended to other similar domains. Our experiments showed that the proposed approach outperforms the *state-of-the-art* on the unsafe maneuvers classification task and empirically showed the superiority of the STAS module compared to other attention-based methods in the literature on this particular task. Finally, we presented qualitative results on the capability of the network to select the relevant object, highlighting its advantages and limitations.

References

[1] M. Simoncini, D. Coimbra de Andrade, S. Salti, L. Taccari, F. Schoen, and F. Sambo, "Two-stream neural architecture for unsafe maneuvers classification from dashcam videos and gps/imu sensors," in *International Conference on Intelligent Transportation Systems*. IEEE, 2020.

[2] World Health Organization and others, "Global status report on road safety 2018: Summary," World Health Organization, Tech. Rep., 2018.

[3] J. M. Hankey, M. A. Perez, and J. A. McClafferty, "Description of the SHRP 2 naturalistic database and the crash, near-crash, and baseline data sets," Virginia Tech Transportation Institute, Tech. Rep., 2016.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint*, 2014.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[6] D. C. de Andrade, S. Leo, M. L. D. S. Viana, and C. Bernkopf, "A neural attention model for speech command recognition," *arXiv*, 2018.

[7] F.-H. Chan, Y.-T. Chen, Y. Xiang, and M. Sun, "Anticipating accidents in dashcam videos," in *ACCV*. Springer, 2016, pp. 136–153.

[8] T. Suzuki, H. Kataoka, Y. Aoki, and Y. Satoh, "Anticipating traffic accidents with adaptive loss and large-scale incident db," in *CVPR*, 2018.

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015, pp. 2048–2057.

[10] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR*, 2018, pp. 6077–6086.

[11] S. Herdade, A. Kappeler, K. Boakye, and J. Soares, "Image captioning: Transforming objects into words," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 137–11 147.

[12] L. Cultrera, L. Seidenari, F. Becattini, P. Pala, and A. Del Bimbo, "Explaining autonomous driving by learning end-to-end visual attention," in *Proceedings of the IEEE/CVF CVPR Workshops*, 2020, pp. 340–341.

[13] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *ECCV*, 2018, pp. 3–19.

[14] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018, pp. 7132–7141.

[15] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, p. 2, 2017.

[16] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *DSAA*. IEEE, 2018, pp. 80–89.

[17] A. Abdul, J. Vermeulen, D. Wang, B. Y. Lim, and M. Kankanhalli, "Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda," in *CHI*, 2018, pp. 1–18.

[18] A. Rosenfeld and A. Richardson, "Explainability in human–agent systems," *Autonomous Agents and Multi-Agent Systems*, 2019.

[19] Y. Yao, M. Xu, Y. Wang, D. J. Crandall, and E. M. Atkins, "Unsupervised traffic accident detection in first-person videos," *arXiv*, 2019.

[20] L. Taccari, F. Sambo, L. Bravi, S. Salti, L. Sarti, M. Simoncini, and A. Lori, "Classification of crash and near-crash events from dashcam videos and telematics," in *ITSC*. IEEE, 2018, pp. 2460–2465.

[21] Y. Yuan, Y. Lu, and Q. Wang, "Adaptive forward vehicle collision warning based on driving behavior," *Neurocomputing*, 2020.

[22] X. Peng, R. Liu, Y. L. Murphey, S. Stent, and Y. Li, "Driving maneuver detection via sequence learning from vehicle signals and video images," in *ICPR*. IEEE, 2018, pp. 1265–1270.

[23] S. A. Zekany, R. G. Dreslinski, and T. F. Wenisch, "Classifying ego-vehicle road maneuvers from dashcam video," in *ITSC*. IEEE, 2019.

[24] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? A unified framework for maneuver classification and motion prediction," *Transactions on Intelligent Vehicles*, pp. 129–140, 2018.

[25] J. Kim, A. Rohrbach, T. Darrell, J. Canny, and Z. Akata, "Textual explanations for self-driving vehicles," in *ECCV*, 2018, pp. 563–578.

[26] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[29] R. Girshick, "Fast r-cnn," in *ICCV*, 2015, pp. 1440–1448.

[30] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017, pp. 1251–1258.

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv*, 2017.

[32] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," *arXiv preprint arXiv:1711.07128*, 2017.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Matteo Simoncini received the MSc degree in computer engineering from the University of Florence, Italy, in 2016. He is currently working toward an industrial Ph.D. degree at Verizon Connect Research, Florence, Italy and the University of Florence, Italy. His research interests include intelligent transportation systems, machine learning, computer vision and their applications.

Douglas Coimbra de Andrade graduated mechanical aeronautical engineer in 2005 and received his D. Sc. in the field of Aerospace Systems and Mechatronics in 2017, both from Instituto Tecnologico de Aeronautica, Brazil. His research interests include AI applied to computer vision, video analytics and HPC.

Leonardo Taccari received the Ph.D. in Operations Research from Politecnico di Milano in 2015. He is currently lead scientist at Verizon Connect. He is author or coauthor of more than 20 scientific articles and 10 patents. His research interests include machine learning and mathematical optimization.

Samuele Salti is currently associate professor at the Department of Computer Science and Engineering (DISI) of the University of Bologna, Italy. His main research interests are computer vision and machine learning. Dr. Salti has co-authored 51 publications and 8 patents. He received two best paper awards runner-up (3DIMPVT 2011, 3DV 2021) and 5 outstanding reviewer awards at major conferences (CVPR, NeurIPS, ICCV, ICML). In 2020, he cofounded the start-up eyecan.ai.

Luca Kubin received the MSc degree in communication engineering from the University of Parma in 2015. His research interests include deep learning, computer vision, and digital signal processing.

Fabio Schoen is full professor of Operations Research at the Department of Information Engineering of the University of Florence. He is the author or coauthor of more than 60 papers in scientific journals, mostly in the field of optimization. He is the scientific coordinator of the PhD program in Information Engineering. His research interests are mainly in global optimization methods and applications.

Francesco Sambo holds a PhD in bioinformatics and artificial intelligence from the university of Padova. He is currently Chief Data Scientist at Verizon Connect. His research interests include road scene understanding and predictive maintenance. He is author or coauthor of more than 50 scientific papers and 10 patents.