Detection of stop sign violations from dashcam data

Luca Bravi*†, Luca Kubin*†, Stefano Caprasecca†, Douglas Coimbra de Andrade†, Matteo Simoncini†, Leonardo Taccari†, Francesco Sambo†‡

***Abstract*—In this article we present a novel machine learning pipeline for automatic detection of stop sign violations from dashcam videos, Inertial Measurement Units (IMU) and Global Positioning System (GPS) data. We developed a two-step approach, including a detector (Stop Sign Detector) capable of identifying stop signs presence, position, and size within video frames, followed by a classifier (Stop Violation Classifier) that assesses the presence of violations along with a severity score. The Stop Sign Detector is a deep convolutional neural network (CNN) for image classification, which leverages the information contained in its deeper layer feature maps in order to extract estimates of position and size of the detected stop signs. The Stop Violation Classifier fuses the information provided by the Stop Sign Detector with IMU/GPS data to assess the presence and severity of a stop sign violation. The proposed approach has been tested on several thousands of real-world videos, recorded from US vehicles, in all kinds of weather conditions, times of the day and environments. Our method achieves an area under the precision-recall curve of 94% with a required computational time of 2.4 seconds to process a 16-second video entirely on CPU.**

***Index Terms*—Stop Sign Violations, Convolutional Neural Networks, Machine Learning, Dashcam, GPS.**

I. Introduction

STOP sign violations are among the main causes of road accidents: each year in the US, nearly 700,000 police reported crashes occur at stop sign intersections, around 1/3 of these involve injuries, and more than 3,000 are fatal [1]. Systematic detection of stop sign violations is thus of the utmost importance in fleet management software and in driver coaching platforms, where data extracted from connected vehicles are processed to identify dangerous behaviors and to coach drivers to improve their driving style. An example of a machine learning system that provides an additional layer of semantics to the analysis of crash and near-crash events is the one proposed in [2], and included in the Verizon Connect video product. The detection of driving violations would also be extremely useful to support researchers in Intelligent Transportation Systems and decision makers in municipalities or government agencies. For instance, a database of stop sign violations would help in the identification of intersections that are more subject to driver violations or would enable analysis on road or weather conditions that are more likely to cause a crash in the presence of stop sign violations. This kind of analysis could even recommend, in some cases, more appropriate traffic control systems, such as traffic lights or roundabouts.

In this article we propose a machine learning method, trained and tested on real-world data, that combines video, GPS speed and IMU data to automatically identify events containing a stop sign violation. Note that we do not leverage GPS positions in conjunction with public maps. This choice was motivated by the fact that no existing map service provides sufficiently high-quality information around stop signs. Moreover, locations provided by GPS are not accurate enough to give a reliable understanding of intersection crossings (in urban areas, the average GPS position error is around 10 meters [3]).

The proposed approach decomposes the detection of stop sign violations into two distinct phases. In the first phase we analyze the video frame by frame, in order to estimate the probability that stop signs are present in its content. In the second stage of our pipeline, we use the generated probabilities along with the GPS speed and IMU information to assess whether a stop sign violation happened, and possibly the severity of the analyzed event. Our main contributions are:

- a fast and effective stop sign detector based on a convolutional neural network (CNN) architecture, that is trained in a semi-supervised way to identify the presence and the location of stop signs in dashcam images of real roads;
- a machine learning pipeline, based on features extracted from video and sensor data, able to discriminate whether a stop sign violation is present in a video, and also classify the severity of the violation into one of three possible different classes;

We also provide an extensive set of experiments, in which we evaluate different network architectures, from shallower to deeper, working on different frame sizes, in order to assess the feasibility of running the entire pipeline within the vehicle, targeting edge and mobile devices. On average, our method achieves an area under the precision recall curve up to 95% (97% during daytime and 93% during night) with a required computational time between 2.4 s and 15.7 s to process a 16 second video on a CPU[1].

The paper is organized as follows: in Section II we review the existing literature; in Section III we describe our proposed approach, highlighting the way in which we composed our dataset and the employed training procedure; finally, experimental results are shown in Section IV followed by concluding remarks in Section V.

II. Related Work

The problem of automatically detecting stop sign violations from vehicle sensors has not yet received much attention. To the best of our knowledge, the only work that addresses it is [4], where the authors design a module capable of

*These authors contributed equally.
†Verizon Connect Research, Florence, Italy
‡Email: `francesco.sambo@verizonconnect.com`

[1]Measured on a Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz, see Section IV for more details.

alerting drivers in dangerous situations, through the emission of acoustic messages. In this work the authors analyze, in a real-time setting, video frames coming from 2 digital cameras mounted on the vehicle roof, along with signal coming from the speedometer embedded inside the vehicle. Among the considered dangerous situations there are stop sign violations. Similarly to us, they adopt a two phases approach: a first phase consisting in stop sign detection, followed by a stop sign violation assessment. The authors perform stop sign detection by means of a classical computer vision technique: after illumination enhancement of the acquired video frames, needed to make their pipeline independent of lighting conditions, they apply a color based segmentation on image regions that are likely to contain traffic signs; finally, they classify the proposed segments by means of shape matching techniques. In order to assess if a violation is present in the analyzed footage, they check whether the vehicle speed falls to zero in proximity of the detected stop signs. The authors claim to reach an overall accuracy of about 92% with the stop sign detector module, by analyzing about 5000 images recorded during a drive test, while the performance of the violation detector is not specified because it was defined as a simple rule on the vehicle speed. There are some key differences between the method we propose and the one described in [4]. First of all, we propose a machine learning method that is more robust to noise, compared to a rule based method with a simple threshold on the vehicle speed, allowing us to cope with inaccurate sensor data (e.g., the GPS speed which is a noisy estimate of the actual speed). Second, the use of a modern deep learning method for stop sign detection gives much better generalization capability than more traditional techniques [5], [6], without the need for employing illumination enhancement techniques to handle different lighting conditions. Third, we include tests on almost 40 hours of videos, recorded in a real-world environment by hundreds of different drivers in different time of the day and weather conditions, thus, we give a better experimental estimate of the accuracy that our system can reach compared to a single drive test. Finally, our proposed method allows us also to classify the severity of the committed violations and not only detect their presence.

Clearly, an important sub-task of the problem we aim to solve is the detection of the stop sign within a video or image. The related problem of traffic signs detection and classification has been widely investigated. In recent years, convolutional neural networks have proven to be extremely effective in both image classification (assessing whether a target class of objects is present in a image or not) and object detection (the problem of identifying an object along with the area that it occupies within the image), clearly outperforming more traditional techniques [7]. A pioneer work using a CNN for traffic signs classification is [8] where a convolutional neural network reaches better-than-human performance, obtaining a recognition rate of 99.46% over the German Traffic Sign Recognition Benchmark [9]. Even though this dataset presents challenging aspects (e.g., different lighting conditions and a natural environment), it only contains images that are already cropped around the traffic sign. The dataset German Traffic Sign Detection Benchmark [10] instead targets the problem of traffic sign detection (i.e., joint classification and localization) in a road scene. This dataset was used in the survey [11] to compare state of the art object detection systems, pre-trained on the Microsoft COCO dataset [12] and fine tuned on German Traffic Sign Detection Benchmark. The survey shows that Faster R-CNN [13] obtains the best performance in terms of mean average precision (mAP), while R-FCN [14] provides the best trade-off between accuracy and execution time. In [15], the authors report state of the art performance on another dataset, the Swedish Traffic Signs Dataset [16]. In their framework they use two separate convolutional neural networks, following an approach similar to the one used in R-CNN [17]: they first use a fully convolutional network to generate region proposals, containing possible traffic signs, and then a CNN classifier able to eliminate false positives and to discriminate among different detected signs. The major novelty of their work is to train a fully convolutional neural network, usually designed for semantic segmentation tasks [18] using bounding-box level annotations. Another important idea the authors use in their work is to exploit priors on the locations of traffic signs, i.e., that they usually appears in the left/right side of the image frame. In [19], the authors tackle the traffic sign detection problem focusing on real-world, noisy images. Their major contribution is the creation of a big traffic sign benchmark dataset, with 100000 images containing 30000 traffic sign instances (Tsinghua-Tencent 100K benchmark). In [20], the authors propose a different approach to deal with the problem that traffic signs are small objects compared to the image size and that they are hard to be distinguished from false targets in complex street scenes without any context information. They propose a stacked model composed of a convolutional encoder, followed by a recurrent neural network. The recurrent neural network, equipped with an attention mechanism, explicitly focuses on local regions of interest allowing better detection performance. In [21], the authors propose an incremental framework from traffic signs detection, tracking, and recognition from driving videos. In their pipeline, the authors split the tracking algorithm into 2 distinct pieces: a motion and appearance models, followed by a scale-based weighted classifier giving higher votes to large-scale detected signs to improve the overall recognition performance. On a different research line, in [22], the authors show the crucial limitations related to sensor fusion algorithms when processing heterogeneous vehicle sensor data. Similarly, in [23] the authors propose a distributed filtering algorithm to mitigate noisy and partial observations in vehicular data.

III. Methodology

As stated in the introduction, and similarly to [4], we decompose the problem of detecting stop sign violations into two distinct sub-tasks: we first assess the presence of stop signs within the content of a video; then, if any stop has been detected, we assess the presence of a violation along with its severity level. Figure 1 shows an overview of the whole pipeline of the proposed method. The footage recorded by the dashcam is processed, independently frame by frame, by the

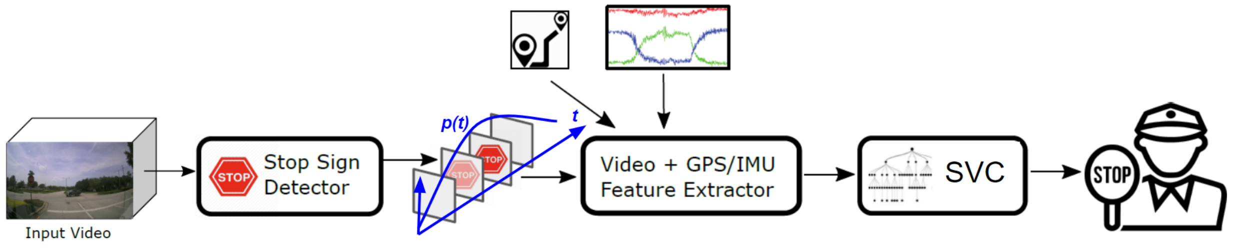

Fig. 1. **Stop Sign Violation Pipeline** - The input video is initially processed by the Stop Sign Detector that outputs, for every input frame t, the probability $p(t)$ that at least a stop sign is present in its content. Stop sign probabilities are then fused together with speed and acceleration by the Feature Extractor block. The obtained features are fed to the Stop Violation Classifier that outputs the severity of the analyzed event.

Stop Sign Detector that outputs the probability that at least a stop sign is visible in the image (i.e., it compresses the whole video into an univariate time-series containing at every timestamp the probability of the presence of at least a stop sign in the frame content). Time varying stop sign probabilities are then fused with GPS speed and vehicle acceleration and then processed into a set of features by the Feature Extractor block. The obtained features are then used by the Stop Violation Classifier (implemented with a Random Forest Regressor) that returns the severity of the analyzed event.

The main technical challenges solved by our approach are related to the several sources of uncertainty that appear in a realistic setting:

- dashcam videos often have suboptimal lighting or visibility conditions, and cameras can have different installation positions;
- the time instant in which the vehicle crosses the intersection is not precisely known: stop signs might disappear from the scene at a variable distance from the intersection crossing;
- the GPS speed is a noisy estimate of the instantaneous speed: various sources of noise affects the GPS speed estimate on both phase (delay) and amplitude components;
- acceleration measurements might be affected by a rotation error due to suboptimal camera mounting position or miscalibration of the IMU sensors.

These sources of uncertainty are the reasons for which we employ an end-to-end machine learning pipeline, instead of manually defined thresholds applied over the vehicle speed (as done in [4]). Indeed, we empirically show in Section IV that the use of single thresholds in a realistic setting is significantly less effective than the adoption of machine learning models that exploit the correlations between several features describing the driver behavior in relation to the stop sign intersection.

In the remaining of this section, we first describe the data we used for our experiments and define the problem (III-A), afterwards we describe the semi-automatic procedure we developed to extract our ground-truth data for training the Stop Sign Detector (III-B). Then, we describe the details of each component of the pipeline: the Stop Sign Detector (III-C), the Feature Extractor (III-D) and the Stop Violation Classifier (III-E).

A. *Stop Violation Dataset*

The stop violation dataset is composed of 8,931 videos recorded from dashcams installed on vehicles traveling in the US. Every video included in the dataset comes with associated GPS speed and accelerometer data. Videos are recorded in all kinds of weather conditions, times of the day and environments. All the samples contained in the dataset are manually annotated into three different semantic classes, representing three distinct levels of severity:

- No-violation: in this case, either the video does not contain a stop-controlled intersection, or, if there is any, the driver comes to a complete stop before entering the intersection (i.e., the driver halts the vehicle at the stop line).
- Mild-violation: the vehicle does not stop completely before the intersection, even though it slows down significantly before reaching the stop line. This kind of violation, also known as ”rolling stop”, is very common, and it usually happens when the driver effectively considers the stop sign as a yield sign.
- Severe-violation: this class includes videos where the driver reaches the stop line at high speed without slowing down before crossing the intersection. In these situations, the driver compromises its awareness of other vehicles approaching the same intersection in the orthogonal direction.

In our data, the class distribution is significantly imbalanced: only 42% of the videos contain a stop-controlled intersection and among them almost 80% contain a violation. Among violations, only 30% are severe, further motivating the need for assessing the severity as well as the presence of violations.

B. *Stop Sign Dataset*

Besides the 8,931 labeled videos, we have several millions of unlabeled video samples at our disposal. We exploit this source of unlabeled data in order to build a dataset of frames containing stop signs (positive samples) and frames without stop signs (negative samples) that we use to train our custom Stop Sign Detector.

In Figure 2, we show some examples of our data. Images are obtained from cameras with a large field of view, where the size of stop signs is typically very small compared to the

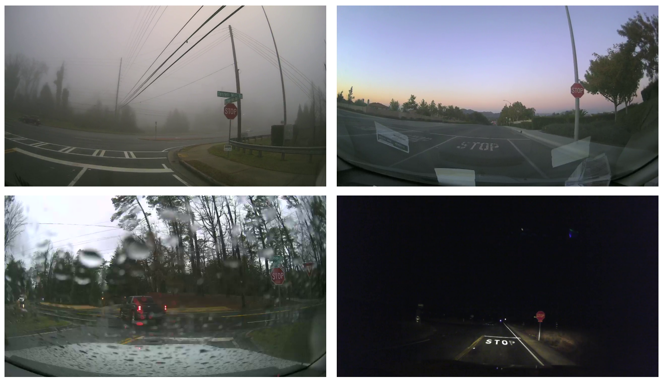
Fig. 2. **Example of frames with stop signs extracted from our dataset.** Stop signs are small, and weather/lighting conditions can be challenging.

whole scene; moreover, videos are recorded in all sorts of real-world conditions, including challenging visibility situations.

Looking for video frames containing stop signs within a set of millions of unlabeled images is a hard task. To facilitate this we can exploit an out-of-the-box object detector to find candidate frames containing stop signs. We decide to use YOLOv3 [24] provided in the deep learning framework GluonCV [25], [26] pre-trained on Microsoft COCO dataset. This choice is motivated by the fact that COCO dataset is a large, highly curated, extremely popular dataset that provides a stop sign category among its set of annotated objects. However, the stop sign category of the COCO dataset contains several traffic signs incorrectly labeled as stop sign, as well as stop signs captured from behind, not relevant and misleading for our task (see Figure 3). This leads to systematic false positive detections produced by YOLOv3 trained on COCO, clearly detrimental for the performance of our system. Therefore, we developed a semi-automatic procedure to clean our dataset from these false positives.

In order to populate our dataset we first choose 130,000 images from a set of $\sim$ 10k videos by running YOLOv3 on each frame. In each video, a subset of frames containing stop signs detected with high confidence by the object detector are selected as positive samples. For each positive example, we also randomly sample another frame from the same video, not containing stop signs, in order to populate the negative class. Including negative samples from the same videos used for extracting positive ones is crucial to avoid bias due to the lighting conditions or the camera position and inclination.

After the initial population is selected, we apply the following semi-automatic pruning procedure to get rid of the false positive stop sign detections. We only focus on the population of positive selected samples P and we use the following semi-supervised iterative label correction procedure:

1) manually annotate a small set $P_{\text{err}} \subset P$ of examples of label error and a small set $P_{\text{ok}} \subset P$ of correct labels
2) train a model [2] Φ on $\{P_{\text{err}}, P_{\text{ok}}\}$ to identify wrong labels
3) run Φ on $P \setminus \{P_{\text{err}}, P_{\text{ok}}\}$ to identify the set of examples that, according to the model Φ, have a wrong label with very high confidence (≥ 0.95)
4) add such examples (if present) to P_{err} and go to 2)
5) otherwise, exit.

[2]In our experiments we use ResNet18 [6] mainly for its fast training procedure.

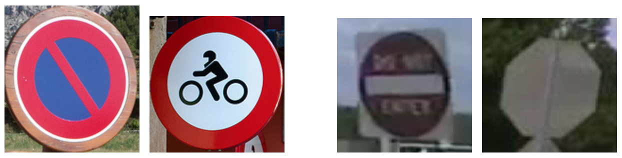
Fig. 3. Examples of incorrect stop sign annotations in COCO (leftmost images) and incorrect YOLOv3 detections on our data (rightmost images).

At each iteration, we use the highest confidence outputs of the model Φ trained at the previous iteration to expand the set of examples with incorrect labels. On our dataset, this procedure needs a small initial set of manually identified errors to fix around 10,000 incorrect labels.

Finally, once this cleaning procedure is concluded, we split our dataset into 2 parts: a training set ($\sim$ 115000 images) and a validation set ($\sim$ 15000 images). We use the validation set to evaluate models during training, avoiding overfitting.

Note that one could think of simply using model Φ, in conjuction with YOLOv3, to filter out the false positive detections. However, using two models would be too computationally demanding, and this is why we choose to use a single lightweight binary model, as described in the next subsection.

C. Stop Sign Detector

The main purpose of the Stop Sign Detector is to recognize the presence of a stop-controlled intersection. Since our dataset does not contain bounding box level annotations, but only binary labels referring to the presence or the absence of stop signs within video frames, we use a binary image classifier estimating the probability that at least a stop sign is present in the frame. However, an object detection model could provide useful additional information compared to an image classifier – namely, the position and size of the detected objects. If provided, this kind of information could be used to filter out stop sign detections not pertinent to the driver vehicle lane (i.e., the lane on which the vehicle mounting the dashcam is traveling), that would cause false detections of violations. However, with a simple adaptation of a modern CNN architecture, we show that it is still possible to recover some information about object sizes and positions.

Figure 4 shows the structure of the Stop Sign Detector. The upper branch is a ResNet [6] architecture, decomposed in its main building blocks (i.e., a stack of convolutional layers followed by a global average pooling and a fully connected layer). Note that the ResNet architecture can be easily replaced by any CNN presenting this structure. The input RGB image is first processed by a sequence of convolutional layers, obtaining a feature map $F \in \mathbb{R}^{h \times w \times c}$, where c is the number of channels, and (h, w) are its spatial dimensions (height and width). This tensor $F = \{F_{ijk}\}$ (where i and j span respectively the height and width dimensions of the tensor, and k spans the channels one) is spatially aggregated with a global average pooling, obtaining a vector s of size c, where:

$$s_k = \frac{1}{hw}\sum_{i=1}^{h}\sum_{j=1}^{w} F_{ijk} \tag{1}$$

Posterior probabilities y are obtained by mapping the vector s through a fully connected layer with weights $\omega \in \mathbb{R}^c$ and by applying the sigmoid σ activation function on the result:

$$y = \sigma\left(\sum_{k=1}^{c} s_k \omega_k\right) \tag{2}$$

The lower branch of the architecture implements an unsupervised method for retrieving information about detected stop sign positions and sizes. Let's consider each position i, j of the tensor F being the vector $F_{ij} \in \mathbb{R}^c$. By taking the inner product between F_{ij} and the weights $\omega \in \mathbb{R}^c$ of the fully connected layer and by passing the obtained result through an element-wise rectified linear unit (ReLu) [27], we compute:

$$f_{ij} = \text{ReLu}(F_{ij}\cdot\omega) = \text{ReLu}\left(\sum_{k=1}^{c} F_{ijk} w_k\right) \tag{3}$$

The obtained matrix $\mathbf{f} \in \mathbb{R}^{w\times h}$, known as saliency map [28], is a convenient representation of the spatial distribution of the network activations. As one can see from Figure 5, when the input image contains stop signs, the network activates its neurons in their neighborhood (i.e., the power of $\mathbf{f}$ is localized around the stop sign positions). Instead, when stop signs are not present in the input image, the activation is negligible almost everywhere. Thus, in case that at least a stop sign is present in the frame, to obtain information about the size and position of the detected stop signs it is sufficient to extract circular Gaussian blobs (e.g., as showed in [29], [30]) from a gray-scale representation of the saliency map: every blob has a mean and a radial standard deviation, that are good proxies for position and size of the stop signs in the image. This approach provides an extremely efficient model that performs both detection and classification in one shot, without relying on computationally demanding object detection models. Moreover, the localization information emerges from the model in a completely unsupervised way, since we trained the network only with annotations related to the presence of a stop sign, not its position.

All the networks we consider in our work are initially trained on the Imagenet dataset and then fine-tuned on our domain specific dataset. We use stochastic gradient descent with momentum as optimizer (learning rate = 0.001, momentum = 0.9) and we scale down the learning rate by a factor 0.1 every 10 epochs. We train the networks for a total amount of 50 epochs, keeping track of the models that achieves best validation accuracy. In order to improve the performance of the Stop Sign Detector, we employ a pipeline of data augmentation during training randomly jittering the image brightness by a factor 0.5, and randomly rotating the image in the range [-10, 10] degrees [31]. We measured in our experiments that these kinds of random transformations improved the validation accuracy of the Stop Sign Detector by almost 1% of accuracy. With this architecture and our training procedure, we reach an accuracy on the validation of more than 99% at the stopping epoch, which is considerably better than 95% obtained with the implementation of YOLOv3 trained on COCO provided by GluonCV[3]. Moreover, our model is able to generalize without fine-tuning on the German Traffic Sign Detection Benchmark, where it can correctly detect 31 out of 32 stop signs.

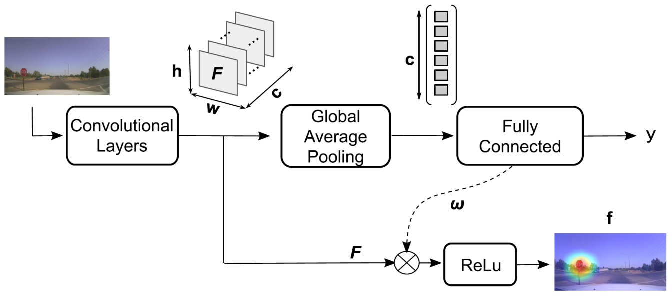

Fig. 4. **Stop Sign Detector.** A sequence of convolutional layers extracts features from the input frame. The extracted features are used by the upper branch of the block scheme to perform classification and by the lower one to compute the saliency map.

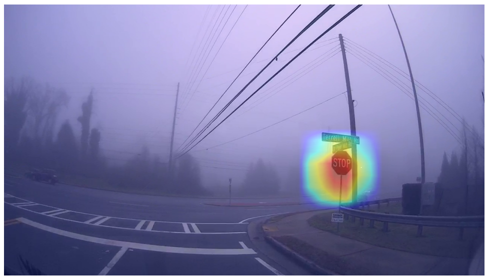

Fig. 5. **Saliency map computed from the stop sign detector.** When the image contains stop signs, the network focuses its spatial activation in the neighborhood around stop sign positions.

D. Feature Extractor

The aim of the two phases pipeline is to first assess whether the vehicle crosses a stop-controlled intersection and, in case a crossing occurs, to identify whether there is a violation along with its severity. In order to assess the severity, in cases in which a stop crossing is present within the video content, we merge the information provided by the Stop Sign Detector together with the telematics data (i.e., speed and acceleration) by extracting features capable of capturing the dynamics of the stop crossing. Along with features related to stop crossing dynamics, we extract features related to the trajectories of the detected stop signs, in order to decrease the number of false positive violations detected by our system. In other words, our feature extraction procedure transforms time varying-signals (i.e., speed, acceleration, stop sign probabilities, and saliency maps) into multi-dimensional vectors containing important characteristics used by the Stop Violation Classifier to assess the severity of the analyzed events.

[3]In order to reduce an object detection problem to a binary classification one, we consider for every frame the maximum stop sign probability detected by YOLOv3 as a proxy for the probability that at least a stop sign is present in the frame.

Figure 6 shows the idea behind the method we use to extract the features related to the stop crossing dynamics. The plot on the upper part of the figure shows the time evolution of the Stop Sign Detector output (i.e., the probability that at least a stop sign is present in the content of a video frame as a function of time, $p(t)$) when a vehicle is approaching an intersection controlled by a stop sign. The plots on the lower part of the figure show the speed and acceleration profiles respectively. The probability profile starts increasing as soon as

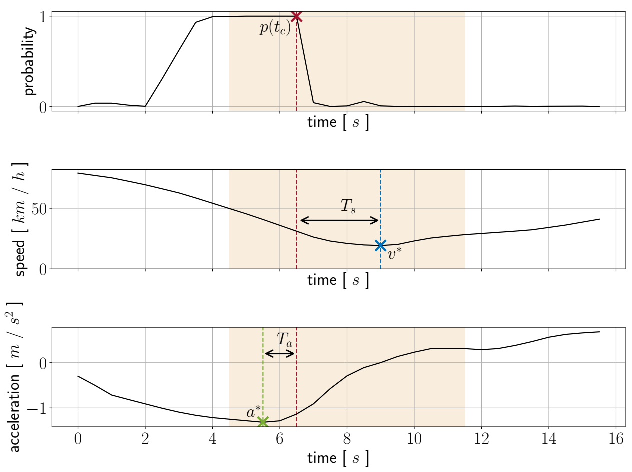

Fig. 6. **Time dynamics of a stop sign violation**: the upper plot shows the output of the stop sign detector for each frame, while vehicle speed and acceleration are shown in the bottom plots. In all the plots we highlight the time window used to extract features related to stop crossing dynamics.

the stop sign becomes visible in the scene. The red cross in the upper plot denotes the instant in which the stop sign disappears from the scene, which we use as estimate for the beginning of the intersection crossing (crossing point). In practice, we choose as crossing point (t_c in the figure) the last time point in which the probability profile assumes a value close to one (i.e., $t_c = \arg\max\{p(t) \approx 1\}$). The features are then extracted in a time window around the crossing point t_c. In particular, we focus on a time slice of 7 seconds, from 2 seconds ahead the estimated crossing time to 5 seconds after it (the window is highlighted in yellow in Figure 6). We found this optimal time window, accounting for random delays among different sensors, through a cross-validated grid search performed on our dataset.

Given the signal slices in the aforementioned time window, we extract the following quantities:

- Minimum speed value (v^* in Figure 6).
- Time shift between the crossing instant and the minimum speed instant (T_s in Figure 6).
- Minimum acceleration value (a^* in Figure 6)
- Time shift between the crossing instant and the minimum acceleration instant (T_a in Figure 6).

The other family of features tries to characterize the trajectories of detected stop signs through video frames. The aim of these features is related to the fact that in some cases stop signs that are not pertinent to the driver lane could appear in the video, typically on the left-hand side of the video frame. In order to characterize stop signs trajectories through video frames, we exploit the saliency map, as described in section III-C, to obtain estimates of the positions and the sizes of the detected stop signs. Saliency maps, computed for every video frame by the Stop Sign Detector, are stacked into a 3-dimensional tensor $S[t, i, j]$ that for each timestamp t contains a gray-scale image, indexed with i and j which span height and width respectively. In practice, we only focus on the first instant in which stop signs appear in the scene (i.e., the first time t_0 such that $p(t_0) \approx 1$) and the last instant before they disappear (i.e., the estimated crossing point t_c). For those time instants, we add as features the estimated positions and sizes of the detected stop signs (namely, $\boldsymbol{\mu}(t_0)$, $\boldsymbol{\sigma}(t_0)$, $\boldsymbol{\mu}(t_c)$, $\boldsymbol{\sigma}(t_c)$).

TABLE I
DEFINITIONS

Description	Name	Formula
First Stop Appearance	t_0	$\arg\min\{p(t) \approx 1\}$
Stop Crossing	t_c	$\arg\max\{p(t) \approx 1\}$
Minimum Speed Time	t_v	$\arg\min\{v(t) : t \in [t_c - 2, t_c + 5]\}$
Minimum Acc. Time	t_a	$\arg\min\{a(t) : t \in [t_c - 2, t_c + 5]\}$
Saliency Map	$S[t, i, j]$	-

TABLE II
STOP VIOLATION FEATURES

Description	Name	Formula
Minimum Speed Value	v^*	$\min\{v(t) : t \in [t_c - 2, t_c + 5]\}$
Speed Time Shift	T_s	$t_v - t_c$
Minimum Acc. Value	a^*	$\min\{a(t) : t \in [t_c - 2, t_c + 5]\}$
Acc. Time Shift	T_a	$t_c - t_a$
Appearance Time Stops	$\boldsymbol{\mu}(t_0)$, $\boldsymbol{\sigma}(t_0)$	$\mathrm{blob}(S[t_0, i, j])$
Crossing Time Stops	$\boldsymbol{\mu}(t_c)$, $\boldsymbol{\sigma}(t_c)$	$\mathrm{blob}(S[t_c, i, j])$

All the features presented in this section are summarized in Table II. We refer to the time-dependent speed profile with $v(t)$ and the acceleration one with $a(t)$; we express time windows and other time quantities in seconds. The blob operator used in the table denotes the blobs detector function, that given a black and white image returns mean and standard deviation of detected blobs. Related definitions needed to compute these quantities are presented in Table I.

E. Stop Violation Classifier

The machine learning model we use as Stop Violation Classifier, that maps the extracted features, presented in the previous subsection, to the severity level is a Random Forest Regressor (RFR) [32]. It is worth noting that even if the rule to implement the detection of a stop sign violation is, in principle, well defined (the driver must completely stop at the intersection), the adoption of a machine learning model is mainly needed for the following reasons:

- One of the goal is to assess the severity of the detected stop sign violations: as presented in III-A our dataset is composed of two classes of violations (i.e., mild and severe) and we want to let a model learn the best set of thresholds to discriminate between them.

- As we mentioned in Section III, we work with multiple noisy variables (i.e., GPS speed, vehicle acceleration and estimated stop sign probabilities and locations): defining multiple handcrafted rules on correlated noisy values is hard because one needs to consider the joint distribution instead of setting independent thresholds on each value.

The choice of a regressor, instead of a classifier, is due to the fact that stop violation labels present an intrinsic order embedded in the severity value (i.e., no-violation $\rightarrow$ 0, mild-violation $\rightarrow$ 1, severe-violation $\rightarrow$ 2). For this reason, and because the RFR loss is governed by the mean square error function, our method penalizes more misclassifications $0 \leftrightarrow 2$, w.r.t. misclassifications $1 \leftrightarrow 2$ and $0 \leftrightarrow 1$. Because the output of the regressor is continuous in the interval $[0, 2]$, we finally convert the RFR continuous predictions to one of the three original classes by means of rounding.

For the seek of completeness, we compare in Section IV-C the performance of the RFR against other regression models to show that our choice is empirically optimal.

IV. Experimental results

The dataset we used to perform our experiments is the one described in III-A and contains a total of 8,931 examples: 5,900 no-violations, 2,122 mild-violations and 909 severe-violations. Videos have been down-sampled, from the original size, to a resolution of 360×640 pixels and to a sample rate of 2 fps. Accelerometer and GPS data are sampled at 2 Hz. Every sample contained in the dataset has a duration of 16 seconds (i.e., each sensor stream is composed of 32 samples).

A. Stop Violation Pipeline Computational Requirements

In order to compare the trade-off between end-to-end system accuracy, memory occupation and required computational time, we considered different architectural choices for the implementation of the Stop Sign Detector, being the bottleneck of our system. The time needed for running the Feature Extractor + Stop Violation Classifier blocks are together an order of magnitude smaller than the one needed for running the Stop Sign Detector. In particular, for running both blocks less than 0.2 seconds are required, with a memory occupation of only 40 MB. All the benchmarks presented in this section were run on a computer equipped with a CPU Intel(R) i7-6820HQ at 2.70 GHz, 32 GB of RAM and a Tesla T4 GPU.

For the implementation of the Stop Sign Detector we considered neural networks from the ResNet family with different depths (namely, ResNet18 and ResNet50) and YOLOv3 (trained on COCO dataset and with Darknet53 as backbone), running on different frame sizes (i.e., 360×640 and 180×320).

Table III reports the inference time (on CPU and on GPU) and the total memory occupation (network weights + allocated input tensor) needed for processing different batch sizes, with the neural architectures considered in this work. Although YOLOv3 is one of the most efficient off-the-shelf object detector, the adoption of a lightweight image classifier can easily halve the amount of required memory and can divide by three the computational time needed for processing one batch of data on devices where a GPU is not available.

B. Stop Violation Pipeline Performance Quality

To evaluate the performance of our approach, we reduced our multiclass problem to two distinct binary problems using two binarization strategies. We first gathered together mild-violations and severe violations in the positive class, using as negative class only the no-violation events, to evaluate the ability of the model to identify all the violations; then, we grouped together no-violations and mild-violations in the negative class, keeping only severe violations in the positive one, to evaluate the model performance specifically on the detection of the most important (and rare) events. As a metric we decided to use the area under the precision-recall (PR) curve (AUCPR, sometimes also known as average-precision (AP) score in the literature), which is particularly suitable for imbalanced problems. This score is obtained by integrating the PR curve that provides precision and recall points for different values of binary decision threshold: it assumes a value of 1 for a perfect classifier, and it decreases for each false positive misclassification performed by the model. Since the number of severe violations is scarce with respect to the cardinality of the other classes, in all the following experiments we used a 10-fold cross validation procedure to perform a comprehensive evaluation of the proposed pipeline.

In our first set of experiments, we first explored how the performance is affected by the depth of the neural network architecture and the video frame resolution, in order to estimate the trade-off between performance and computational requirements. In particular, we compared all the combinations of two different variants of the ResNet family (ResNet18 and ResNet50) with two different frame resolutions (360×640 and 180×320). Moreover, to assess the performance difference gained by using our implementation of the Stop Sign Detector, we compared it with YOLOv3 running on frame resolutions of 360×640 and 180×320. AUCPR values are

TABLE III
CNN COMPUTATIONAL REQUIREMENTS

Network	Batch Shape	Time CPU	Time GPU	RAM
ResNet18	32×180×320	2.1 s	0.05 s	180 MB
ResNet18	32×360×640	8.3 s	0.12 s	320 MB
ResNet50	32×180×320	4.6 s	0.41 s	320 MB
ResNet50	32×360×640	15.5 s	0.75 s	450 MB
YOLOv3	32×180×320	10.5 s	0.45 s	800 MB
YOLOv3	32×360×640	43.5 s	1.68 s	1300 MB

TABLE IV
PIPELINE CLASSIFICATION PERFORMANCE WITH RFR AND DIFFERENT STOP SIGN DETECTOR CONFIGURATIONS

Network	Resolution	Violation (AUCPR)	Severe (AUCPR)
ResNet18	180×320	94%	74%
ResNet18	360×640	94%	76%
ResNet50	180×320	**95%**	79%
ResNet50	360×640	**95%**	**80%**
YOLOv3	180×320	90%	67%
YOLOv3	360×640	93%	73%

reported in Table IV for all the tested combinations, for both the identification of violations (mild + severe, see Violation column) and for the identification of severe violations only (see Severe column). For both tasks, the custom trained ResNet models provide a significant improvement over the off-the-shelf YOLOv3 model, especially at lower resolution (compare the 67% obtained with YOLO with the 79% that can be obtained using our Stop Sign Detector with a ResNet50 backbone). We can see that there is not a huge difference in performance changing the depth of the ResNet architecture for the identification of generic violations. A larger variation is noticeable in the severe violation column, where frame resolution affects the performance more than the network depth. This can be explained by the fact that there is likely no need for an extremely deep network architecture for the identification of a single, well-defined object (like a stop sign), while a lower resolution can significantly affect the performance causing false stop sign detections from similar traffic signs when they are not close enough to the subject vehicle.

In the second part of our experiments, we fixed our network to ResNet18 with an input frame resolution of 180×320, mainly because it reached the best trade-off between classification accuracy, computational time and memory occupation. We then show a more detailed ablation study comparing our pipeline with different alternatives to assess the effectiveness of each component of our approach (in particular our main contributions, the Stop Sign Detector and the Feature Extractor + Stop Violation Classifier):

- a pipeline where we kept the Feature Extractor + Stop Violation Classifier but we replaced our Stop Sign Detector with an off-the-shelf object detector YOLOv3 trained on COCO dataset;
- a pipeline where we kept our Stop Sign Detector but we replaced the Feature Extractor + Stop Violation Classifier with a rule-based model (similar to the one used in [4]), that uses a threshold on the vehicle speed to identify violations;
- a pipeline where we replaced both our Stop Sign Detector with YOLOv3 and the Feature Extractor + Stop Violation Classifier with a rule-based model, similar to [4].

Figure 7 shows the precision-recall curves for the first binarization strategy (i.e., severe + mild-violations vs no-violations). The orange curve is obtained by using the Stop Sign Detector with a ResNet18 backbone, while the blue curve uses YOLOv3. As one can see from the plot legend, the former approach gains 4% of AUCPR compared with the latter one, reaching 94%. Since both curves are obtained with the same Feature Extractor + Stop Violation Classifier blocks, this performance boost, attributed to the reduction of stop sign false positives, is due to our custom model trained on an ad-hoc stop sign dataset. The black point and the green cross, in the upper right of the plot, show the precision-recall points obtained with the rule-based baseline used in [4], respectively using ResNet18 and YOLOv3 as Stop Sign Detector. Note that we could not obtain a PR curve since the rule-based model does not provide a confidence, but only a classification. From this plot, we can conclude that both the components used in our pipeline, i.e., the custom Stop Sign Detector and the Feature Extractor + Stop Violation Classifier, are crucial in significantly improving the performance of the system on the detection of violations.

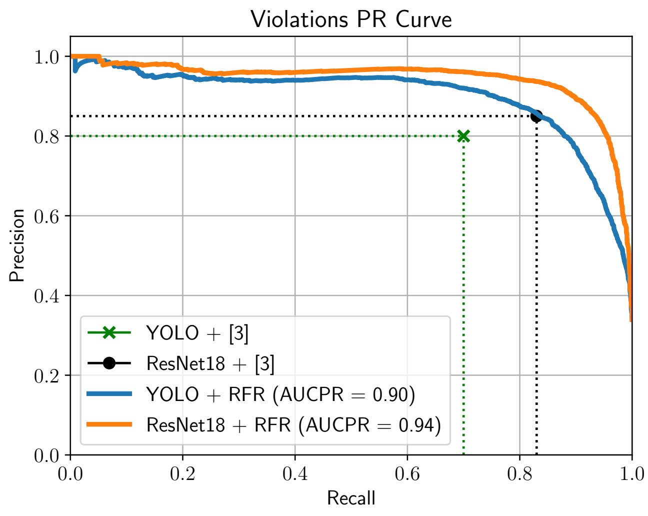

Fig. 7. **PR curve for Violation (Severe + Mild) vs No Violation**: the blue curve refers to the experiment using YOLOv3 as Stop Sign Detector, the orange to the one using ResNet18. The green cross and the black point indicate the performance of the rule-based pipelines. All methods applied at a resolution of 180×320.

Figure 8 shows the precision-recall curve for the second binarization strategy (i.e., severe-violations vs mild-violations + no-violations). In this case, the use of ResNet18 as stop sign detector in place of YOLOv3 gives a performance boost of 7% in terms of AUCPR. The reason of this fact is that YOLOv3 false positive detections, when the vehicle is likely traveling at a high speed, are vastly deteriorating the performance of the whole pipeline in recognizing severe violations.

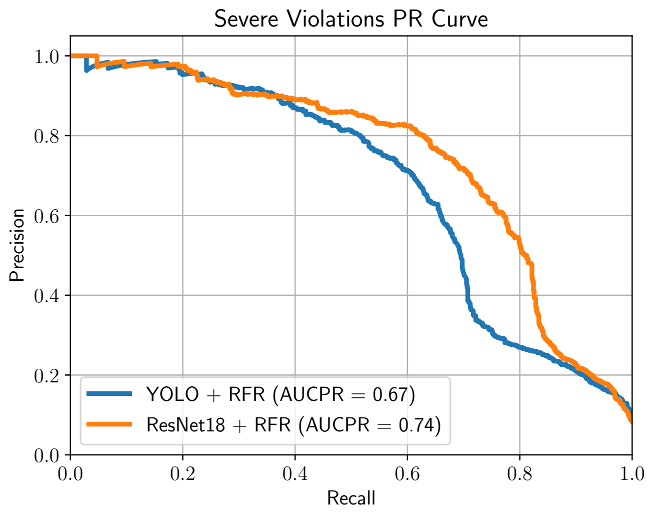

Fig. 8. **PR curve for Severe Violations vs Rest (Mild + No Violation)**: the blue curve refers to the experiment using YOLOv3 as Stop Sign Detector, the orange to the one using ResNet18. Both methods applied at a resolution of 180×320.

TABLE V
CONFUSION MATRIX OBTAINED WITH THE RESNET-BASED STOP SIGN DETECTOR @180 × 320 AND RANDOM FOREST REGRESSOR

		Predicted		
		No Violation	Mild	Severe
True	No Violation	**5560**	329	11
	Mild	200	**1879**	43
	Severe	84	333	**492**

TABLE VI
PIPELINE CLASSIFICATION PERFORMANCE WITH DIFFERENT STOP VIOLATION CLASSIFIER IMPLEMENTATIONS

Network	**Resolution**	**Violation (AUCPR)**	**Severe (AUCPR)**
Random Forest Regressor			
ResNet18	180×320	94%	74%
ResNet18	360×640	94%	76%
ResNet50	180×320	**95%**	79%
ResNet50	360×640	**95%**	**80%**
Linear Regression			
ResNet18	180×320	84%	40%
ResNet18	360×640	84%	41%
ResNet50	180×320	85%	46%
ResNet50	360×640	85%	49%
K-Nearest Neighbors			
ResNet18	180×320	88%	66%
ResNet18	360×640	90%	68%
ResNet50	180×320	89%	66%
ResNet50	360×640	90%	65%
Multi-layer Perceptron			
ResNet18	180×320	92%	75%
ResNet18	360×640	94%	76%
ResNet50	180×320	93%	77%
ResNet50	360×640	94%	**80%**

Finally, to further give an insight about the performance brought by our approach on the 3-class problem, Table V shows the full confusion matrix obtained by our proposed pipeline using ResNet18 as Stop Sign Detector.

From the point of view of computational cost, the running time of the proposed approach (using ResNet18 as Stop Sign Detector) takes around 2.4 seconds to process a 16 seconds video on an Intel(R) i7-6820HQ and 8.1 seconds on a Rasberry Pi 4, while the same pipeline employing YOLOv3 for stop sign detection requires more than 10 seconds per video on the Intel CPU. On the Raspberry CPU, running YOLOv3 with its default Darknet53 backbone was not possible. These results clearly show the advantage of using a dedicated CNN architecture compared to general purpose object detectors leading to an accurate and lightweight solution that could run in real time on edge devices.

C. Stop Violation Classifier Comparison

In this subsection, we compare different results we obtain by switching the implementation of the Stop Violation Classifier from the Random Forest Regressor to other regressors. The scope of this paragraph is to show that our choice, motivated by the fact that a Random Forest implements optimal adaptive threshold-based cuts on input features data, is also empirically optimal under a performance point of view. In particular, we focus on the following regression models with optimal hyper-parameters tuned through a cross-validated grid search:

- Linear Regression with ℓ_2 regularization (regularization strength $\alpha = 0.5$)
- K-Nearest Neighbors Regressor ($K = 8$)
- Multi-layer Perceptron Regressor (with 2 hidden layers of size 30 and 20 neurons respectively, with Rectified Linear Unit as activation functions).

In Table VI we report the results in terms of AUCPR for different combinations of Stop Sign Classifier, input frame resolution and Stop Violation Classifier model. For all the models, except for the Random Forest Regressor, we pre-process the features by subtracting their mean and dividing by their standard deviation (whitening normalization). The experimental results show that the RFR is able to exploit the extracted features at their full potential. The multi-layer perceptron is also only slightly inferior.

V. CONCLUSIONS

This paper introduced a method to automatically detect stop sign violations, leveraging joint information coming from video, GPS and IMU sensors. We proposed a two-stage machine learning model that firstly detects the presence of an intersection controlled by a stop sign, within a video stream, through the use of an ad-hoc CNN, and then infers if a violation occurs, also assessing its severity. The experimental results showed the effectiveness of our approach, in terms of both computational time and model accuracy. Moreover, we showed that the ad-hoc CNN for stop detection can run with low computational demanding settings in such a way that the entire pipeline can run in real time even on edge devices.

Our future direction of research is to generalize the proposed system in order to detect other violations in real-time, as well as other potentially interesting or dangerous situations occurring in road scenes. Another possible avenue of research would be towards the anticipation of violations, to be integrated in Advanced Driving Assistance Systems. Moreover, if a larger dataset were available, possible improvements could involve employing end-to-end time-aware neural networks, working directly on raw video and GPS/IMU data streams.

REFERENCES

[1] R. A. Retting, H. B. Weinstein, and M. G. Solomon, "Analysis of motor-vehicle crashes at stop signs in four US cities," *Journal of Safety Research*, vol. 34, no. 5, pp. 485–489, 2003.

[2] 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 *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2018, pp. 2460–2465.

[3] K. Merry and P. Bettinger, "Smartphone gps accuracy study in an urban environment," *PloS one*, vol. 14, no. 7, p. e0219890, 2019.

[4] N. Aliane, J. Fernandez, M. Mata, and S. Bemposta, "A system for traffic violation detection," *Sensors*, vol. 14, no. 11, 2014.

[5] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. Kruthiventi, and R. V. Babu, "A taxonomy of deep convolutional neural nets for computer vision," *Frontiers in Robotics and AI*, vol. 2, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural networks*, vol. 32, pp. 333–338, 2012.

[9] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

[10] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.

[11] A. Arcos-Garcia, J. A. Alvarez-Garcia, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, 2018.

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Cbjects in Context," in *European conference on computer vision*. Springer, 2014.

[13] 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.

[14] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[15] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," *Neurocomputing*, vol. 214, pp. 758–766, 2016.

[16] F. Larsson and M. Felsberg, "Using Fourier descriptors and spatial models for traffic sign recognition," in *Scandinavian conference on image analysis*. Springer, 2011, pp. 238–249.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[19] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[20] Y. Yuan, Z. Xiong, and Q. Wang, "Vssa-net: vertical spatial sequence attention network for traffic sign detection," *IEEE transactions on image processing*, vol. 28, no. 7, pp. 3423–3434, 2019.

[21] ——, "An incremental framework for video-based traffic sign detection, tracking, and recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1918–1929, 2016.

[22] P. Kaur and R. Sobti, "Sensor fusion algorithm for software based advanced driver-assistance intelligent systems," in *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. IEEE, 2018, pp. 457–460.

[23] J. Gao and H. Tembine, "Distributed mean-field-type filters for traffic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 507–521, 2018.

[24] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[25] J. Guo, H. He, T. He, L. Lausen *et al.*, "GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing," *Journal of Machine Learning Research*, vol. 21, no. 23, pp. 1–7, 2020.

[26] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of freebies for training object detection neural networks," *arXiv preprint arXiv:1902.04103*, 2019.

[27] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8609–8613.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[29] H. Kong, H. C. Akakin, and S. E. Sarma, "A generalized laplacian of gaussian filter for blob detection and its applications," *IEEE transactions on cybernetics*, vol. 43, no. 6, pp. 1719–1733, 2013.

[30] S. Hinz, "Fast and subpixel precise blob detection and attribution," in *IEEE International Conference on Image Processing 2005*, vol. 3. IEEE, 2005, pp. III–457.

[31] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[32] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, 2001.

Luca Bravi received the M.Sc. degree in computer engineering and the Ph.D. degree in optimization from the University of Florence, in 2012 and 2016, respectively. Since 2015 and currently he is a data scientist in Verizon Connect. His current research interests include areas of machine learning applied to telematics and nonlinear optimization.

Luca Kubin received the MSc degree in communication engineering from the University of Parma in 2015. He is currently a Ph.D. candidate at Verizon Connect Research in collaboration with the University of Florence. His research interests include deep learning, computer vision, and digital signal processing.

Stefano Caprasecca is a Data Scientist at Verizon Connect since 2018. He holds a MSc in Theoretical and Computational Chemistry and a PhD in Physics, and is the author of more than 30 articles in peer-reviewed journals. His current research is focussed on the predictive maintenance of vehicle telematics devices.

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.

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.

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 applied to transportation, logistics, and energy.

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 40 scientific papers and 10 patents.