ISSN 2038-0909

Commun. Appl. Ind. Math. 13 (1), 2022, 78–95

DOI: 10.2478/caim-2022-0007

Research Article

# Deep learning-assisted analysis of automobiles handling performances

**Davide Sapienza[14] , Davide Paganelli[2] , Marco Prato[1] , Marko Bertogna[1] , Matteo Spallanzani[3*]**

[1]Dipartimento di Scienze Fisiche, Informatiche e Matematiche, Università di Modena e Reggio Emilia, Via Giuseppe Campi 213/B, 41125 Modena, Italy

[2]Vehicle Dynamics, Maserati, Via Sallustio 6, 41123 Modena, Italy

[3]Departement Informationstechnologie und Elektrotechnik, ETH Zürich, Gloriastrasse 35, 8092 Zürich, Switzerland

[4]Department of Mathematical, Physical and Computer Sciences, University of Parma, Parco Area delle Scienze 7/A, Parma, 43124, Italy

[*]Email address for correspondence: spmatteo@ethz.ch

Communicated by Giorgio Fotia

Received on 04 26, 2022. Accepted on 11 09, 2022.

## Abstract

The luxury car market has demanding product development standards aimed at providing state-of-the-art features in the automotive domain. Handling performance is amongst the most important properties that must be assessed when developing a new car model. In this work, we analyse the problem of predicting subjective evaluations of automobiles handling performances from objective records of driving sessions. A record is a multi-dimensional time series describing the temporal evolution of the mechanical state of an automobile. A categorical variable quantifies the evaluations of handling properties. We describe an original deep learning system, featuring a denoising autoencoder and hierarchical attention mechanisms, that we designed to solve this task. Attention mechanisms intrinsically compute probability distributions over their inputs' components. Combining this feature with the saliency maps technique, our system can compute *heatmaps* that provide a visual aid to identify the physical events conditioning its predictions.

*Keywords:* `Automotive engineering, Automobiles design, Deep learning, Saliency maps, Attention mechanisms`

*AMS subject classification:* `68T07,90C90`

## 1. Introduction and problem statement

In human-vehicle systems, the human driver represents a complex controller in a feedback loop. Handling is defined as the way a vehicle responds and reacts to its driver's inputs. This definition encompasses an extremely large range of situations since it depends on several variables, including the vehicle's mechanical properties, the context of the driving experience (road, weather) and the driver. For these reasons, handling properties of different vehicles are usually assessed and compared by interviewing professional drivers about their experiences when performing specific manoeuvres with the different automobiles on a given circuit.

The typical development process for such complex products is framed as a cycle composed of three phases, which are iterated until a satisfactory solution is attained. Before the first iteration, the engineers and the drivers define a list of handling properties that should be evaluated (e.g., stability during overtaking, understeering, oversteering) and of related test manoeuvres (e.g., overtakings, tight curves). These choices are kept fixed through all the iterations. The drivers assess handling properties by grading the response of the vehicle during the agreed-upon manoeuvres. Then, the development cycle proceeds as follows:

1. [**design**] the engineers instantiate a tuple of *synthesis parameters* (e.g., the wheelbase, the ground clearance) that describe the vehicle to be tested; from these parameters, a physical prototype of the vehicle can be built, or a virtual model of the car can be created;

2. **[test]** the drivers perform the agreed manoeuvres using the physical (or virtual) automobile and grade its performance; during the tests, numerical records tracking the automobile's mechanical state are registered;
3. **[analysis]** the engineers analyse the driving sessions' records to identify the events that have informed the drivers' evaluations; once these events have been identified and their causes have been traced back to the synthesis parameters, the engineers can apply changes to the synthesis parameters to correct the undesired behaviours; the cycle starts over.

Each iteration of this development cycle is expensive and time-consuming. Hence, it is desirable not only to reduce the number of iterations, but also to reduce the time spent on each iteration. In particular, the qualitative nature of the questions submitted to the drivers and the high degree of subjectivity of the answers make it extremely difficult and time-consuming to identify the events that informed the drivers' evaluations. Moreover, attributing an evaluation to an incorrect event can mislead the identification of the responsible synthesis parameters. The analysis step is thus a critical bottleneck for the whole development cycle. In the following, we propose a formalisation of this problem.

We identify a vehicle with its mechanical setting, described by a tuple

$$\mathbf{r} = (r_1, \dots, r_{N_r})' \tag{1}$$

of $N_r$ **synthesis parameters**. We use the apex ($'$) to denote transposition. This vector is chosen from some design space $R$. In our experiments we could modify $N_r = 46$ parameters. We used a simulation environment to turn this vector into a virtual model of the vehicle, which we could load onto a driving simulator.

We then asked a small number of professional pilots to perform specific manoeuvres with this vehicle while completing some laps on a virtual circuit. The circuit was kept fixed to reduce the number of factors that could impact the evaluations. During each test run, data from sensors (mainly accelerometers and dynamometers) and actuators (steering wheel, brake and throttle pedals) was collected at a fixed frequency, producing a **record**

$$\{\mathbf{z}(t) = (z_1(t), \dots, z_{N_z}(t))'\}_{t=1,\dots,T} \,. \tag{2}$$

The components $z_h$ $(h = 1, \dots, N_z)$ of the vectors $\mathbf{z}(t)$ are called **channels**. As described above, they represent both data coming from sensors (**passive** channels) and from actuators (**active** channels). This induces a natural split of (2) into two records composed of $N_x$- and $N_c$-dimensional vectors respectively (where $N_x + N_c = N_z$):

$$\{\mathbf{x}(t) = (x_1(t), \dots, x_{N_x}(t))'\}_{t=1,\dots,T} \,, \tag{3}$$

$$\{\mathbf{c}(t) = (c_1(t), \dots, c_{N_c(t)})'\}_{t=1,\dots,T} \,. \tag{4}$$

These time series describe the state of the vehicle and the control performed by the driver, respectively. We suppose that the vectors $\mathbf{x}(t)$ and $\mathbf{c}(t)$ take values in subsets $X \subset \mathbb{R}^{N_x}, C \subset \mathbb{R}^{N_c}$. In our experiments we collected data from $N_x = 12$ sensors and $N_c = 3$ actuators.

After each simulation, the driver who performed it was asked to answer a predefined survey

$$Q = \{q_1, q_2, \dots, q_{N_q}\} \tag{5}$$

composed of $N_q$ questions about selected handling properties (e.g., the stability of the vehicle during overtaking manoeuvres). For each question, the answer $y$ must be chosen amongst $N_y$ possible options. In our experiments the survey contained $N_q = 22$ questions, whereas the answers took values in the finite **scores space**

$$Y = \{1, 1.5, \dots, 9.5, 10\} \tag{6}$$

($N_y = 19$). A higher score reflects a higher quality of the corresponding handling property.

Due to a non-disclosure agreement, the precise nature of the synthesis parameters, the channels and the specific experimental questions asked to the drivers can not be reported.

With this formalisation and assuming $N_q = 1$ (which amounts to assuming the $N_q$ components of the scoring are i.i.d.), we model a driver $d$ as a measurement instrument $f_d$ used to evaluate a vehicle $\mathbf{r}$:

$$\begin{aligned} f_d \,:\, R &\to Y \\ \mathbf{r} &\mapsto y\,. \end{aligned}$$

Our experience shows that the evaluation process is non-deterministic: even the same driver could give different feedbacks about the same vehicle setting $\mathbf{r}$ depending on the driving experience, personal biases, and even the state of mind. Therefore, it is more natural to represent $f_d$ as a map

$$\begin{aligned} f_d \,:\, R &\to \mathcal{P}(Y) \\ \mathbf{r} &\mapsto \mathbb{P}(Y) \end{aligned}$$

that outputs (discrete) probability distributions over the scores space. Here,

$$\mathbb{P}(Y) = (p(y_1), p(y_2), \ldots, p(y_{N_y}))'$$

denotes a probability mass function (PMF) taken from the space $\mathcal{P}(Y)$ of all such functions over $Y$.

Humans are noisy controllers in the driving feedback loop. This implies that even if the vehicle setting $\mathbf{r}$, the driver $d$ and the test circuit are the same, the records (3) and (4) will likely differ. Thus, the state record is modelled as a random variable distributed according to the conditional probability $p(\mathbf{x}|\mathbf{r}, d)$.

The second factor that needs to be incorporated into the model is the subjectivity of the scores. When the driver is asked to evaluate a driving session, a personal evaluation scale and the individual state of mind can inform the decision. For example, the same vehicle $\mathbf{r}$ could be assigned a 7.0 by a driver $d_1$ and a 5.5 by a second driver $d_2$. Another example: suppose that a driver has assigned the latest car setting he tested, $\mathbf{r}_1$, a 6.5, but that he had also tested a particularly good-performing vehicle $\mathbf{r}_2$ before trying the current setting; if $\mathbf{r}_2$ had been particularly poor-performing instead, he could be willing to assign $\mathbf{r}_1$ a 7.5. We model this subjectivity with the conditioning $p(y|\mathbf{x}, d)$.

The distributions $p(\mathbf{r})$ and $p(d)$ are assumed to be independent: in fact, the vehicles can be arbitrarily defined by engineers, whereas the population of the drivers is determined by nature and experience.

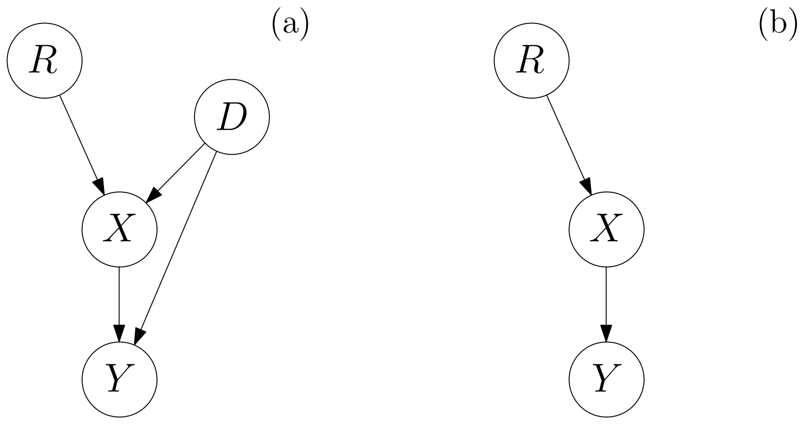


Figure 1. Graphical models describing the problem. If we suppose that the circuit is kept fixed, (a) the vehicle setting $R$ and the driver $D$ interact to generate a driving experience $X$, which is then evaluated by the driver to return a score $Y$ about the quality of the vehicle setup; (b) if we suppose the distributions of the driving experiences and the scores do not depend on the driver, then we have a simplified model where the final evaluation can be inferred directly from the driving experience.

The discriminative process carried out by the drivers can be modelled as depicted in Figure 1(a):

$$p(y, \mathbf{x}, \mathbf{r}, d) = p(y|\mathbf{x}, d)p(\mathbf{x}|\mathbf{r}, d)p(\mathbf{r})p(d)\,. \tag{7}$$

The fact that the drivers hired to evaluate vehicles are professionals is not a chance: they represent the least subjective measurement instruments available on the market.

The drawbacks of these instruments are their scarcity and their cost. Therefore, we did not have sufficient data to characterise the distribution $p(d)$. We thus simplified the model (7) by integrating over the set of all possible drivers, deriving the simplified model depicted in Figure 1(b). Its joint distribution is the following:

$$p(y, \mathbf{x}, \mathbf{r}) = \int_D p(y, \mathbf{x}, \mathbf{r}, d) = p(y|\mathbf{x})p(\mathbf{x}|\mathbf{r})p(\mathbf{r})\,. \tag{8}$$

We anyway observe that the richer model (7) could be used when scaling the analysis to *big data* collected from a larger set of drivers. The interpretation of the factors in expression (8) should clarify the scope of our work.

The term $p(\mathbf{x}|\mathbf{r})p(\mathbf{r}) = p(\mathbf{x}, \mathbf{r})$ characterises the impact of the mechanical setup on the driving experience. Automotive engineers can use their domain knowledge to tune the setup $\mathbf{r}$ to reduce the probability of certain experiences $\mathbf{x}$ happening [1]. For example, if the problem turns out to be the perceived torque on the steering wheel, engineers should modify the synthesis parameters which can impact this characteristic.

The leftmost term

$$p(y|\mathbf{x}) \tag{9}$$

is the least quantifiable, in the sense that even a professional pilot could have a hard time telling which parts of the experience $\mathbf{x}$ concurred more to the final evaluation. Typically, the pilot points out subsections of the test circuit where he believes the vehicle behaved anomalously; then, the engineers start a time-consuming and tedious analysis of the record (2) to identify the physical causes of the perceived anomalies. Thus, we focused on the development of a *deep learning* model that not only could reproduce the discriminative model (9), but could also “show” which portions of the record (3) informed its output the most.

In Section 2, we review the background literature and related research. In Section 3, we describe our proposed system. In Section 4, we analyse the output of the model. Finally, in Section 5, we discuss some additional properties of the technique and open research questions.

## 2. Related work

The problem of modelling the human driver has been extensively studied [2]. This research, though, was not directed towards modelling the relationship between the mechanical state of a vehicle and the subjective perceptions of the vehicle’s handling properties. To the best of our knowledge, no closed-form description of such a relationship is known.

**Deep learning** (DL) has gained a reputation as a powerful tool to model non-linear relationships in big data sets [3–5], and the automotive market has not remained indifferent to its effectiveness. Apart from *autonomous driving* (AD) and *advanced driver assistance systems* (ADAS), sensor data analytics has been investigated.

Driving styles clustering and classification are interesting applications for both road safety monitoring and insurance purposes. Existing research works preprocessed time series GPS data to extract information about velocities and accelerations, then used it to train *deep neural networks* (DNNs) to identify driving styles [6,7]. The chosen DL systems follow a hierarchical structure typical of neural machine translation models: the input time series is split into windows creating *paragraphs*, each of which is then summarised into a single vector using a *recurrent neural network* (RNN). This paragraph-level information is then further compressed using **autoencoders** [8], and the resulting sequences of vectors are interpreted as *documents* ready to be classified.

The first difference between these works and ours is the scope: whereas the aim of the systems above is classifying driving styles from GPS data, we analyse sensor data to understand which features of the vehicle’s mechanical state history inform the driver’s subjective perceptions.

The second difference is in the architecture of the DL system itself, which reflects our goal: we need a system which can point us to those parts of its input which are the most influential on its output. First, RNNs lack the capability of quantifying the importance of an input vector with respect to the others. Hence, we had to select alternative techniques to perform the temporal processing. In addition, we give the state vectors $\mathbf{x}(t)$ the geometric interpretation of points lying on a curve embedded in a high-dimensional space. Therefore, our system first projects these points into a lower-dimensional space using autoencoders, then feeds them to the layers responsible for temporal information processing.

Dimensionality reduction studies unsupervised learning techniques that can discover low-dimensional structures in high-dimensional data. *Autoencoders* are DNNs that learn an *encoder* map $\phi_{enc} : \mathbb{R}^N \to \mathbb{R}^M$ $(M \ll N)$ to compress input vectors $\mathbf{x} \in \mathbb{R}^N$ into lower-dimensional counterparts $\mathbf{y} \in \mathbb{R}^M$. Autoencoders are trained to minimise the squared Euclidean distance

$$d^2(\mathbf{x}, \phi_{dec}(\phi_{enc}(\mathbf{x}))) = \|\mathbf{x} - \phi_{dec}(\phi_{enc}(\mathbf{x}))\|^2$$

between the original vectors and the reconstructed ones (the map $\phi_{dec} : \mathbb{R}^M \to \mathbb{R}^N$ is called a *decoder*). **Denoising autoencoders** [9] are more robust autoencoders trained to reconstruct the original vectors $\mathbf{x}$ even when they receive noisy versions $\mathbf{x} + \boldsymbol{\nu}$ in input ($\boldsymbol{\nu}$ is usually a zero-mean noise).

**Attention mechanisms** [10] are a powerful DL tool to process temporal information. Apart from the improved stability during training with respect to RNNs, attention mechanisms intrinsically define a PMF over their input sequences. An attention mechanism takes in input sequences of vectors $\{\mathbf{x}(t)\}_{t=1,\dots,T}$ and compares each element with a query vector $\mathbf{q}$ using the Euclidean inner product. These values are then passed through a softmax function yielding a PMF over the elements of the sequence:

$$p(\bar{t}) = \frac{\exp \langle \mathbf{q}, \mathbf{x}(\bar{t}) \rangle}{\sum_{t=1}^{T} \exp \langle \mathbf{q}, \mathbf{x}(t) \rangle}, \; \bar{t} = 1, \dots, T. \tag{10}$$

This PMF is used to perform a *soft query* against the "database" represented by the sequence, returning the weighted average

$$\bar{\mathbf{x}} = \sum_{t=1}^{T} p(t)\mathbf{x}(t).$$

Research on *convolutional neural networks* (CNNs) has introduced tools to understand which parts of the inputs to a feedforward network impact its output the most. In particular, the **saliency maps** technique [11] defines PMFs on the data structures taken as input by a given network. Suppose a feedforward network $\phi$ realises a differentiable scalar map

$$\phi : \mathbb{R}^N \to \mathbb{R}$$

on some $N$-dimensional Euclidean space, and suppose we need to know which parts of the input $\mathbf{x} \in \mathbb{R}^N$ we should change to get a *target change* $\delta \in \mathbb{R}$ in the output. If we denote by

$$\mathbf{g} := \nabla_{\mathbf{x}} \phi = \left( g_i := \frac{\partial \phi}{\partial x_i} \right)'_{i=1,\dots,N}$$

the gradient of $\phi$ with respect to $\mathbf{x}$, the saliency map of $\phi$ with respect to the target change $\delta$ is a PMF over $\{1, \dots, N\}$:

$$p(\bar{i}) = \frac{|g_{\bar{i}}|}{\sum_{i=1}^{N} |g_i|}, \; \bar{i} = 1, \dots, N. \tag{11}$$

We can combine saliency maps and attention mechanisms to output a PMF over the data structure representing (3), as we will describe in Section 3.

Other methods to simplify and visually analyse time series are available in the literature. *Piecewise aggregate approximation* (PAA) [12] partitions a continuous time series in windows and replaces them with the respective window averages, thus reducing the dimensionality of the data. As an extension of PAA, *symbolic aggregate approximation* (SAX) [13] analyses the uni- or multi-variate distribution generated by PAA and uses its quantiles to partition real variables into symbols of a finite alphabet. These transformations are used to turn real-valued time series into strings which can be analysed to identify recurrent substrings (corresponding to temporal features or *motifs*). *VizTree* [14] is a visualisation method designed to identify recurrent substrings in a given string.

With respect to our approach, these techniques drop the geometric interpretation of the time series, and they require training a separate classifier for the generated strings. Even though VizTree can focus the attention on recurrent motifs of the input time series, it does not return information about their importance for the classifiers trained on the series.

We remark that the problem we wanted to solve poses unique challenges. We were not able to find other techniques which could be directly applied to our task, and whose performances could be compared to our solution.

## 3. A deep learning system for the analysis of handling performances

Before describing our proposed DL system, we describe the data set we used for this project. We do this for two reasons: first, its structure motivated the modelling process described later in this section, and second, this brief discussion also clarifies certain details of the experiments which we report in Section 4.

**The data set** The data for this project was collected by a single test driver who performed all the driving simulations. We kept the circuit fixed to reduce the number of factors which could impact his evaluation.

We experimented with 24 different mechanical settings $\mathbf{r}$. For each vehicle setting, the driver performed an average of four driving experiences, collecting $K = 89$ records (2) in total. We note that the driver assigned scores only once for each of the 24 vehicle settings, after all the driving experiences; therefore, we labelled with the same scores all the records associated with a given setting.

We ran the simulator at a sampling frequency $f = 1000Hz$, and the collected records contained on average $T \sim 75000$ vectors with 12 components for sensors (passive) and 3 components for actuators (active), for a total amount of approximately 6.5 million 15-dimensional vectors.

An *end-to-end* machine learning system would likely overfit on such a data set, due to the unbalance between the high dimension of the input space and the small number of available labelled points. Therefore, we needed to design suitable features to make the final classifier more robust.

Not all the driving experiences had the same duration, and not all of them represent complete laps on the test circuit. To simplify the design of our model, we defined the following **homogenisation** and **selection** operations on the records.

To each vector $\mathbf{z}(t)$, the driving simulator attaches information about the relative position of the vehicle with respect to the test circuit's starting line, in the form of a curvilinear abscissa $\varsigma(t)$. Since the test circuit had a fixed length, we set fixed sampling positions $\varsigma_1 < \ldots < \varsigma_S$, called the **waypoints**. Given a record $\{\mathbf{z}(t)\}_{t=1,\ldots,T}$ and a waypoint $\varsigma_s, s \in \{1, \ldots, S\}$, we defined

$$\mathbf{z}(s) \coloneqq (1-\lambda)\mathbf{z}(t(s)) + \lambda \mathbf{z}(t(s)+1)\,,$$

where

$$t(s) \coloneqq t \in \{1, \ldots, T\} \mid \varsigma_s \in [\varsigma(t), \varsigma(t+1))\,,$$
$$\lambda \coloneqq \frac{\varsigma_s - \varsigma(t(s))}{\varsigma(t(s)+1) - \varsigma(t(s))}\,.$$

With this procedure, we were able to trasform the **original** (time-aligned) records

$$\{\mathbf{z}^{(k)}(t)\}_{t=1,\ldots,T^{(k)}}\,,\; k = 1, \ldots, K$$

into **homogenised** (waypoint-aligned) records

$$\{\mathbf{z}^{(k)}(s)\}_{s=\varsigma^{(k)}_{start},\dots,\varsigma^{(k)}_{end}}\,,\ k = 1,\dots,K\,. \tag{12}$$

Here, $k = 1, \dots, K$ indexes the collected records. The homogenised state records (3) and homogenised control records (4) can be redefined accordingly.

Given two waypoints $\varsigma_A, \varsigma_B \,|\, 1 \leq \varsigma_A < \varsigma_b \leq S$ which delimit a specific portion of the test circuit, it is possible to identify all and only the homogenised records (12) corresponding to those $k \in \{1, \dots, K\}$ such that

$$\varsigma^{(k)}_{start} \leq \varsigma_A < \varsigma_B \leq \varsigma^{(k)}_{end}\,.$$

We call this process a **selection** of the records which describe the circuit portion delimited by $\varsigma_A$ and $\varsigma_B$.

To simplify the notation, we will use the indexing in $T$ and talk of *original records* and *homogenised records* to distinguish between the two.

We used this homogenisation and selection operations extensively during the modelling and experimental activities, as we will report below.

**The target model** Let $T$ denote a positive integer and $X \subset \mathbb{R}^{N_x}$ a subset of the $N_x$-dimensional Euclidean space. We denote by $X^0 = (X)^T$ the set of finite trajectories $\{\mathbf{x}(t)\}_{t=1,\dots,T}$ of length $T$ sampled from the space $X$. Note that these vector sequences can be represented as a collection of $N_x$ scalar sequences

$$\{x_h(t)\}_{t=1,\dots,T}\,,\ h = 1,\dots,N_x\,.$$

We denote by

$$U = \{(h,t)\,|\,h = 1,\dots,N_x\,,\ t = 1,\dots,T\} \tag{13}$$

the **index space** that identifies their components. Let $Y$ denote a set of scores (6).

We want to build a system that, given a state record (3), outputs a PMF over $Y$ and a PMF over $U$:

$$\begin{aligned} \Phi\,:\, X^0 &\to \mathcal{P}(Y) \times \mathcal{P}(U) \\ \{\mathbf{x}_t\} &\mapsto (\mathbb{P}(Y), \mathbb{P}(U))\,. \end{aligned} \tag{14}$$

As previously observed, the very small number of labelled data is the critical issue we need to be aware of when designing the model. Increasing either *naturally* , i.e., collecting additional data points, or *artificially*, i.e., applying some generative model to create synthetic data points, might seem a mandatory task to increase the robustness of the results, but in both cases there are practical and theoretical shortcomings. As concerns the collection of additional data points, one should develop a virtual model of the car, design a software capable of running the simulation, and hire one or more professional pilots to collect feedback about the vehicle. These requirements would lead to a notable budget (the cost of each data point is in the order of thousands or tens of thousands of euros, and the cost of the data set scales linearly in the number of data points) and excessively long time constraints. Moreover, generative models such as generative adversarial networks (GANs) are trained under the supervised learning paradigm, not the unsupervised one. Thus, training a generative model requires a corpus of observed natural data points that is sufficient to approximate the unknown probability distributions, and this is not our case for the reasons highlighted before.

As a result of these considerations, we designed a model able to exploit the structure and semantics of the (few) available data points. In particular, the DL system we designed to learn $\Phi$ is composed of three stages:

1. a trajectory condenser;
2. a local trajectory analyser;
3. a global trajectory classifier.

**Trajectory condenser** The records (3) are composed of 12-dimensional vectors $\mathbf{x}(t)$. We preprocessed each vector using the standard affine transformation

$$\tilde{\mathbf{x}}(t) \coloneqq \Sigma^{-1}(\mathbf{x}(t) - \boldsymbol{\mu}) \,,$$

where $\boldsymbol{\mu}$ is the component-by-component average over any example in the training set and over any time, and $\Sigma$ is the diagonal matrix whose element $\sigma_{hh}$ represents the standard deviation of the $h$-th component of such vectors. In the following, to simplify the notation, we suppose the vectors $\mathbf{x}(t)$ are already normalised and thus identify $\tilde{\mathbf{x}}(t) = \mathbf{x}(t)$.

The trajectory condenser should map trajectories in the $N_x$-dimensional Euclidean space into trajectories in the $N_e$-dimensional Euclidean space, where $N_e \ll N_x$:

$$\varphi^1 \,:\, (\mathbb{R}^{N_x})^T \to (\mathbb{R}^{N_e})^T.$$

To this end, we trained a denoising autoencoder

$$\phi^1_4 \circ \phi^1_3 \circ \phi^1_2 \circ \phi^1_1 \,:\, \mathbb{R}^{N_x} \to \mathbb{R}^{N_x}$$

($\circ$ denotes composition), where $\phi^1_{enc} \coloneqq \phi^1_2 \circ \phi^1_1$ is the encoding map and $\phi^1_{dec} \coloneqq \phi^1_4 \circ \phi^1_3$ is the decoding map. The layers corresponding to the maps $\phi^1_1, \phi^1_3$ contain 10 neurons each. To set the dimension $N_e$ of the *encoding space* (i.e., the number of neurons in the layer corresponding to $\phi^1_2$), we applied a technique from the field of manifold learning to our data set and got an estimate of $N_e = 4$ [15]. First, the technique sub-samples the available data sets with different sampling frequencies; then, for each sub-sample and candidate dimension, it uses a corresponding bounded support kernel function to collect statistics associated with the given sub-sample at a properly-chosen scale; finally, it estimates the dimension as the candidate dimension for which the collected statistics are most stable across different scales. Since the underlying hypothesis of this technique are hard to test in practice, we used it to establish a reasonable prior for the dimension of the encoding space, as opposed to guessing it or trying out all the possible dimensions, but we resorted to the autoencoder loss as an indirect measure of the estimate's quality. All the layers except the last are activated by the hyperbolic tangent function.

To create the data sets needed to train this autoencoder, we used all the vectors $\mathbf{x}(t)$ contained in the available original records $\{\mathbf{x}^{(k)}(t)\}_{t=1,\dots,T}, k = 1, \dots, K$. We put 80% of the available vectors in the training set, creating pairs $(\mathbf{x}(t)+\boldsymbol{\nu}, \mathbf{x}(t))$ where $\boldsymbol{\nu}$ denotes a noise vector whose components were sampled from a univariate Gaussian distribution with $\mu = 0$ and $\sigma = 0.01$. We used the remaining 20% of the vectors as a validation set to perform *early stopping* of the training algorithm.

We trained the network minimising the mean square error (MSE) loss, applying the Adam optimization algorithm [16] with a learning rate $\eta = 0.001$ and a batch size $B = 25$ for $n = 500$ epochs.

The trajectory condenser

$$\begin{aligned} \varphi^1 \,:\, X^0 &\mapsto X^1 \\ \{\mathbf{x}(t)\}_{t=1,\dots,T} &\mapsto \{\mathbf{x}^1(t) \coloneqq \phi^1_{enc}(\mathbf{x}(t))\}_{t=1,\dots,T} \end{aligned} \tag{15}$$

is obtained by a pointwise application of the encoder to all the points of the trajectory. Here, $X^1 \subset (\mathbb{R}^{N_e})^T$. The output sequence

$$\{\mathbf{x}^1(t)\}_{t=1,\dots,T} \tag{16}$$

is called the **condensed trajectory**.

Let $J \coloneqq (\mathbf{g}_j \coloneqq \nabla_{\mathbf{x}}\phi^1_{enc,j})_{j=1,\dots,N_e}$ denote the Jacobian of $\phi^1_{enc}$. Since $\mathbf{g}_j$ describes how changes in the input $\mathbf{x}(t)$ affect $\phi^1_{enc}$'s $j$-th component, we can weight the columns of $J$ using the actual changes $\delta(t) \coloneqq \mathbf{x}^1(t+1) - \mathbf{x}^1(t)$ that occur in the feature space, obtaining the *directional gradient*

$$\mathbf{g} = J\delta(t) \,.$$

At this point, we apply the saliency maps technique (11) to obtain a PMF over $\{1,\dots,N_x\}$:

$$p_{\phi^1_{enc}}(\bar{h}) = \frac{|g_{\bar{h}}|}{\sum_{h=1}^{N_x}|g_h|},\ \bar{h} = 1,\dots,N_x\,, \tag{17}$$

where $g_h$ denotes the $h$-th component of $\mathbf{g}$.

The intuition behind this choice is that the directional gradient should provide information about which changes in the space of the passive channels $X^0$ are deemed relevant by the autoencoder and therefore mapped to the feature space $X^1$.

**Trajectory analyser** Discussions with the driver suggested that his final evaluations are based on a sequence of perceptions collected during the test, plus some global contextual information (e.g., the time required to complete a lap on the test circuit). We thus modelled the evaluation task (i.e., the assignment of a score to a vehicle) as a *supertask* accomplished using information collected while performing simpler *subtasks*, hierarchically.

The natural subtasks we selected are the controls $\mathbf{c}(t)$ actuated by the driver on the vehicle: we assumed that the information used by the driver to control the vehicle informs his final evaluation. This hierarchical composition of information, from subtasks to supertasks, has already been applied successfully to document classification using hierarchical attention mechanisms [17].

We preprocessed the homogenised control records (4) to create the training targets. The controls $\mathbf{c}(t)$ are 3-dimensional vectors representing the steering angle, the amount of pressure on the brake pedal and the amount of pressure on the throttle pedal. Since the brake pedal and the throttle pedal are used in a mutually exclusive way, we combined them into a single *acceleration control* variable ranging in the continuous interval $[-1,+1]$, with $-1$ corresponding to *full brake* and $+1$ corresponding to *full throttle*. In the same way, the steering angle variable was scaled to take values in the interval $[-1,+1]$, with $-1$ corresponding to *full left* and $+1$ corresponding to *full right*. In this way, the control $\mathbf{c}(t)$ was represented as a two-dimensional vector taking values in the set $[-1,+1]^2 \subset \mathbb{R}^2$.

If $T$ denote the length of the condensed trajectory (i.e., the number of waypoints considered), we denote by $W < T$ an integer *window size* such that $T = LW$, with $L$ a positive integer. This choice allows to partition the condensed trajectory (16) into $L$ non-intersecting contiguous subsequences called **windows**:

$$\boldsymbol{v}_\ell = \{\mathbf{x}^1((\ell-1)W+1),\dots,\mathbf{x}^1(\ell W)\}\,,\ \ell = 1,\dots,L\,. \tag{18}$$

The intuition is that a window should provide a description of the state of the automobile on the localised sequence of waypoints $\{\varsigma_{(\ell-1)W+1},\dots,\varsigma_{\ell W}\}$.

We trained a predictive model

$$\phi^2_2 \circ \phi^2_1 \,:\, (\mathbb{R}^{N_e})^W \to [-1,+1]^2 \tag{19}$$

to mimic how the driver controls a car depending on its mechanical state. The first map

$$\begin{aligned} \phi^2_1 \,:\, (\mathbb{R}^{N_e})^W &\to \mathbb{R}^{N_e} \\ \boldsymbol{v}_\ell \mapsto \mathbf{x}^2(\ell) &= \sum_{t=(\ell-1)W+1}^{\ell W} p_{\phi^2_1}(t)\mathbf{x}^1(t) \end{aligned} \tag{20}$$

is an attention mechanism (10) parametrised by $\mathbf{q}^2_1 \in \mathbb{R}^{N_e}$, whereas

$$\begin{aligned} \phi^2_2 \,:\, \mathbb{R}^{N_e} &\to [-1,+1]^2 \\ \mathbf{x}^2(\ell) &\mapsto \tilde{\mathbf{c}}(\ell W+1) \end{aligned}$$

is a hyperbolic tangent-activated regression layer trained to predict the control $\mathbf{c}(\ell W+1)$ actuated by the driver at the immediately successive waypoint with respect to the analysed window $\boldsymbol{v}_\ell$.

To create the data sets needed to train this network, we first passed all the homogenised records $\{\mathbf{x}^{(k)}\}_{t=1,\dots,T}, k = 1,\dots,K$ through the trajectory condenser (15), obtaining condensed trajectories $\{\mathbf{x}^{1(k)}(t)\}_{t=1,\dots,T}, k = 1,\dots,K$. Then, we selected 80% of these condensed trajectories to create the training set and used the remaining 20% to create the validation set. From each condensed trajectory, we extracted all the windows of length $W$ with unconstrained extrema,

$$\boldsymbol{v}_{\bar{t}} \coloneqq \{\mathbf{x}^{1(k)}(t)\}_{t=\bar{t}+1,\dots,\bar{t}+L}$$

(i.e., we did not limit $\bar{t} = (\ell-1)W\,,\ \ell \in \{1,\dots,L\}$ as it is done at inference stage). We attached the appropriate control labels $\mathbf{c}(\bar{t}+L+1)$, to both training and validation samples, and used the validation set to perform early stopping.

We minimised the MSE loss applying the Adam optimization algorithm with a learning rate $\eta = 0.001$ and a batch size $B = 10$ for $n = 20$ epochs. We set a window size $W = 10$, determined empirically.

The trajectory analyser is realised applying (20) to every window that partitions the condensed trajectory:

$$\begin{aligned} \varphi^2 &: X^1 \mapsto X^2 \\ \{\mathbf{x}^1(t)\} &\mapsto \{\mathbf{x}^2(\ell)\} \coloneqq \{\phi_1^2(\boldsymbol{v}_1), \phi_1^2(\boldsymbol{v}_2), \dots, \phi_1^2(\boldsymbol{v}_L)\}\,, \end{aligned} \tag{21}$$

being $X^2 \subset (\mathbb{R}^{N_e})^L$. The sequence

$$\{\mathbf{x}^2(\ell)\}_{\ell=1,\dots,L} \tag{22}$$

is called the **compressed trajectory**. The intuition is that these features should encode some of the information used by the driver to control the car along the test circuit.

Note that the application of (20) to the windows (18) generates $L$ PMFs over $\{1,\dots,W\}$, one for each window that partitions the condensed trajectory:

$$\{p_{\phi_1^2}(t)\,,\ t = (\ell-1)W+1,\dots,\ell W\},\ \ell = 1,\dots,L\,. \tag{23}$$

In this way, a local *importance measure* is assigned to each vector of the condensed trajectory taken as input.

**Trajectory classifier** The standard way to train a classifier using DNNs is turning the classification problem into a *multinomial logistic regression*. To this end, class labels $y_j \in Y$ are turned into points of the $N_y$-dimensional simplex using the so-called *one-hot encoding*, and the model is then trained as a regression model on the simplex. One-hot encoding a class label $y_j \in Y$ is performed by transforming it into a $N_y$-dimensional vector whose $j$-th component takes value one, whereas the others are set to zero. This transformation is equivalent to defining a PMF $\mathbb{P}(Y)$ which assigns probability one to the $j$-th class and zero to the others.

The goal of the trajectory classifier is using the local information contained in compressed trajectories (22) to generate probability distributions $\mathbb{P}(Y)$ over $Y$:

$$\begin{aligned} \varphi^3 &: (\mathbb{R}^{N_e})^L \to \mathcal{P}(Y) \\ \{\mathbf{x}^2(\ell)\} &\mapsto \mathbb{P}(Y) = (p(y_1), p(y_2), \dots, p(y_{N_y}))\,. \end{aligned} \tag{24}$$

Similarly to (19), we implemented it using a two-layer neural network:

$$\phi_2^3 \circ \phi_1^3 : (\mathbb{R}^{N_e})^L \to \mathcal{P}(Y)\,.$$

The map

$$\begin{aligned} \phi_1^3 &: (\mathbb{R}^{N_e})^L \to \mathbb{R}^{N_e} \\ \{\mathbf{x}^2(\ell)\} &\mapsto \mathbf{x}^3 = \sum_{\ell=1}^{L} p_{\phi_1^3}(\ell)\mathbf{x}^2(\ell) \end{aligned} \tag{25}$$

is an attention mechanism (10) parametrised by $\mathbf{q}_1^3$. The second map is instead a classification layer:

$$\begin{aligned}\phi_2^3 : \mathbb{R}^{N_e} &\to \mathcal{P}(Y)\\ \mathbf{x}^3 &\mapsto \operatorname{softmax}(\mathbf{w}_2^3\mathbf{x}^3 + \mathbf{b}_2^3)\,,\end{aligned}$$

where $\mathbf{w}_2^3 \in \mathbb{R}^{N_y \times N_e}$ is the weight matrix, and $\mathbf{b}_2^3 \in \mathbb{R}^{N_y}$ is the bias vector.

The data sets used to train the trajectory classifiers are described more in-depth in Section 4.

In all our experiments, we trained the trajectory classifiers using the cross-entropy (CE) loss and applying the Adam optimization algorithm with a learning rate $\eta = 0.001$ for $n = 350$ epochs. Note that the application of (25) returns a probability distribution over $\{1, \dots, L\}$:

$$p_{\phi_1^3}(\ell)\,,\ \ell = 1, \dots, L\,. \tag{26}$$

In this way, each window that partitions the condensed trajectory is assigned a relative importance with respect to the others, quantifying its contribution to the final classifier.

**Heatmap construction** The second function that (14) should implement is the generation of a probability distribution over the coordinates space (13) associated with the record data structure (3).

For every waypoint $t = 1, \dots, T$, let $\ell(t)$ denote the window to which the corresponding vector $\mathbf{x}^1(t)$ belongs:

$$\ell(t) := \ell \in \{1, \dots, L\} \,|\, (\ell - 1)W + 1 \leq t \leq \ell W\,.$$

We combine (17), (23) and (26) to get the **heatmap**:

$$p((h,t)) := p_{\phi^1}(h)\,p_{\phi_1^2}(t)\,p_{\phi_1^3}(\ell(t))\,,\ (h,t) \in U\,. \tag{27}$$

The resulting map $\Phi := \varphi^3 \circ \varphi^2 \circ \varphi^1$, obtained composing (15), (21) and (24) is depicted in Figure 2.

We end this section with few considerations about lack of data and possible overfitting. Although we are aware that only 24 labelled data points have been provided for our analysis, each data point has a complex structure made up by a large number of observations, and this is exploited by the trajectory compressor. Note that to train a denoising autoencoder we use each (noisy) input training point as its own label, in what we could call a “self-supervised” fashion. Since we can create a training pair $(t + \varepsilon, t)$ for each observation in all the available trajectories, we have access to a data set containing approximately 5M observations, whereas the denoising autoencoder uses less than 400 parameters. Moreover, since the trajectory compressor operates on each observation independently of the neighbouring ones, the interpretation of the input sequence as a trajectory translates also into the abstract feature space (i.e., the encoding space of the autoencoder). Since the trajectory compressor maps time series to a lower-dimensional space, the number of parameters required to process a sub-sequence of given length is reduced by the compression factor of the autoencoder.

Then, each trajectory in the feature space is cut into consecutive windows, which are processed again by a unique feature extractor model. In this way, we start mixing temporal information while sharing the parameters across different time windows. Since each observation $t$ carries information about both passive and active channels, we can use the values of the active channels as fine-grained labels for supervised training (as opposed to the unique score label $y$ associated with the record $x$ from which the observation $t$ is taken). In our case, by considering windows of length $L = 10$, we had access to hundreds of thousands of windows, each of which was labelled with the driver’s controls at the end of that sub-trajectory. Given that the trajectory analyser uses just a few tens parameters, the risk of overfitting should be low.

## 4. Experimental results

To turn the synthesis parameters (1) into virtual vehicle models, we used a vehicle modelling custom tool developed internally by Maserati. Our professional driver performed the driving experiences using the industrial VI-grade Static Simulator [18]. We implemented, trained and tested our networks using the TensorFlow framework [19].

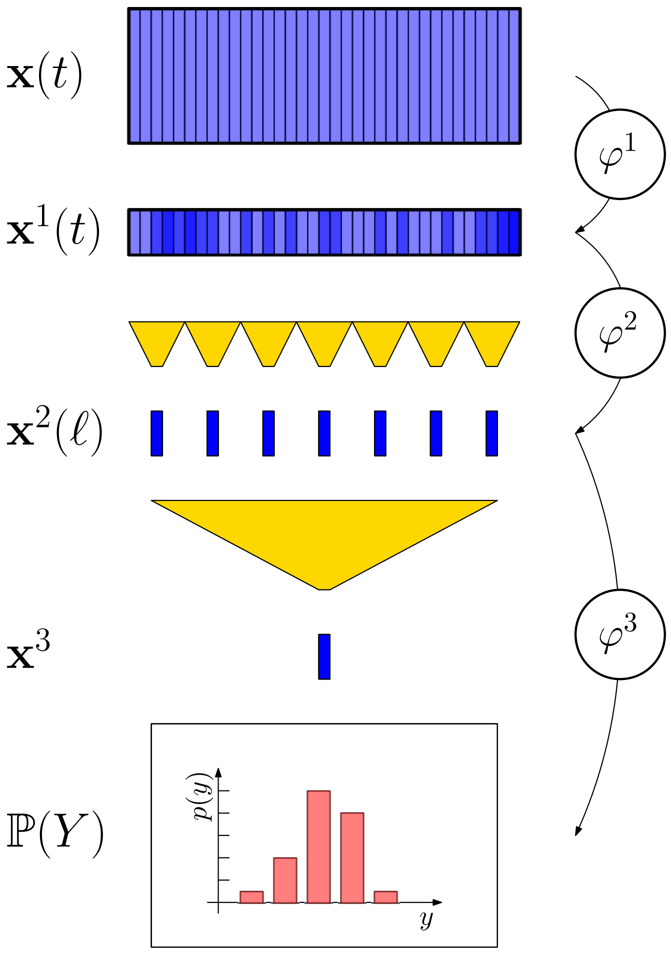


Figure 2. The proposed deep learning system for the analysis of handling properties. The trajectory condenser $\varphi^1$ reduces the size of each point in the input trajectory (the first blue rectangle); the trajectory analyser $\varphi^2$ summarises *local* portions of the driving experience into fewer feature vectors; the trajectory classifier $\varphi^3$ outputs a probability mass function $\mathbb{P}(Y) = (p(y_1), p(y_2), \dots, p(y_{N_y}))$ over the scores space, representing a *global* evaluation of the driving experience. Yellow trapezoids represent attention mechanisms.

**Model parameters** In the following, we provide a summary on the computational details of our deep learning models. In details, the autoencoder uses 356 parameters, 174 for the encoder (i.e., the trajectory compressor) and 182 for the decoder:

- $\phi^1_1$ has a $10 \times 12$ weight matrix and a 10-component bias vector;
- $\phi^1_2$ has a $4 \times 10$ weight matrix and a 4-component bias vector;
- $\phi^1_3$ has a $10 \times 4$ weight matrix and a 10-component bias vector;
- $\phi^1_4$ has a $12 \times 10$ weight matrix and a 12-component bias vector.

The second model uses 14 parameters:

- $\phi^2_1$ (i.e., the trajectory analyser) has a 4-component query vector;
- $\phi^2_2$ has a $2 \times 4$ weight matrix and a 2-component bias vector.

The last model (i.e., the trajectory classifier) uses $4 + (4 + 1)N_y$ parameters:

- $\phi^3_1$ has a 4-component query vector;
- $\phi^3_2$ has a $N_y \times 4$ weight matrix and a $N_y$-component bias vector.

The data sets used to train the different models have the following characteristics:

- autoencoder: $\sim$ 5.3M points for the training set, $\sim$ 1.3M points for the validation set; each input

and label is a 12-dimensional vector;
- second model: $\sim$ 380k points for the training set, $\sim$ 95k points for the validation set; each input is a $(4 \times 10)$-dimensional array, whereas each label is a 2-dimensional vector;
- last model: $\sim$ 20 points for the training set, $\sim$ 2/3 points for the validation set; the size of the training set is increased by a factor of 20 by means of output augmentation, yielding $\sim$ 400 points; each input is a $(4 \times 1000)$-dimensional array, whereas each label is a $N_y$-dimensional vector.

Note that the last model is trained multiple times, once for each cross-validation fold.

**Model training** At the beginning of Section 3 we reported that the records (3) which describe driving experiences were not scored independently. Instead, records related to the same vehicle setting $\mathbf{r}$ share the same scores. Therefore, when creating the data sets to train and validate the trajectory classifiers, we deemed it important to partition the records in such a way that records describing the same vehicle setting did not appear simultaneously in the training set and in the validation set.

We also reported that we were able to test only 24 vehicle settings. Combining these two observations (the small number of tested settings and the statistical requirement of not mixing observations about the same setting), we decided to avoid creating a test set, and instead used cross-validation [20] to estimate the performances of our system. Therefore, we applied the following procedure to the experiments we will describe below.

First, we split the vehicle settings in $D = 10$ **folds**. If we denote by $\bar{d} \in \{1, \dots, D\}$ a given fold, $X^{(\bar{d})}$ denotes the collection of the homogenised records $\mathbf{x}^{(k)}$ related to the vehicles belonging to the $\bar{d}$-th fold, whereas $\overline{X}^{(\bar{d})} := \bigcup_{d \neq \bar{d}} X^{(d)}$ denotes the collection of the homogenised records related to vehicles not belonging to the $\bar{d}$-th fold.

Then, in each experiment and depending on the goal of the experiment, we identified a portion of the test circuit on which to train and validate our system. The portions were delimited by waypoints $\varsigma_A, \varsigma_B \,|\, 1 \leq \varsigma_A < \varsigma_B \leq LW$. We identified window indices

$$\begin{aligned} \ell_A &:= \underset{\ell \in \{1,\dots,L\}}{\arg\max} \{(\ell - 1)W + 1 \leq \varsigma_A\} \,, \\ \ell_B &:= \underset{\ell \in \{1,\dots,L\}}{\arg\min} \{\varsigma_B \leq \ell W\} \,, \end{aligned}$$

and defined

$$\begin{aligned} \varsigma_{start} &:= (\ell_A - 1)W + 1 \,, \\ \varsigma_{end} &:= (\ell_B)W \,, \end{aligned}$$

so that from each fold $X^{(d)}$ we could select only the records covering all the waypoints $\varsigma$, $\varsigma_{start} \leq \varsigma \leq \varsigma_{end}$.

This selection operation was such that the application of the trajectory condenser (15) and the trajectory analyser (21) always returned compressed trajectories (22) of fixed length $\ell_B - \ell_A + 1$. Note that (17) and (23) are computed for all the records by these two stages of the system.

Finally, to train and validate the classifiers, we took the following steps. For each fold $d = 1, \dots, D$,

1. we performed data augmentation on the labels of the records in $\overline{X}^{(d)}$, to increase the number of available training points;
2. we trained a trajectory classifier (24) on top of this set;
3. we collected the desired statistics (accuracy and heatmaps (27)) on the outputs corresponding to the records in $X^{(d)}$.

To perform the label augmentation on the records in a given fold's training set $\overline{X}^{(d)}$ we reasoned as follows. Although professional pilots are the least subjective measurement instruments to assess the handling properties of a new car prototype, they still suffer from subjective biases and uncertainties.

Therefore, we applied a small amount of additive noise to the original scores. Introducing a small amount of noise into the labels has been shown to be beneficial when training supervised learning models [21].

For each record/score pair $(\mathbf{x}, y)$ in the training set, we sampled 20 alternative labels from a truncated univariate Gaussian distribution with extrema $a = 1.0$ and $b = 10.0$, using the score $y$ as the mean and a standard deviation $\sigma = 0.2$ (this parameter was kept small to account for the relative precision of the drivers as measurement instruments). The sampled values were then rounded to the nearest elements in the scores set (6), and turned into a one-hot encoded binary vector.

**Results** In Section 1, we reported that the driver had to answer a survey (5) composed of $N_q = 22$ questions after each simulation. We refer to each question $q \in Q$ as to a **handling property**. Remember that every handling property admits answers in the scores set (6).

In our first experiment, we wanted to investigate the accuracy of the system and test its interpretability. Therefore, we selected a specific question $\bar{q} \in Q$ and asked the driver to identify the portion of the test circuit that he thought was the most important for evaluating this specific handling property. This portion was determined by its extremal waypoints $1 < \varsigma_A < \varsigma_B < LW$ (i.e., it did not cover the complete circuit). Then, we trained trajectory classifiers $\varphi^{3(\bar{q},d)}$, $d = 1, \dots, D$, for the handling property $\bar{q}$ on the resulting reduced condensed trajectories.

The accuracy estimated by cross-validation was 31.4%, which increased to 80.0% when we considered whether the driver's actual score was amongst the three most probable scores predicted by the model. As a comparison, a random guess in the range of the most common scores (from 5.5 to 8.5, for a total of seven classes) would be correct around 15% of the times, rising to 45% if we had three guesses. These figures would drop to around 5% and 15%, respectively, for guesses on the complete scores set (6): the performance of the classifier is remarkable.

To give an idea of the output of our system, in Figure 3 and Figure 4 we report the output for an example validation point. Figure 3 depicts the prediction on the scores set (6). For each score class $y \in Y$, the model outputs the probability $p(y)$ that the input driving experience $\{\mathbf{x}(t)\}_{t=1,\dots,T}$ will be given score $y$: the higher the probability, the more confident the model is that the record will be given that score. Figure 4 depicts instead the application of the heatmap output (27) generated by the model to interpret and understand the score prediction: the higher the value of the heatmap, the more vivid an array value will be in Figure 4(d). More vivid colours indicate that the corresponding input values contributed more to the feature vector used by the trajectory classifier to predict the scores.

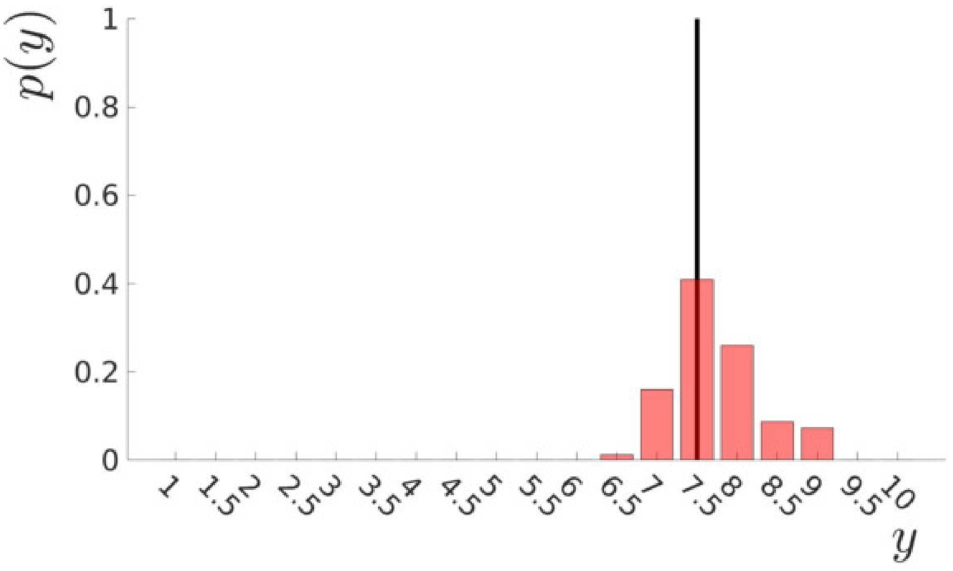
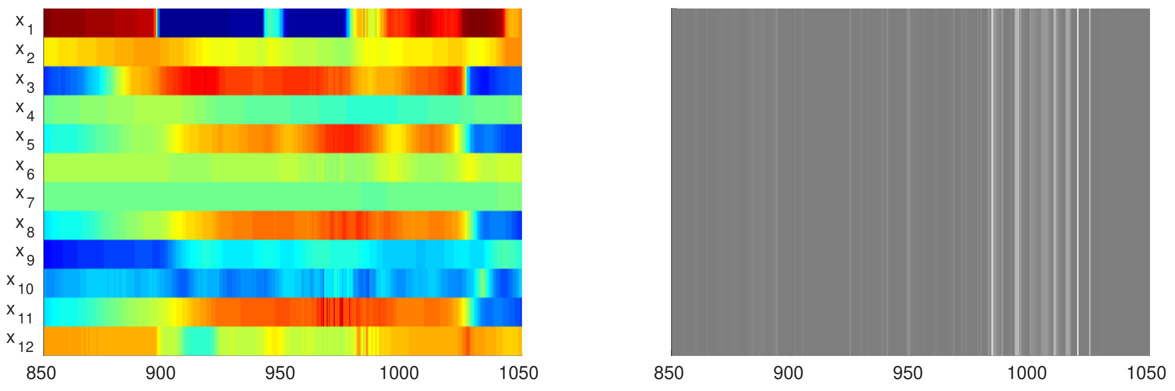


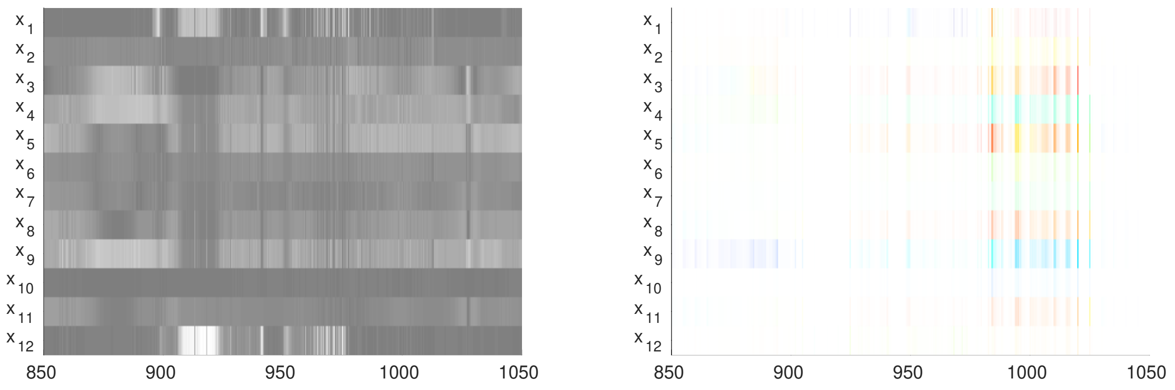
Figure 3. The model using the trajectory classifier $\varphi^{3(\bar{q},d)}$ is pretty confident that the input record will be given a score ranging from 7 to 8 (large bars). The driver actually scored the handling property $\bar{q}$ for this setting with a 7.5 (thin bar).

(a)



(b)

(c)



(d)

Figure 4. An example heatmap output of the model using the trajectory classifier $\varphi^{3(\bar{q},d)}$. This heatmap was computed on the same data point for which the scores predictions are depicted in Figure 3. (a) The array data structure describing sensors data: different lines represent different physical quantities, with lower values represented in blue and higher values in red; different columns represent samples at different waypoints. (b) The PMFs (23) and (26) computed by the attention modules and (c) the saliency maps associated with the autoencoder are combined to get a heatmap (27). Bright cells indicate high importance of the corresponding component, whereas dark cells indicate low importance of that component. The heatmap can be multiplied pointwise with the input array, yielding (d) the final visual aid to understand which waypoints and which physical quantities contributed the most to the classification. In this particular example, channels $h = 3, 4, 5, 9$ appear to be the most relevant, especially at waypoints $\varsigma$ in the ranges $\{875, \ldots, 900\}$ and $\{990, \ldots, 1025\}$.

With the second experiment, we wanted to understand which physical properties are the most important for the system when it evaluates different handling properties. To investigate the *natural* distinctions made by the system, we trained 22 trajectory classifiers (one for each handling property) using the same input records.

We did not choose distinct portions $1 \le \varsigma_A^{(q)} < \varsigma_B^{(q)} \le LW$ for each handling property $q \in Q$. Instead, from each fold $X^{(d)}$, $d = 1, \ldots, D$, we selected the records

$$\{\mathbf{x}^{(k)}(t)\}_{t=1,\ldots,\ell W}$$

describing complete laps on the test circuit.

For each handling property $\bar{q} \in Q$, we then proceeded as follows. For each fold $d = 1, \ldots, D$, we trained a trajectory classifier $\varphi^{3(\bar{q},d)}$ on $\overline{X}^{(d)}$ and computed the heatmaps (27) for all the points in the corresponding validation set $X^{(d)}$:

$$p^{(\bar{q},d,k)}(h,t) \,|\, \mathbf{x}^{(k)} \in X^{(d)} \,.$$

We then computed the PMFs over the set of channels $\{1, \ldots, N_x\}$ by reducing over the waypoints (temporal) dimension:

$$p^{(\bar{q},d)}(h) := \frac{1}{|X^{(d)}|} \sum_{\mathbf{x}^{(k)} \in X^{(d)}} \sum_{t=1}^{T} p^{(\bar{q},d,k)}(h,t)\,,\; d = 1, \dots, D\,.$$

Finally, we averaged these statistics to estimate the importance assigned by the system to the different channels for property $\bar{q} \in Q$:

$$p^{(\bar{q})}(h) := \frac{1}{D} \sum_{d=1}^{D} p^{(\bar{q},d)}(h)\,.$$

The results for all the $N_q = 22$ handling properties are reported in Figure 5. Although we cannot disclose the identity of the channels $h$ nor of the handling properties $q$, this plot shows interesting characteristics of the system. For example, the trajectory classifiers associated with the stability of the running direction tend to assign more mass to the physical quantities related to longitudinal accelerations. Instead, the trajectory classifiers associated with entering or exiting turns tend to assign more mass to the physical quantities related to vehicle speed and pitch. These and other insights resulted very helpful in the design process since they allowed to modify the mechanical setting (1) to improve multiple handling properties simultaneously, reducing the number of experimental iterations required to converge to a new car model design.

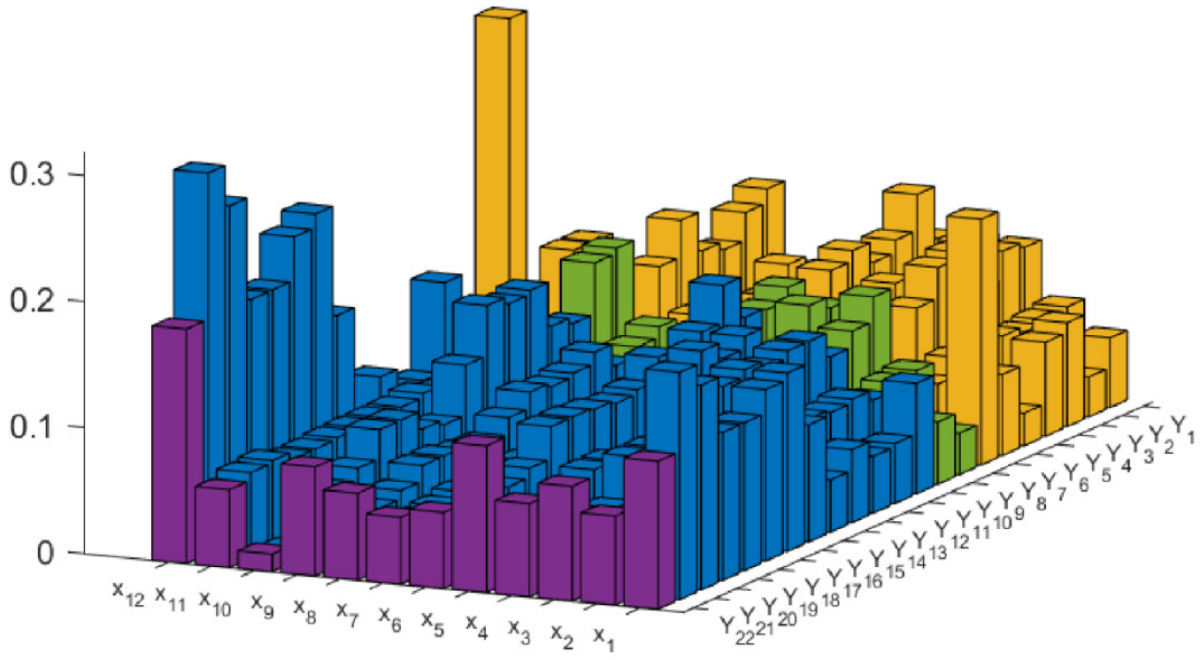


Figure 5. The trajectory classifiers built to predict different handling properties $q \in Q$ assign different importance to the passive channels $h = 1, \dots, N_x$. Different colours represent different driving regimes (e.g., driving on straightaways or performing sporty driving manoeuvres). Higher columns indicate that more mass is put by the model on the corresponding channels.

## 5. Conclusions

The luxury car market is a challenging *business-to-consumer* (B2C) market which requires its players to continually improve the user experience. Keeping the strategic initiative in such a competitive environment requires accurate, fast, and possibly cheap product development processes. Tools that allow engineers to prioritise their efforts on the most relevant tasks can accelerate the design-test-analysis development cycle.

We designed a novel tool to classify records of driving sessions while maintaining the interpretability needed to identify the physical events inducing these classifications. The amount of available labelled data was insufficient to create an *end-to-end* machine learning system without the risk of overfitting. Therefore, we designed a hierarchical feature extraction system composed by an unsupervised stage (the trajectory condenser) and a first supervised stage (the trajectory analyser). The features extracted by these two stages appeared sufficiently robust to train accurate trajectory classifiers on top of them. The modular structure also has the side effect of facilitating the transfer of intermediate representations to the analysis of different handling properties.

The system combines saliency maps and attention mechanisms to define PMFs over the input data structures. We used these functions to highlight those parts of the input sequences that impact the outputs of the model the most. We investigated the importance assigned by the trajectory classifiers corresponding to different handling properties to the measured physical quantities: the results suggest that there might be correlations amongst properties which are typically evaluated by drivers during different driving regimes.

Incorporating global information (e.g., the time required to complete a test lap) into our handling performances analysis system could help improve its performance. Indeed, our experience and recent research [22] suggest that human evaluation processes are also influenced by global contextual information. It is unclear, though, how such an integration should be performed.

We interpreted the records as samples from manifolds embedded in high-dimensional spaces. Typically, the physical state of automobiles is also conditioned on categorical variables (e.g., the gear set). We hypothesise that these variables could induce heterogeneity in the space of trajectories, making it an *ensemble* of possibly intersecting manifolds indexed by the levels of the categorical variables. If this were the case, the description of the vehicle state space, which we now obtain using a simple autoencoder, could be refined. For example, an analysis of the impact of the synthesis parameters or of different "driver archetypes" (e.g., standard consumer, amateur driver, professional driver) on the structure of these geometric descriptions could also be performed if more data were available.

We observe that our system's architecture could be applied to a broader class of tasks than the one presented in this paper. In principle, every problem which can be modelled as a hierarchical extraction of information from a time series taking values in a given Euclidean space is a suitable candidate for the methodology.

## References

1. T. D. Gillespie, *Fundamentals of Vehicle Dynamics*. Society of Automotive Engineers, 1992.
2. C. C. MacAdam, Understanding and modeling the human driver, *Vehicle System Dynamics*, vol. 40, pp. 101–134, 2003.
3. G. E. Hinton, Learning multiple layers of representation, *Trends in Cognitive Sciences*, vol. 11, pp. 428–434, 2007.
4. Y. Bengio, Learning deep architectures for AI, *Foundations and Trends in Machine Learning*, vol. 2, pp. 1–127, 2009.
5. Y. Bengio, A. Courville, and P. Vincent, Representation learning: a review and new perspectives, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1798–1828, 2012.
6. W. Dong, J. Li, R. Yao, C. Li, T. Yuan, and L. Wang, Characterizing driving styles with deep learning, *CoRR*, 2016.
7. W. Dong, T. Yuan, K. Yang, C. Li, and S. Zhang, Autoencoder regularized network for driving style representation learning, *CoRR*, 2017.
8. G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science*, vol. 313, pp. 504–507, 2006.

9. P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, Extracting and composing robust features with denoising autoencoders, in *Proceedings of the 25th International Conference on Machine Learning*, ACM, 2008.

10. D. Bahdanau, K. H. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, *CoRR*, 2014.

11. K. Simonyan, A. Vedaldi, and A. Zisserman, Deep inside convolutional neural networks: visualizing image classification models and saliency maps, *CoRR*, 2013.

12. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, *Knowledge and Information Systems*, vol. 3, pp. 263–286, 2001.

13. J. Lin, E. Keogh, L. Wei, and S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, *Data Mining and Knowledge Discovery*, vol. 15, pp. 107–144, 2007.

14. J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom, VizTree: a tool for visually mining and monitoring massive time series databases, in *Proceedings of the 30th Annual International Conference on Very Large Data Bases*, Springer, 2004.

15. M. Hein and J.-Y. Audibert, Intrinsic dimensionality estimation of submanifolds in $\mathbb{R}^d$, in *Proceedings of the 22nd International Conference on Machine Learning*, ACM, 2005.

16. D. P. Kingma and J. L. Ba, Adam: a method for stochastic optimization, *CoRR*, 2014.

17. Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, Hierarchical attention networks for document classification, in *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2016.

18. VI-grade. https://www.vi-grade.com/, 2018.

19. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: large-scale machine learning on heterogeneous systems. http://tensorflow.org/, 2015.

20. R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, ACM, 1995.

21. L. Breiman, Randomizing outputs to increase prediction accuracy, *Machine Learning*, vol. 40, pp. 229–242, 1998.

22. L. Pappalardo, P. Cintia, D. Pedreschi, F. Giannotti, and A. L. Barabási, Human perception of performance, *CoRR*, 2017.