Modeling Longitudinal Vehicle Dynamics with Neural Networks

Mauro Da Lio[a], Daniele Bortoluzzi[a] and Gastone Pietro Rosati Papini[a]

[a]Department of Industrial Engineering, University of Trento, Trento, Italy

ARTICLE HISTORY
Compiled January 10, 2020

ABSTRACT
This paper studies neural network models of vehicle dynamics. We consider both models with a generic layer architecture and models with specialized topologies that hard-wire physics principles. Network pre-wiring is limited to universal laws; hence it does not limit the network modelling abilities on one side but allows more robust and interpretable models on the other side. Four different network types (with and without pre-wired structure, recursive and non-recursive) are compared for the longitudinal dynamics of a car with gears and two controls (brake and engine). Results show that pre-wiring effectively improves the performance. Non-recursive networks also look to be preferable for several reasons.

KEYWORDS
Vehicle dynamics; Mathematical models; Neural Networks;

1. Introduction

Traditional models of vehicle dynamics are engineered from physical principles by expert scientists. The art of making such models is the art of deciding which physical phenomena are relevant for the intended use, and which may be neglected. Models created in this way predominantly take the form of parametric differential equations (for a continuous time dynamical system). The models are easily interpretable by designers and the parameters – once a model is fitted onto a specific vehicle – also characterize the vehicle itself. However, because of the trade-off between modeled and unmodeled physical phenomena, analytical models do not work well if the assumptions about neglected aspects are not met. So, to make one example, there are models for vehicle lateral dynamics that assume horizontal roads. These models will not work correctly for curves that have significant lateral elevation. Furthermore, if a parametric model is fitted onto a situation for which the unmodeled phenomena are instead important, the model will try to use its parameters to explain (to some extent and unsuccessfully) the unmodeled phenomena and, hence, the model parameters will be altered and will no longer have the intended meaning.

As another option, models based on machine learning – and for this paper neural networks – may be conceived. Recurrent neural networks (with internal states) may be used to model dynamical systems and fitted to observations of the real vehicle dynamics. Alternatively, models can be made with non-recurrent networks: in this

CONTACT D. Bortoluzzi. Email: daniele.bortoluzzi@unitn.it

case the output of the network is calculated as a function of a sufficiently long past history of the inputs.

Traditional neural networks topologies (both recurrent and non-recurrent ones) foresee the connection of input neurons to output neurons via generic hidden layers, with no physically inspired design in the connection graph. This way, albeit a network can be trained onto the input-output vehicle dynamics map, the individual weights and connections have no physical meaning and do not clarify almost anything about the genesis of the system dynamics. On the other hand, however, neural networks do not suffer from the unmodeled phenomena issue. With sufficient neurons and training examples, neural networks may in principle learn any input-output function (hence, in the above example a network modeling lateral dynamics can be further trained to model the effects of lateral slope if data become available).

The contribution of this paper concerns an approach where neural network topologies are given some degree of physics inspired pre-wired structure. Pre-wiring is limited to basic physical principles (such as that dynamics is of second-order type and forces are additive). This way, neural networks with structured topology that do not limit the modeling ability can be engineered. They are more robust than networks without such structure on one side and they are far more interpretable on the other side, hence combining the advantages of machine learning and analytic approaches.

The paper presents a comparison of four neural networks covering all the combinations of recurrent/non-recurrent and structured/non-structured network types. The case study of the longitudinal vehicle dynamics is used for demonstration, ~~which is more challenging than lateral dynamics because the longitudinal control~~which is challenging in terms of neural network structure because the longitudinal control is split between two different commands (engine torque and brake) and because the system includes a discrete state (the gear). The networks are trained and validated with experimental data. R2.1 R2.1

Application domains where learned models like these may be proficiently used are finally discussed in the conclusions.

2. Related work

Physics based vehicle dynamics modeling is a well established discipline for which many publications and textbooks are available, e.g., [1] to make just one example. Conversely, the field of artificial neural networks (ANN) received an important propulsion very recently, when algorithms for efficient network training became available and capable of training the so-called deep networks. Since then, ANN have outperformed competing machine learning approaches in an increasing number of applications [2,3].

In vehicular applications neural networks count a number of examples related to perception (in particular artificial vision by means of deep learning). Applications to vehicle dynamics and control, on the other hand, are a relatively less explored area. One notable example is a network for lane keeping that was demonstrated by NVIDIA [4] which implements end-to-end control (from camera to steering angle). The EU H2020 "Dreams4Cars" Research and Innovation Action [5] (which originated the work of the present paper) is another example where neural networks are used for the complete sensor-motor loop. In particular models of the vehicle dynamics are learned and used for vehicle control and state estimation at run-time; and they are also used offline to instantiate so called "embodied" simulations (that vaguely resemble human dreams) to synthesize motor strategies [6].

Concerning the ability of neural networks to model dynamical systems, there is a theoretical background [7] which proves that an autonomous dynamical system can be approximated by a recurrent neural network to any degree of accuracy, while in [8] this conclusion is extended to non-autonomous systems, *i.e.* subjected to an input. A comparison between three different types of vehicle longitudinal dynamics model, respectively a parametric analytic model, a generic state-space model and a neural network, has been recently provided in [9] indicating progressive improvements from analytic to neural network models.

3. Types of physical and non-physical models (for neural networks implementations)

In general, physical models of a dynamical system assume that it behaves as a causal entity which, according to a repeatable logic, converts a set of inputs into a set of outputs. In white-box models, the mathematical description of such a logic may rely on physical principles and the relevant model parameters have a physical interpretation. Conversely, black-box models are built independently of any physical consideration, and a direct interpretation of their parameters is not always possible. Intermediate situations – that are the topic of this paper – can be conceived, where a pre-defined *structure* is assigned to an otherwise black-box model, orienting the space of its possible outputs towards a physically interpretable system.

3.1. Causal systems

Causal systems are systems whose output depends on past and current inputs. Let us consider discrete time models (which are compatible with artificial neural networks implementations), where all the quantities are known at given time samples identified by the subscript k. If $\boldsymbol{u}_k$ is the set of applied inputs to the system at time t_k and y_k is the output at the same time (assumed scalar), for a causal dynamical system a function f exists such that:

$$y_k = f(\boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \boldsymbol{u}_{k-2}, ...\boldsymbol{u}_{k-n}, t_k) \tag{1}$$

where n is the number of time steps at which the past time history of the inputs affects the outputs of less than the instrument noise, *i.e.* some multiple of the larger time constant of the system, if linear. If we apply this approach to the longitudinal dynamics of a vehicle, we can make the hypothesis that, at least in the timescale of application of the model here formulated, it is time-invariant, therefore we can discard the direct dependence of the function f on the variable t_k.

If nothing were known about f, eq. (1) would constitute a black-box model. On the other hand, if f were given analytically, eq. (1) would constitute a white-box model.

3.1.1. Gray-box models oriented to neural network implementations

Dealing with a vehicle longitudinal dynamics, we focus on its longitudinal acceleration, which is ruled by the projection of Newton's law along the longitudinal direction

(curvilinear coordinate s):

$$\ddot{s} = \frac{1}{M} \sum_{i=1}^{m} F_i \tag{2}$$

where M is the mass of the vehicle, F_i the longitudinal forces and m is the number of them.

Taking the vehicle longitudinal acceleration $\ddot{s}(t_k)$ as the output y_k of the plant, equation (1) can be rewritten according to the structure of equation (2):

$$\begin{aligned} y_k = & f_1(u_{1,k}, u_{1,k-1}, \ldots, u_{1,k-n}) + \\ & f_2(u_{2,k}, u_{2,k-1}, \ldots, u_{2,k-n}) + \ldots \\ & f_m(u_{m,k}, u_{m,k-1}, \ldots, u_{m,k-n}) \end{aligned} \tag{3}$$

and the subscript of f refers to the force being accounted for.

The usefulness of such structuring lies in the fact that i-th force, called f_i, usually is a function of *only a subset* $u_{i,j}$ of the inputs u_i (the time step being j). For example, the engine propulsive force does not depend on the brake command if the brake and engine systems are independent. So, with little insight into the physical system, one can tell which forces act and which is the input for each of the force modeling functions f_i. That is far less information than providing a precise analytical expression for f in eq. (1). If functions f_i are still not given analytically (apart from the specification of which inputs apply to each), the model is white at the level of eq. (3) and somehow gray at the level of the individual components f_i. The latter is gray, and not completely black, because each f_i carries in the argument list the information of which inputs affect and do not affect the force.

If eq. (3) is implemented with neural networks it retains the ability to learn (via the f_i) but within a framework that is generally more effective and robust than eq. (1). The structure given by (3) may be further developed, e.g. to model hybrid dynamical systems such as the case of different engaged gears. In this case we can imagine that the propulsive force is modeled by as many f_i as the number of gears, only one of which being active at any given time.

3.2. State space models

As an alternative to eq. (1), time-invariant dynamical system may be given the well known state space representation, such as:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{g}(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ y_k &= h(\boldsymbol{x}_k, \boldsymbol{u}_k) \end{aligned} \tag{4}$$

where $\boldsymbol{x}_k$ is the system state at time step k, which keeps the memory of the effects of past inputs.

State space representations may be implemented with *recurrent neural networks*, where $\boldsymbol{x}_k$ are the network recurrent states. Again, the functions $\boldsymbol{g}$ and h can be assigned a structure, reflecting some physical properties of the modeled system, or can be left generic, with no restriction to the generated set of solution manifolds.

3.3. Linear models/submodels

Very often, physical systems are operated within domains where they behave (approximately) as linear systems.

If the i-th dynamic subsystem in eq. (3) may be considered linear, the corresponding f_i is linear, and may be computed as the convolution of the function inputs with the impulse response of the subsystem itself, generically as:

$$y_i = \boldsymbol{\phi}_i * \boldsymbol{u}_i \tag{5}$$

where $\boldsymbol{\phi}_i$ is the vector of the impulse response coefficients of subsystem i, $\boldsymbol{u}_i$ is the vector of the past subsystem inputs and $*$ is the convolution operator.

In artificial neural networks, linear subsystems may be effectively implemented with linear layers and, remarkably, the *layer weights* may be interpreted as the *impulse response coefficients* of the subsystem, hence turning the sub-models f_i to be completely white boxes. Furthermore, at the time of network training, the linearity hypothesis is implicitly checked, because any non-linearity (where linearity was assumed) appears as training residuals.

Similarly, state space models of linear (sub)systems have the form of linear equations such as, e.g.:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k \\ y_k &= \mathbf{C}\boldsymbol{x}_k + \mathbf{D}\boldsymbol{u}_k \end{aligned} \tag{6}$$

where **A, B, C, D** are constant matrices. When linear state space models are implemented in neural networks with basic (linear) recurrent layers, the network weights can also be interpreted.

4. Vehicle longitudinal dynamics case of study

Four different types of neural networks are studied here[1]:

(i) a generalized convolutional network without structure, like eq. (1);
(ii) a generalized convolutional network with structure, like eq. (3);
(iii) a recurrent neural network without structure, like eqs. (4);
(iv) a recurrent neural network with structure (like eqs. (4); but where g and h are structured like f was in eq. (3)).

This way, all combinations of recursive versus non-recursive (generalized convolution) and black versus gray/white boxes approaches are compared[2]. The neural networks are evaluated for their capability of predicting the longitudinal acceleration. All the networks are trained with the same hyperparameters: batch size 1000, L2-norm weights regularization coefficient 0.0001, number of epochs 1000. The optimization procedure is performed with Adam's method.

[1]The networks were built, trained and verified in Wolfram Mathematica, which was also used for the signal pre-processing phase and the evaluation of the results.

[2]Comparisons with analytic models is carried out in [9]

4.1. Experimental data and signal pre-processing

In an experiment, a medium-sized car (Lancia Delta) was driven along an extra-urban course characterized by a combination of straights, curves, hills and intersections for a total test time of ~~approximately 1 hour~~50 minutes (55.7 km). The driving situations experienced in such a test are assumed to be reasonably rich and complete in terms of vehicle dynamics involved. The car was driven by an ordinary driver in ordinary conditions on a dry road. No exceptional events (hard brakes, slippages or interventions of the traction control and electronic stability program) occurred either in the training or the validation datasets. R2.2 R2.4

The training set was extracted from the whole records by taking about half an hour (36.0 km) in the first part of the trip. The validation set was extracted from the second part of the trip by taking about 15 minutes (19.7 km). These two sections were chosen to exclude parts of the records where the car was standing still very frequently. ~~The test and validation sets are related to completely different parts of the test track and with different traffic situations, thus avoiding as much as possible correlation between trained and verified maneuvers.~~The validation set is related to completely different parts of the test track and with different traffic situations, thus avoiding any correlation between trained and verified maneuvers. In Appendeix, figure A1 shows the whole data set divided in training set and validation set. R2.2 R2.2 R1.1

The longitudinal acceleration was measured by an accelerometer, while the relevant inputs for the longitudinal dynamics are identified in the forward velocity (used only for the modeling of the air drag), engine torque, brake plant pressure, altitude and gear, which are sensed by various devices. Additional signals that were also available were longitude and latitude; but these were not used to develop the forward dynamics models. All the signals (except the GPS) were sampled at 100 Hz; however since the significant dynamics is expected to be concentrated in a much more limited bandwidth, down-sampling at 20 Hz and proper digital filtering were performed to reduce noise effects.

Since the quality of the acceleration signal directly affects the training process of the neural networks and, in turn, their predictive performance, for the purpose of comparisons it was convenient to pre-process the data, maximizing the acceleration signal to noise ratio (which constitutes the training signal for the model output). Thanks to the availability of a set of redundant speed information from the odometers, a Kalman filter was hence used to improve both the velocity (that is not important here) and the acceleration signal. Details of a Kalman filter very similar to the present one and used on the same car are given in [10], which also describes the data and course slightly more in depth.

5. Generalized convolutional approaches

This section deals with networks of type (i) and (ii), which are non recursive networks that respectively implement the schemes eq. (1) and eq. (3). For linear systems the schemes reduce to convolutions of past inputs with the (learned) system impulse response. For nonlinear systems the schemes are nonlinear (learned) functions of the same past input history, which we can regard as a generalization of convolutions. The structured network (ii) is intentionally realized with a much smaller (less than 1/3rd) number of neurons, to appreciate the advantages and role of the network structure in modeling and prediction performances.

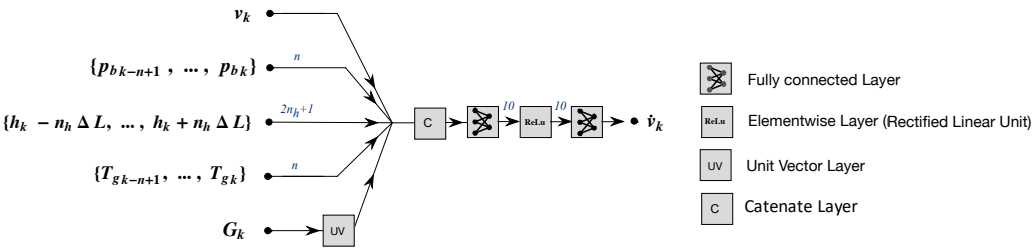

Figure 1. Unstructured convolutional network (i).

5.1. (i) Unstructured convolution-like neural network

The unstructured neural network (i) is shown in fig. 1. It is realized with a shallow architecture: one hidden layer (layer 3) of 10 fully connected neurons, followed by a ramp activation function (ReLu), feeds the second neural layer, composed of just 1 neuron (layer 5). The number of learn-able parameters (weights and biases) for this network is 821 (810 for layer 3 plus 11 for layer 5).

As for the network input, each neuron of the first layer receives a short time history of the input signals: n=25 past brake pressures (up to the current time), n=25 past engine torque values, 21 height values[3], the current velocity and the current gear. The latter is encoded by a unit vector layer (layer 1) which returns a vector of length 8 (number of gears counting the neutral gear) with all elements set at zero except the element corresponding to the active gear which is set at 1. This vector is concatenated with the other inputs on layer 2 (hence layer 3 receives 8 signaling neurons that specify which is the currently active gear). Note that only the instantaneous velocity is given in the inputs: the network can thus use this information to model air drag, but has no cue to derive acceleration form differentiation of the velocity.

Given this structure, the length of the time history of each input had to be defined with care. Extending to excessively old samples increases the number of learn-able parameters, with the risk of over-fitting. Conversely, too short an history makes the network unable to grasp the system dynamics that occurs in larger timescales. Note that the longest windows, extending back to 25 samples, means 1.25 seconds, which is a time interval sufficient for most of the longitudinal dynamics time scales.

Figure 3 gives an example of the network performance. In the excerpted record both acceleration and deceleration maneuvers are present with related gear shifts. The network captures the main longitudinal dynamics phenomena (including drive-line vibrations), as shown by the typical oscillations of the acceleration that follow gear shifts, especially at low gears.

5.2. (ii) Structured convolution-like neural network

In the structured network (ii) the neurons are organized as shown in fig. 2 in order to process the input signals according to the physics inspired logic eq. (3). Here, the various converging network branches model different f_i in eq. (3). On top the instantaneous velocity is squared and enters a layer which learns the air drag coefficient and the rolling resistance (gain and bias of the single neuron constituting the layer).

[3]Heights are extracted from a digital map sampled at $2n_h+1$ regular distances from the vehicle position (n_h=10 with a total of 21 points); the network is left with the estimation of the slope from the noisy heights.

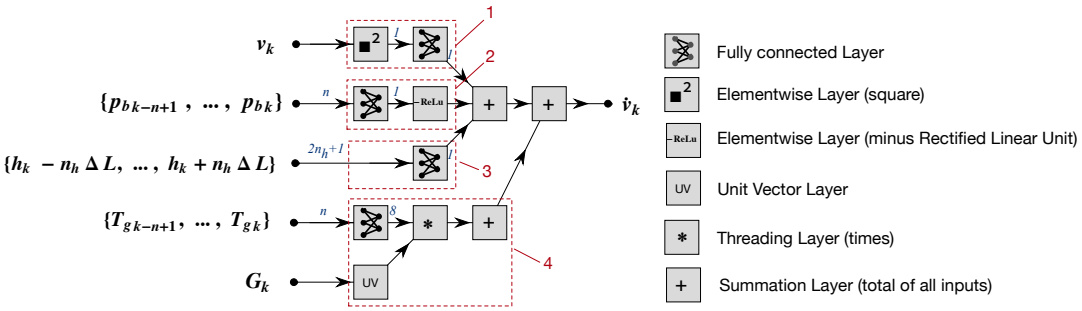

Figure 2. Structured convolutional network (ii). Dotted boxes correspond to (1) air drag, (2) braking force, (3) slope effect and (4) engine torque respectively.

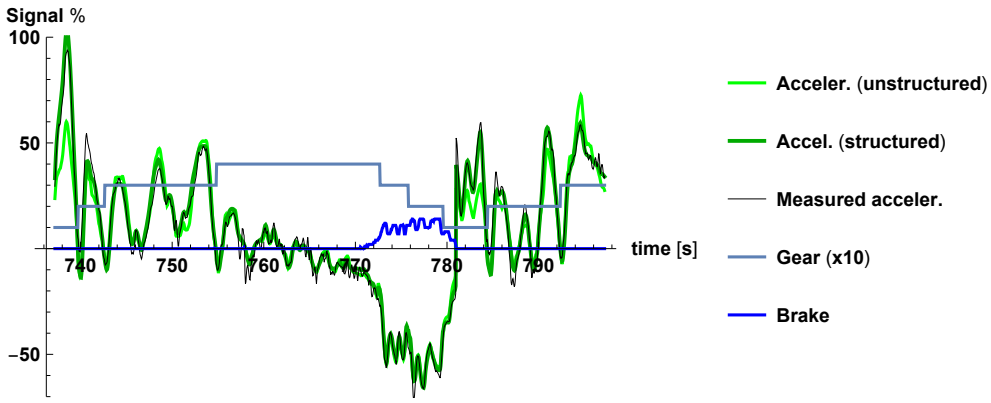

Figure 3. Example of measured signals and convolutional networks prediction. Accelerations are rescaled with respect to the maximum measured acceleration ($3.1ms^{-2}$).

On the second row, the brake pressure history – same 25 values than in network (ii) – enters a neural layer which learns the acceleration effect of the brake, which was supposed linear (and the hypotheses was then verified in the training). The following ReLu (Rectified activation unit) was included to enforce the fact that brake forces can only cause decelerations (which helps the training convergence). The third row estimates the acceleration caused by road slope. Finally the bottom row estimates the tractive force and acceleration. Here the 25 past engine torque values enter a neural layer with 8 output neurons. Each neuron learns the propulsive acceleration that would be caused by one gear. Only the neuron corresponding to the active gear is then passed downstream. The total number of learn-able parameters in this network is 248 of which 200 are in the engine layer which learns the effects of the drive-line for the 8 different gears in parallel.

5.3. Comparison between unstructured and structured convolutional networks

In fig. 3 the predicted acceleration by the unstructured and structured networks are compared with the measured acceleration. It can be seen that the structured network, despite the reduced number of parameters, provides a better fit.

The networks performances are evaluated by examining the power spectral density

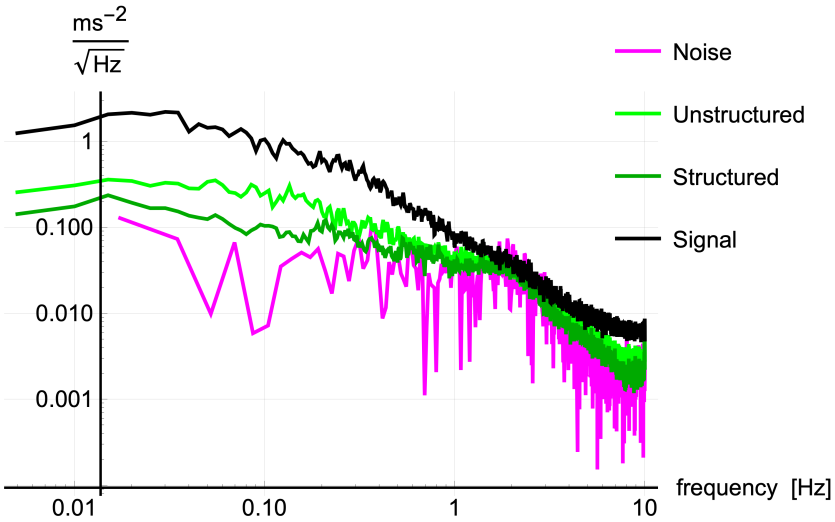

Figure 4. Power spectral density of acceleration measurement noise, convolutional neural networks fit residuals and measured acceleration signals (training set).

[11] of: a) the acceleration signal, b) the fit residuals of network (ii), c) the fit residuals of network (i) and d) the measurement noise (fig. 4). The latter is estimated by extracting the parts of the acceleration signal (after Kalman filtering) where the vehicle is running with constant velocity. This stretch of data, of only about 50 s in total, is de-trended, windowed (Hamming type) and its power spectral density is calculated with the periodogram method [12] as a reference level (the pink line in the chart). The limited time span of the extracted acceleration noise does not allow the calculation of the noise spectral density as the mean of many estimates, resulting in a significantly greater scatter of the plotted line (compared to the other curves). Conversely, spectral densities of the signal and residuals are the average of about 10 estimates.

We make the assumption that the system is of output-error type, which sees the noise superimposed to the system output without being subjected to the system dynamics. This allows for a direct comparison between fit residuals and measurement noise, which constitutes its theoretical bottom limit. Residuals spectral densities lower than the measurement noise indicate an over-fitting behavior of the network. This is also revealed during training by the validation loss.

Figure 4 shows that the acceleration signal (black line) is significantly stronger than the noise (pink line) up to 2-3 Hz, where an evident cutoff is present and the signal-to-noise ratio drops to zero dB. ~~In the 2-3 Hz bandwidth, the noise is fairly constant.~~In this band the longitudinal vibration modes due to the driveline/chassis/suspensions, which are not causally linked to the input, are superimposed to the acceleration caused by the longitudinal control. The vehicle controlled dynamics is mainly concentrated in the low frequency band, with an initial plateau (up to about 0.1 Hz) and then following $1/f$ decrease. R2.5

The unstructured network (light green line) in the relevant bandwidth produces fit residuals significantly larger than the noise, revealing imperfect capability to describe the vehicle dynamics especially at the low-frequency end. Conversely, the residuals produced by the structured network (dark green line) lie closer to the noise level, showing that this network model performs fairly similarly to the measured dynamics. The structured network performs better than the unstructured one also at the high-

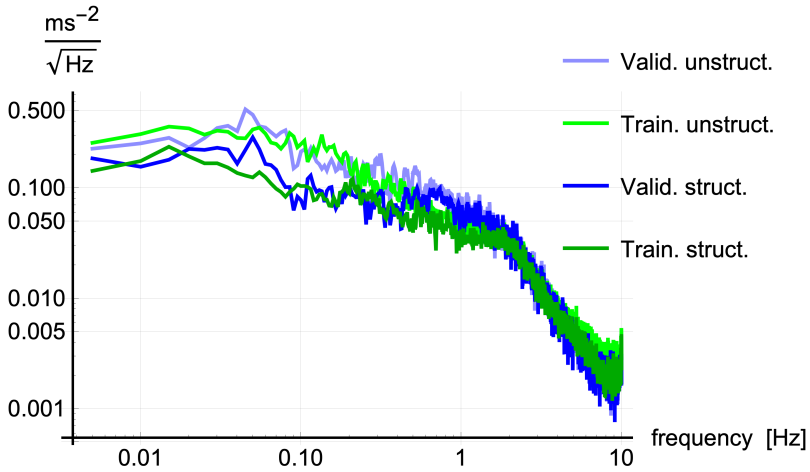

Figure 5. Power spectral density of the fit residuals of the training and validation sets for unstructured and structured convolutional networks.

frequency end (up to 10 Hz), where it is basically consistent with the noise. The larger bandwidth of the structured network can be appreciated also in the time domain (fig. 3), where it is able to produce sharper acceleration peaks/valleys and follow quick signal changes with no relevant error (e.g. first peak at about 2 s).

In fig. 5 the power spectral density of the fit residuals of the training and validation sets for both unstructured and structured networks are compared. Around 1 Hz, in both unstructured and structured cases the network produces slightly larger residuals in the validation set with respect to the training set. However, the difference between training and validation is negligible elsewhere and if compared with the gap to the signal. The plot confirms that the networks are correctly optimized and the dynamic characteristics of the track are homogeneous.

6. Recurrent Neural Networks

As an alternative option, we consider recursive networks. Two networks, of type (iii) and (iv) are used to implement the scheme eq. (4). According to this approach, the interpretation of the network parameters (e.g., related to matrices A, B, C and D in the linear case) is in principle possible. However, in the absence of a structure the network parameters are allowed to converge to a configuration in which no physical meaning can be given (for instance, mixing the effect of the different inputs at an earlier stage than the final summation in Eq. 3). This issue is exacerbated if an oversized state (required to model nonlinearities) is chosen, which increases the possibility of multiple locally optimal solutions of the network parameters. Both the structured and unstructured recursive networks (iii) and (iv) are realized with a similar number of neurons as their convolutional equivalent, for direct comparison.

6.1. (iii) Unstructured recurrent neural network

The unstructured recurrent neural network is shown in fig. 6. The input is made of the instantaneous velocity v_k, brake pressure p_{bk}, altitude h_k, engine torque T_{gk} and the gear signalling vector (output of the UV layer). They are concatenated at layer C,

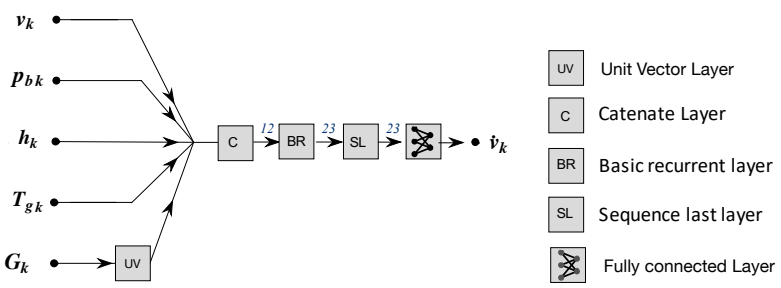

Figure 6. Unstructured recurrent network (iii).

forming the instantaneous input vector $\boldsymbol{u}_k$, of dimension n=12, which enters a basic recursive layer BR. Basic Recurrent layers implement the following state transition, where $\boldsymbol{s}_k$ are the neural network recurrent states at time step k.

$$\boldsymbol{s}_{k+1} = \mathrm{Tanh}(\mathbf{A}\,\boldsymbol{s}_k + \mathbf{B}\,\boldsymbol{u}_k + \boldsymbol{b}) \tag{7}$$

With adequately sized state $\boldsymbol{s}$, basic recurrent layers can model non-linear dynamics. However the states $\boldsymbol{s}$ are not directly interpretable. The predicted acceleration is assumed to be a linear combination of these states, which are combined at the last fully connected layer.

The learnable parameters are located in the matrices $\mathbf{A}$, $\mathbf{B}$ and $\boldsymbol{b}$ of the BR layer and in the weights and biases of the linear layer. The state dimension (m=23) is chosen such that the number of learnable parameters (852) is very close to that of the unstructured convolutive network.

The network is trained as stateless: a sequence of 25 input vectors – the same size of the history of networks (i) and (ii) –, $\boldsymbol{u}_{k-24}, ..., \boldsymbol{u}_k$ is taken, beginning with the null state $\boldsymbol{s}_{k-24} = 0$. The output of the recurrent layer is a sequence of 25 states $\boldsymbol{s}_{k-24}, ..., \boldsymbol{s}_k$ of which the first 24 are not considered to ignore the transient phase (it was assumed and verified that the transient response lasts less than 25 steps). The last state $\boldsymbol{s}_k$ is considered (extracted via the Sequence Last (SL) layer) and transformed into the estimated acceleration $\dot{v}_k$ that is trained against the actual acceleration.

6.2. (iv) Structured recurrent neural network

Figure 7 shows the structured recurrent network (iv). In this case each individual signal feeds a dedicated BR recursive layer, where overall the structure follows the (ii) template (fig. 2). The individual BR layers are assigned a reduced state dimension (m=6) in order to produce a total number of train-able parameters (262) similar to that of the convolutive version (ii). The network is trained the same way as network (iii).

6.3. Comparison between unstructured and structured recurrent networks

The analysis of the fit residuals of the unstructured and structured recurrent networks in the frequency domain shown in fig. 8 highlights that the performance of the unstructured network looks slightly better in the bandwidth up to 2-3 Hz and both

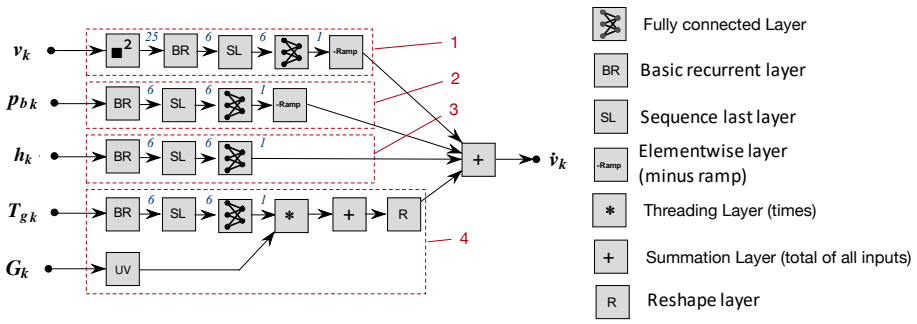

Figure 7. Structured recurrent network (iv). Dotted boxes correspond to (1) air drag, (2) braking force, (3) slope effect and (4) engine torque respectively.

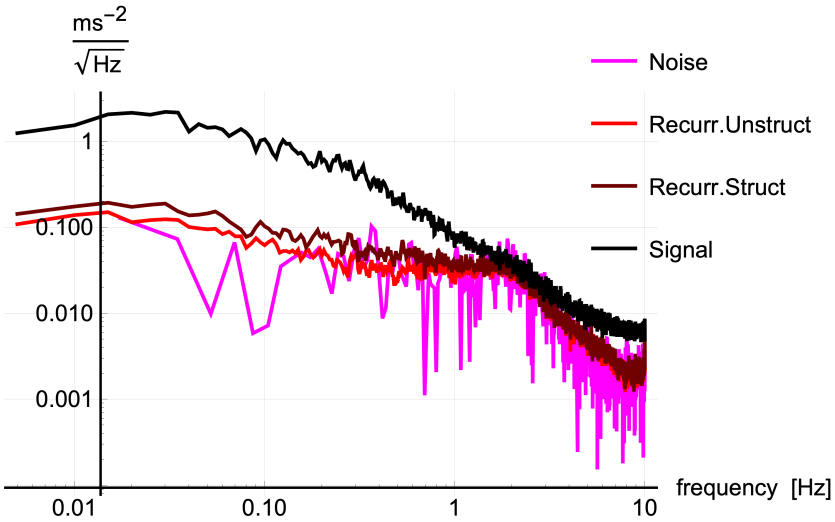

Figure 8. Power spectral density of acceleration measurement noise, recurrent neural networks fit residuals and measured acceleration signals (training set).

consistent with the measurement noise. We notice however that in some frequency bins the unstructured network may produce residuals with a smaller power spectral density than the noise, *i.e.* overfitting the acceleration signal.

Figure 9 shows the power spectral density of the fit residuals for recurrent networks, both with training and validation sets. The performance of the networks is more affected by the dataset, with a larger increase of residuals' spectral content from training to validation if compared with the convolution case of fig.5, distributed on a larger bandwidth. The unstructured recurrent network shows the worst performance in terms of robustness to the choice of the dataset.

In fig. 10 the performances of the recurrent networks are compared in the same sample interval that was used in fig. 3. At a first glance it seems that the recurrent networks outperform the convolutive ones. However, a comparison in the validation set, as will be shown in fig. 9 and fig. 12 reveals that the apparent better fit here has to be interpreted as over-fitting (see discussion in section 7 and Table 1). R.1.2

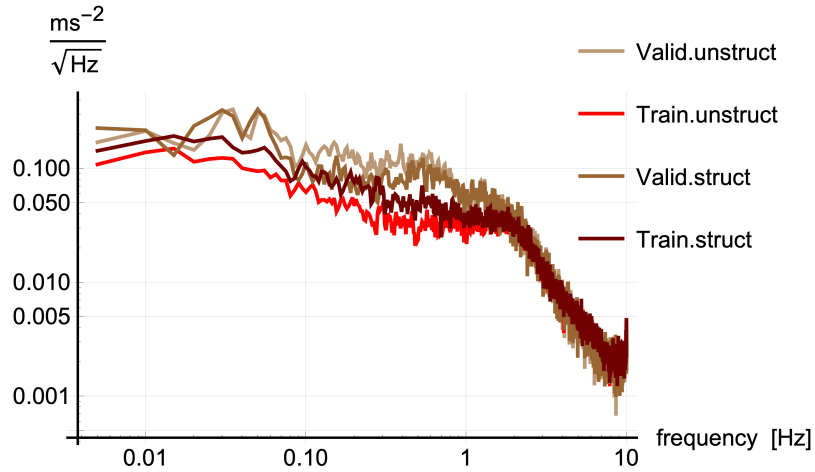

Figure 9. Power spectral density of the fit residuals of the training and validation sets for unstructured and structured recurrent networks.

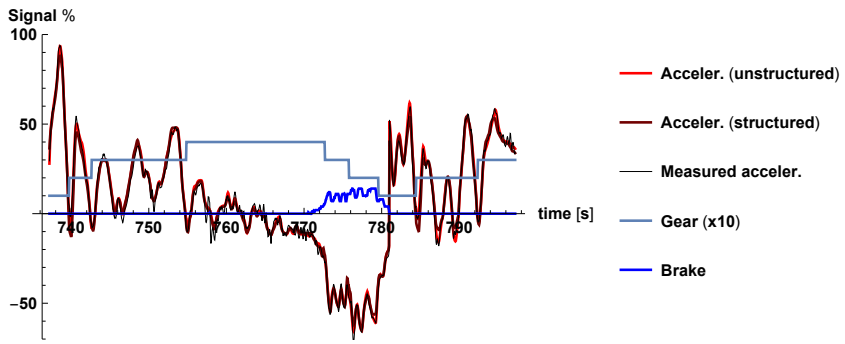

Figure 10. Example of measured signals and recurrent neural networks prediction. Accelerations are rescaled with respect to the maximum measured acceleration (3.1 ms^{-2}).

7. Comparison and performances

We first compare in Table 1 the performances of the four networks by analyzing a simple scalar quantity expressing the overall power of the fit residuals, that is the root mean square (RMS). The reference value is the acceleration noise RMS, 0.067 ms^{-2}.

There are clear indications that the unstructured recurrent network is overfitting the signal, because the RMS of the residuals of the training set is smaller than that of the noise. This is confirmed by the quite large RMS increase of the residuals in the validation set, which highlights a significant dependence of the network performance on the data set.

From Table 1, and focusing on the validation sets, it is possible to conclude that

	Convolutive		**Recurrent**		
	Unstructured	Structured	Unstructured	Structured	Noise
Training	0.141	0.084	0.063	0.080	0.067
Validation	0.158	0.105	0.129	0.115	

Table 1. Root mean square of the fit residuals (ms^{-2}).

the structured architectures, both convolutive and recurrent, albeit with far less parameters, yet provide a more robust description of the longitudinal dynamics without loss of modelling capability and in the fairly wide and representative conditions of extra-urban driving conditions of a medium-sized car that we used.

Concerning the spectral distribution, fig. 11 compares the frequency content of the fit residuals focusing on the structured networks only. Both produce residuals fairly consistent with the measurement noise up to the maximum frequency (10 Hz).

Of the many other scalar metrics that may be used to evaluate the quality of model predictions (e.g. model fit, variance accounted for $v.a.f.$, Akaike's Final Prediction Error FPE [13]), we calculate here the variance accounted for metric, which evaluates the variance of the residuals with respect to the variance of the measured signal (consistently with [14]):

$$v.a.f. = 1 - \frac{var(a_m - a_{net})}{var(a_m)} \tag{8}$$

Where a_m is the measured acceleration vector and a_{net} is the network predicted acceleration vector.

Table 2 lists the $v.a.f.$ of the four networks for both training and validation sets. The values confirm the overfitting situation for the unstructured recurrent network, which produces a clear performance drop from the training to the validation set.

Considering the overall performance (both in terms of absolute values and their robustness), the structured configurations provide the best description of the vehicle dynamics, with a nearly equivalent performance. However, the convolutive network has the following advantages.

1) It is more robust with respect to the data set. As shown in Table 1, the absolute performance of the structured convolutive network in the validation set is better, with a RMS increase from the training set of 25% against 44%. Also, Table 2 shows that the drop of performance of the structured convolutive network from the training to the validation sets is 0.008 whereas the structured recurrent scores 0.015 (about a factor 2 better).

2) Easier implementation, larger stability with respect to back-propagation, faster convergence in training.

3) It can be given a direct interpretation in terms of impulse response of the system.

	Convolutive		**Recurrent**	
	Unstructured	Structured	Unstructured	Structured
Training	0.938	0.978	0.987	0.980
Validation	0.935	0.970	0.956	0.965

Table 2. Variance accounted for metric for the tested networks.

Finally, fig. 12 compares the actual acceleration to the predictions of the 4 networks in the time domain for an excerpted window of about 15 s, during a typical situation experienced in the validation set involving both high-frequency and low-frequency dynamics. Findings are confirmed, with the convolutive structured network performing best.

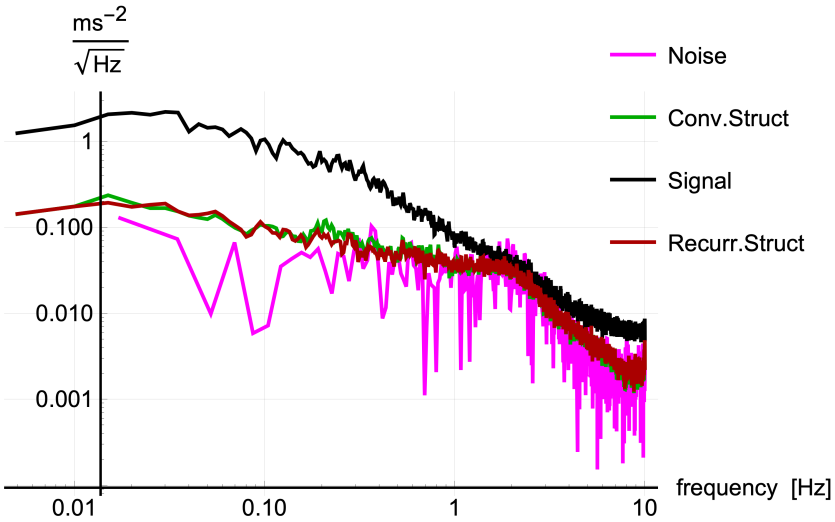

Figure 11. Power spectral density of acceleration measurement noise, convolutional and recurrent structured neural networks fit residuals and measured acceleration signals (training set).

8. Conclusion

In this paper neural networks of different architectures are implemented as tools for the modelisation of the longitudinal dynamics of a medium-sized vehicle. In particular, models of different levels of *structure* and *interpretability* may be conceived, spanning from white-box models, where physical principles are directly implemented and model parameters have a physical interpretation, to black-box models, which are opaque to any insight. Pre-wired structures of both convolutive and recurrent networks guarantee that the trained network maintain some degree of interpretability, that is allow the user to identify the effect of the different inputs (and related forces) to the vehicle acceleration, still preserving the flexibility to adapt to unexpected dynamics. The comparison of the modeling performances of convolutive and recurrent networks shows that, with a similar number of trainable parameters, the convolutive architecture is more accurate and robust.

As models for vehicle dynamics neural networks necessarily model one specific vehicle, unlike analytically models that may be parametric. Their natural application is fine-tuned learning of the dynamics of one vehicle for a) vehicle control, b) sensor anticipation (to filter sensor data based on a dynamic model) and c) to learn vehicle models that may be used for vehicle specific offline simulations for the synthesis of tactical-level motor strategies (a process that is also known as "embodied" simulations when it refers to how human manipulate learned models of their body and manipulate objects). All these uses are actually implemented in the Dreams4Cars project [5]. Furthermore, just like human beings, the learning of several models in parallel, for example reflecting the variable vehicle dynamics in varying environmental and operating conditions, may be implemented to construct a library of vehicle models among which to select for the actual control in one specific day (just like human beings do).

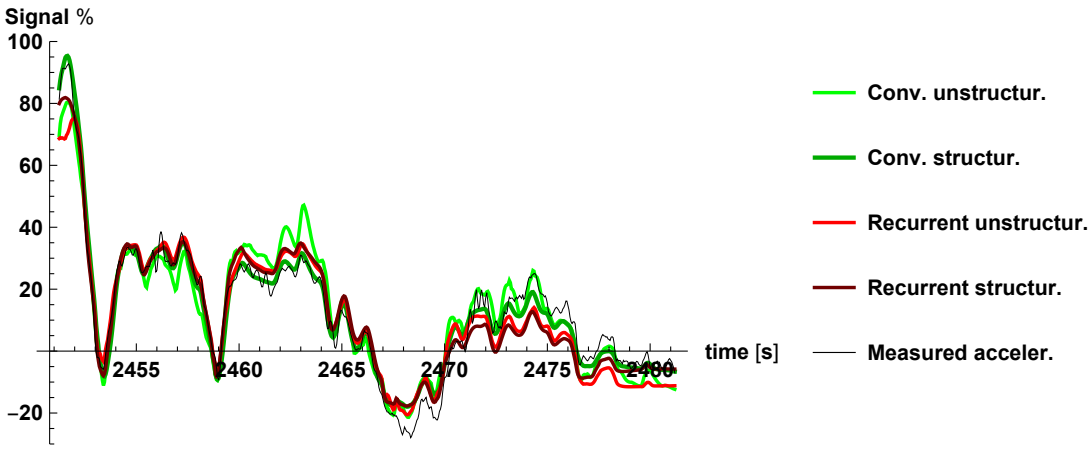

Figure 12. Example of measured and predicted acceleration (rescaled with respect to 3.1ms^{-2}) by convolutional and recurrent networks in the validation set.

9. Acknowledgements

This work is supported by the European Commission under Grant 731593 (Dreams4Cars) and partially under Grant FP7 610428 (AdaptIVe).

Appendix A. Training and validation datasets

The signals collected in the experimental campaign are reported in fig. A1, evidencing the training and validation subsets. Two sample intervals are highlighted in light green: the first refers to fig. 3 (comparison between the convolutional networks) and fig. 10 (comparison between the recurrent networks) while the second refers to fig. 12 (comparison among all the networks). In tables A1 and A2 the statistical parameters of the signals are summarized for the training and validation sets respectively. R1.1 R2.3

We can note that the validation set involves slightly wider ranges of acceleration brake and slope.

	Acceleration (ms^{-2})	Velocity (m/s)	Engine Torque (Nm)	Brake (bar)	Slope (rad)
Quantile 0.0005	-2.60	0.00	-10.98	0.00	-0.027
Quantile 0.005	-1.96	0.00	-10.58	0.00	-0.025
Mean	0.03	18.02	13.03	0.68	0.000
Quantile 0.995	1.94	31.39	61.00	15.00	0.030
Quantile 0.9995	2.69	32.05	71.81	22.00	0.033

Table A1. Statistics of the training set.

	Acceleration (ms^{-2})	Velocity (m/s)	Engine Torque (Nm)	Brake (bar)	Slope (rad)
Quantile 0.0005	-3.45	1.38	-11.29	0.00	-0.036
Quantile 0.005	-2.47	5.75	-10.82	0.00	-0.032
Mean	0.02	19.73	13.16	0.67	0.001
Quantile 0.995	1.97	30.76	62.01	19.20	0.044
Quantile 0.9995	3.15	30.95	73.70	29.60	0.052

Table A2. Statistics of the validation set.

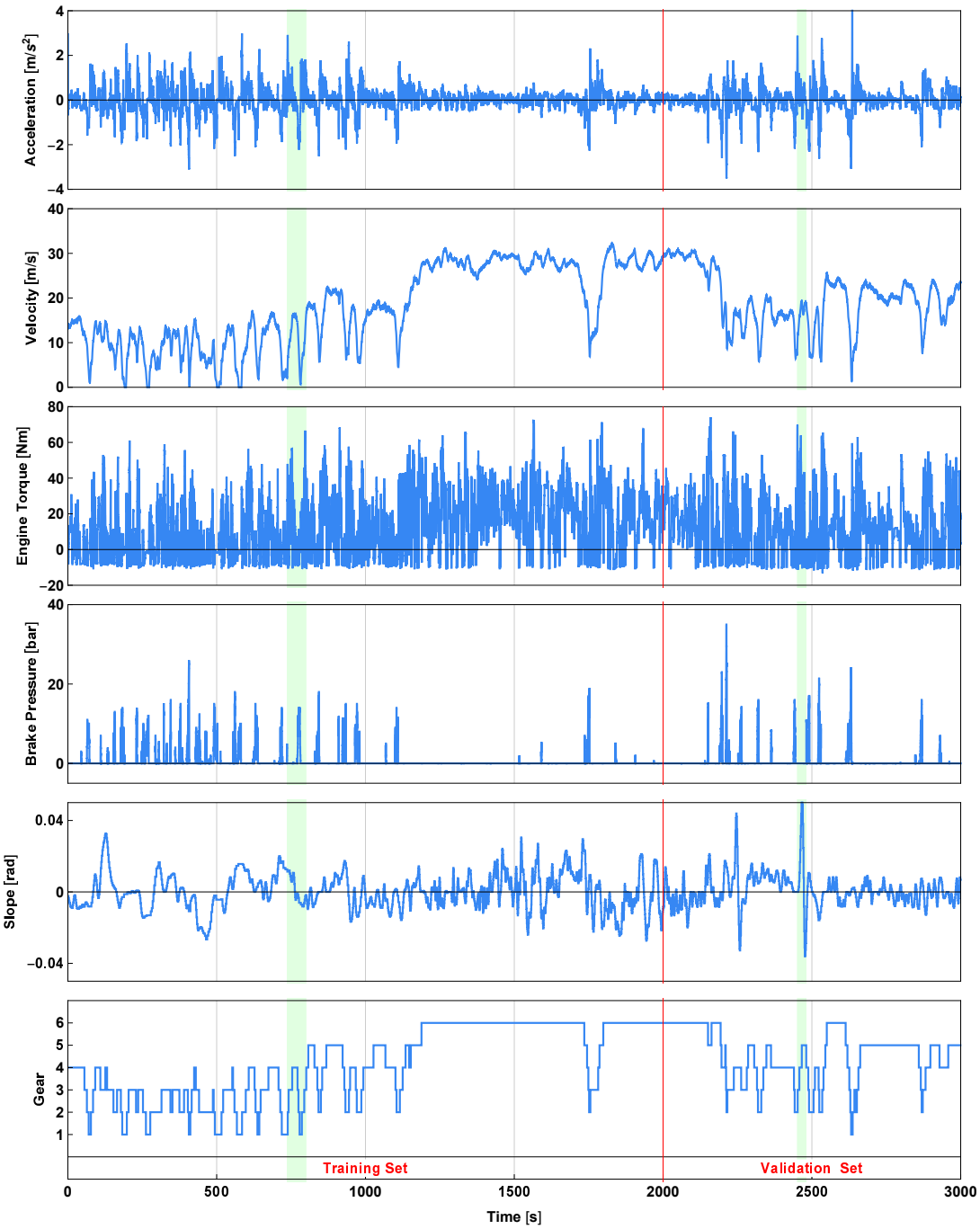

Figure A1. Experimental data. The first part of the data is used as training set and the second part is used as validation set. The green bands refer to the sample intervals used in fig. 3, fig. 10 and fig.12.

References

[1] Abe M. Vehicle handling dynamics: theory and application. Butterworth-Heinemann; 2015.

[2] Liu W, Wang Z, Liu X, et al. A survey of deep neural network architectures and their applications. Neurocomputing. 2017;234:11–26.

[3] Jones W, Alasoo K, Fishman D, et al. Computational biology: deep learning. Emerging Topics in Life Sciences. 2017;1:136–161.

[4] Bojarski M, Yeres P, Choromanska A, et al. Explaining how a deep neural network trained with end-to-end learning steers a car. CoRR. 2017;abs/1704.07911. Available from: http://arxiv.org/abs/1704.07911.

[5] Dream-like simulation abilities for automated cars ; Accessed: 2019-01-21. Available from: https://cordis.europa.eu/project/rcn/206390/factsheet/en.

[6] Plebe A, Papini Rosati GP, Donà R, et al. Dreaming mechanism for training bio-inspired driving agents. In: International Conference on Intelligent Human Systems Integration; Springer; 2019. p. 429–434.

[7] Funahashi Ki, Nakamura Y. Approximation of dynamical systems by continuous time recurrent neural networks. Neural networks. 1993;6(6):801–806.

[8] Chow TW, Li XD. Modeling of continuous time dynamical systems with input by recurrent neural networks. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications. 2000;47(4):575–578.

[9] James S, Anderson SR, Da Lio M. Longitudinal vehicle dynamics: A comparison of linear state-space, nonlinear physical and neural networks models. IEEE Transactions on Intelligent Vehicles. submitted;.

[10] Bisoffi A, Biral F, Da Lio M, et al. Longitudinal jerk estimation of driver intentions for advanced driver assistance systems. IEEE/ASME Transactions on Mechatronics. 2017; 22(4):1531–1541.

[11] Papoulis A. Probability, random variables, and stochastic processes. McGraw-Hill; 2015.

[12] Welch P. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Transactions on audio and electroacoustics. 1967;15(2):70–73.

[13] Akaike H. Statistical predictor identification. Annals of the institute of Statistical Mathematics. 1970;22(1):203–217.

[14] James S, Anderson SR. Linear system identification of longitudinal vehicle dynamics versus nonlinear physical modelling. In: 2018 UKACC 12th International Conference on Control (CONTROL); IEEE; 2018. p. 146–151.