POLITECNICO DI TORINO
Repository ISTITUZIONALE

An imitation learning framework for neural network-based path tracking control of multi-actuated electric vehicles

Original
An imitation learning framework for neural network-based path tracking control of multi-actuated electric vehicles / Formento, C., Ingraffia, S., Dimauro, L., Velardocchia, M., Sorniotti, A.. - In: VEHICLE SYSTEM DYNAMICS. - ISSN 0042-3114. - ELETTRONICO. - (2026), pp. 1-30. [10.1080/00423114.2026.2739308]

Availability:
This version is available at: 11583/3017192 since: 2026-10-29T10:49:22Z

Publisher:
Taylor & Francis

Published
DOI:10.1080/00423114.2026.2739308

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

04 November 2026



Taylor & Francis
Taylor & Francis Group

Vehicle System Dynamics
International Journal of Vehicle Mechanics and Mobility

ISSN: 0042-3114 (Print) 1744-5159 (Online) Journal homepage: www.tandfonline.com/journals/nvsd20

An imitation learning framework for neural network-based path tracking control of multi-actuated electric vehicles

Cecilia Formento, Salvatore Ingraffia , Luca Dimauro, Mauro Velardocchia & Aldo Sorniotti

To cite this article: Cecilia Formento, Salvatore Ingraffia , Luca Dimauro, Mauro Velardocchia & Aldo Sorniotti (21 Oct 2026): An imitation learning framework for neural network-based path tracking control of multi-actuated electric vehicles, Vehicle System Dynamics, DOI: 10.1080/00423114.2026.2739308

To link to this article: https://doi.org/10.1080/00423114.2026.2739308

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 21 Oct 2026.

Submit your article to this journal

Article views: 285

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=nvsd20

VEHICLE SYSTEM DYNAMICS
https://doi.org/10.1080/00423114.2026.2739308

OPEN ACCESS Check for updates

An imitation learning framework for neural network-based path tracking control of multi-actuated electric vehicles

Cecilia Formento, Salvatore Ingraffia, Luca Dimauro, Mauro Velardocchia and Aldo Sorniotti

Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Torino, Italy

ABSTRACT

This paper presents a path tracking (PT) control framework for electric vehicles equipped with multiple chassis actuators. The methodology combines nonlinear model predictive controllers (NMPCs) acting as experts, with the derivation and application of imitation learning (IL) neural policies. Three actuation configurations are considered: (i) front-wheel-steering (2WS); (ii) four-wheel steering (4WS); and (iii) front-wheel steering and torque-vectoring (TV). The NMPC formulations use an accurate vehicle dynamics model to achieve precise trajectory tracking under aggressive driving conditions. Their high computational load, however, limits deployment in real-time embedded systems. A dedicated IL toolchain trains neural networks (NNs) to reproduce NMPC behaviour with drastically lower turnaround time, also in the presence of a reduced set of input variables with significant latencies. The resulting controllers are evaluated through simulations and experiments on a prototype vehicle platform. In simulation, the NNs match NMPC tracking performance. In vehicle experiments in limit handling conditions, the maximum lateral error is reduced by an average of 58% across the three configurations (2WS, 4WS and TV) relative to the corresponding benchmarking controllers (BCs). Robustness is assessed through: (a) a Monte Carlo analysis; (b) manoeuvres with swiftly variable tyre-road friction conditions; and (c) trajectories substantially different from those in the training set, all confirming the generalisation capability of the NN-based approach.

ARTICLE HISTORY
Received 6 April 2026
Revised 15 July 2026
Accepted 6 September 2026

KEYWORDS
Automated vehicles; multiple chassis actuators; path tracking control; nonlinear model predictive control; imitation learning; neural networks

1. Introduction

In recent years, automated vehicles have attracted increasing attention due to their potential to improve road safety and efficiency in transportation systems. Since human errors account for most traffic accidents [1–2], reducing driver-dependent variability and mistakes represents a key motivation for the development and deployment of automated driving technologies [3].

An important component of a typical automated driving system is represented by the PT algorithm, which is usually responsible for generating the front steering commands as well as the traction and braking torque requests, to follow the reference trajectory and speed

CONTACT Aldo Sorniotti aldo.sorniotti@polito.it

© 2026 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

profile set by the path planning layer [4]. High-performance path tracking is crucial for safe and robust manoeuvring, especially at and beyond limit handling conditions, which are typical of emergency scenarios. A very broad range of PT algorithms is available in the literature. For example, kinematic controllers, e.g. based on the pure pursuit or Stanley methods, are easy to implement but tend to degrade during high lateral acceleration operation. In [5] Hoffmann et al. augment the Stanley approach with a feedback term to compensate for the yaw rate error, and a term based on the measured steering angles at the current and previous time steps, to compensate for the actuation delay and overshoot. For limit handling operation on a race track, reference [6] proposes a control structure combining a feedforward (FF) contribution accounting for the vehicle's understeer gradient, and a proportional feedback (FB) contribution considering the lateral error with a look-ahead distance. More advanced linear approaches, such as linear quadratic regulators [7] and H_∞ controllers [8], offer structured model-based design frameworks, with formal consideration of robustness in the H_∞ case. However, these methodologies still have significant performance limitations in the nonlinear cornering response region, which motivates the use of nonlinear model predictive controllers (NMPCs), see the review in [9]. NMPCs can embed nonlinear prediction models and constraints, and thus achieve excellent performance also during large sideslip operation. This is the case in [10], where an NMPC implementation is used to perform controlled drifting to facilitate the completion of evasive manoeuvres in emergency road scenarios, and in [11], whose NMPC targets vehicle stability at the cornering limit. However, NMPCs are computationally heavy and difficult to implement in real time. Specialised solvers as well as high-performance control hardware enable real-time operation under simplifying assumptions, e.g. in terms of prediction model formulation; see [12,13].

Artificial intelligence (AI) methods, typically resulting in neural network (NN) controllers, offer alternative means of achieving highly performing control algorithms, often with low online computational cost. In the field of automated driving applications, the study in [14] proposes a path planner, i.e. an algorithm to generate the reference trajectory, based on an agent that clones the behaviour set by an advanced controller – a model predictive controller in the specific case – through a so-called imitation learning (IL) architecture. Within IL frameworks, the expert typically provides optimal or near-optimal control actions that cannot be directly used for control purposes, either because the corresponding algorithm is not real-time implementable (e.g. in case of an advanced NMPC), or because they do not come from an algorithm, but from a human driver. Reference [15] adopts an NN as prediction model within a model predictive PT controller, thus obtaining a neural network model predictive controller (NNMPC), where the NN can better approximate the vehicle dynamics than a typically simplified control-oriented and physics-based formulation, and possibly requires a reduced set of input variables. As an alternative, reinforcement learning (RL) algorithms generate agents that learn policies based on rewards and loss functions derived from the interaction with the environment. RL agents are implemented either as PT algorithms on their own, or as supportive or corrective layers for other controllers. For example, reference [16] shows that pure RL agents can achieve high accuracy, but raise safety and interpretability concerns, making RL-augmented approaches more practical. RL augmentations – covered by the same study – can be implemented for varying the control gains or cost function weights during controller operation, or to generate corrective control commands. In [17] Caponio et al. provide an

example of deep RL agent for PT, including an online NN weight adaptation architecture, which is experimentally assessed on small-scale QCar vehicles.

In terms of IL-based PT control, reference [18] proposes an end-to-end IL algorithm for highway lane keeping, where the NN learning routine uses a virtual driver model for the direct mapping from the front camera images to the steering command, in a co-simulated highway circuit with large-radius corners. A conceptually similar end-to-end IL formulation is presented in [19], where the controller operates on a limited amount of information – camera images and high-level route commands – along urban driving scenarios. Neither study involves demanding manoeuvres. Instead, in [20] Hawke et al. imitate a human driver in more demanding scenarios, and then the resulting NN controller is subject to continuous online training based on loss minimisation. Since the effectiveness strongly depends on the quality of the initial IL controller, an offline IL phase capable of producing a good baseline policy is still required. In [21], IL is adopted only as a first step providing a PT NN that is subsequently refined through RL. Other studies apply IL to more specific and demanding tasks. In [22], IL is used to design a controller for an evasive manoeuvre managed by an NMPC master. The resulting controller is specialised for collision avoidance – rather than general PT – and relies only on steering and brake actuations. The NN receives the full set of vehicle states, implying that a real-world implementation would require to accurately estimate or measure the sideslip angle. The need to reduce sensor dependency is explicitly addressed in [23], developing an IL routine to drive a scaled vehicle at high speed with a very limited set of inputs, by imitating a model predictive controller. However, the method is only for lane keeping in off-road conditions rather than accurate PT. The distance travelled without crashing is employed as a performance metric, which would be difficult to implement for safety reasons in full-scale vehicles. The best performance is obtained via online NN training. In [24], an IL scheme is proposed in which an NN learns a PT policy from an NMPC based on a kinematic model with four states, which are also used as NN inputs.

In general, independently from the adopted algorithms, PT implementations are very sensitive to sensor sampling time and accuracy. In [25] Xu et al. discuss the implications of communication delays and steering lag, which affect tracking performance, steering smoothness, and vehicle stability. The proposed solutions are based on a forward state predictor and a delay augmentation setup.

As a summary, the literature lacks: (i) centralised IL PT implementations for multi-actuated vehicles, e.g. including multiple electric powertrains or rear-wheel-steering [26]. Recent PT studies, e.g. [27–29], highlight the importance of multi-actuation, but do not explore IL; (ii) IL PT algorithms for vehicle operation in highly demanding conditions, since most of the IL PT literature presents visual end-to-end techniques without stringent performance requirements – see the review in [30] – in presence of which model predictive controllers are typically used; (iii) PT controllers that do not require the full set of vehicle states for effective limit handling operation; and (iv) PT algorithms that provide robust and accurate performance also in presence of significant sensor delays. This study addresses the identified gaps with the following contributions:

- A complete IL toolchain to train robust and high-performing NN-based PT controllers that enable safe experimental deployment across a broad range of scenarios.

The involved NNs imitate the control actions generated in simulation by non-real-time implementable NMPCs capable of effectively managing vehicle nonlinearities and aggressive manoeuvres.
- Reduction of sensor requirements and elimination of explicit state estimation and delay compensation schemes by training NNs on a reduced set of input variables and exposing them directly to non-ideal sensor outputs reconstructed from realistic latency models.
- Centralised IL strategies managing multiple actuators within a single controller, and experimental analysis of the PT benefits deriving from four-wheel-steering and torque-vectoring control based on four in-wheel powertrains.

The remainder is organised as follows: Section 2 describes the test vehicle and PT development framework; Section 3 presents the NMPC used as expert controller as well as the benchmarking PT algorithms; Section 4 deals with the IL framework and resulting NN-based controllers; Section 5 reports the simulation and experimental results; finally, Section 6 draws the main conclusions.

2. Case study vehicle and development framework

2.1. Case study vehicle

The case study electric vehicle (EV) is the driverless PixMoving prototype platform in Figure 1, which is used as a demonstrator within the European project SmartCorners. The vehicle is equipped with four permanent-magnet brushless DC in-wheel motors (IWMs), which allow independent control of the wheel torque levels. The front and rear suspension systems adopt a double wishbone layout. Each axle is equipped with an independent, centralised and by-wire electric steering actuator, covering a steering angle range of ± 27 deg at the wheel. The braking system has a by-wire electro-hydraulic architecture, with

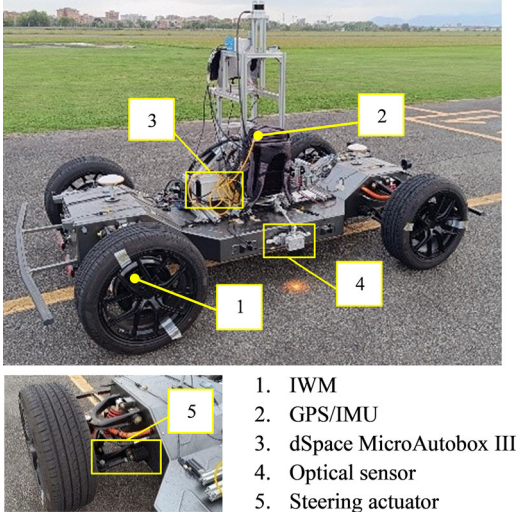


Figure 1. Experimental PixMoving vehicle, with indication of the relevant sensors, actuators, and rapid control prototyping unit.

Table 1. Main parameters of the PixMoving platform.

Description	Symbol	Value	Unit
Vehicle			
Front semi-wheelbase	a_F	0.975	m
Rear semi-wheelbase	a_R	0.925	m
Track width	b	1.470	m
Centre of gravity height	h_{CG}	0.3	m
Wheel radius	R	0.305	m
Total mass	m	511	kg
Roll mass moment of inertia	J_x	325	kg m^2
Yaw mass moment of inertia	J_z	948	kg m^2
Wheel mass moment of inertia	J_w	1.5	kg m^2
IWM			
Rated motor torque	T_{EM}^{rat}	126	Nm
Maximum motor torque	T_{EM}^{max}	220	Nm
Rated motor speed	ω_{EM}^{rat}	230	rpm
Maximum motor speed	ω_{EM}^{max}	550	rpm
EM torque time constant	τ_{EM}	40	ms
EM pure time delay	$\tau_{EM,p}$	50	ms
Steering actuator (SA)			
SA time constant	τ_{steer}	80	ms
SA pure time delay	$\tau_{steer,p}$	130	ms

a closed-loop brake caliper pressure controller. The maximum software-limited vehicle speed is 33 km/h, and the nominal driving range amounts to $\sim$130 km. The main vehicle parameters are reported in Table 1.

The vehicle is equipped with a set of sensors, including: (i) an integrated CHCNAV CGI-410 GNSS (global navigation satellite system) / INS (inertial navigation system) unit using a dual antenna for centimetric-level accuracy, providing vehicle position with a 5 Hz update frequency; and (ii) a Kistler Correvit S-Motion sensor, including a non-contact optical Doppler unit measuring the sideslip angle β – with a 250 Hz sampling frequency, $\pm$20 deg range, 0.01 deg resolution, and $\pm$0.1 deg accuracy – as well as the longitudinal and lateral velocity components.

For rapid control prototyping, the platform includes a dSPACE MicroAutoBox III real-time system, which executes the PT and vehicle dynamics control algorithms, and includes the interfaces with the actuators.

2.2. Path tracking development framework

Figure 2 shows the PT development framework, including:

(a) The set of EV sensors described in Section 2.1.
(b) The sensor processing system, which collects and filters the signals from the vehicle sensors. The main relevant variables are: (i) the absolute linear coordinates of the actual position of the vehicle centre of gravity, X_{meas} and Y_{meas}, in an inertial reference frame; (ii) the yaw angle and yaw rate, ψ_{meas} and $\dot{\psi}_{meas}$; (iii) the longitudinal and lateral accelerations, $a_{x,meas}$ and $a_{y,meas}$; (iv) the longitudinal and lateral velocity components, $v_{x,meas}$ and $v_{y,meas}$; and (v) the wheel speeds, $\omega_{ij,meas}$. Here and in the remainder, the subscript '*meas*' refers to the measured values of the variables, while the subscripts '*des*' and '*act*' refer to the desired and actuated values of the control actions.

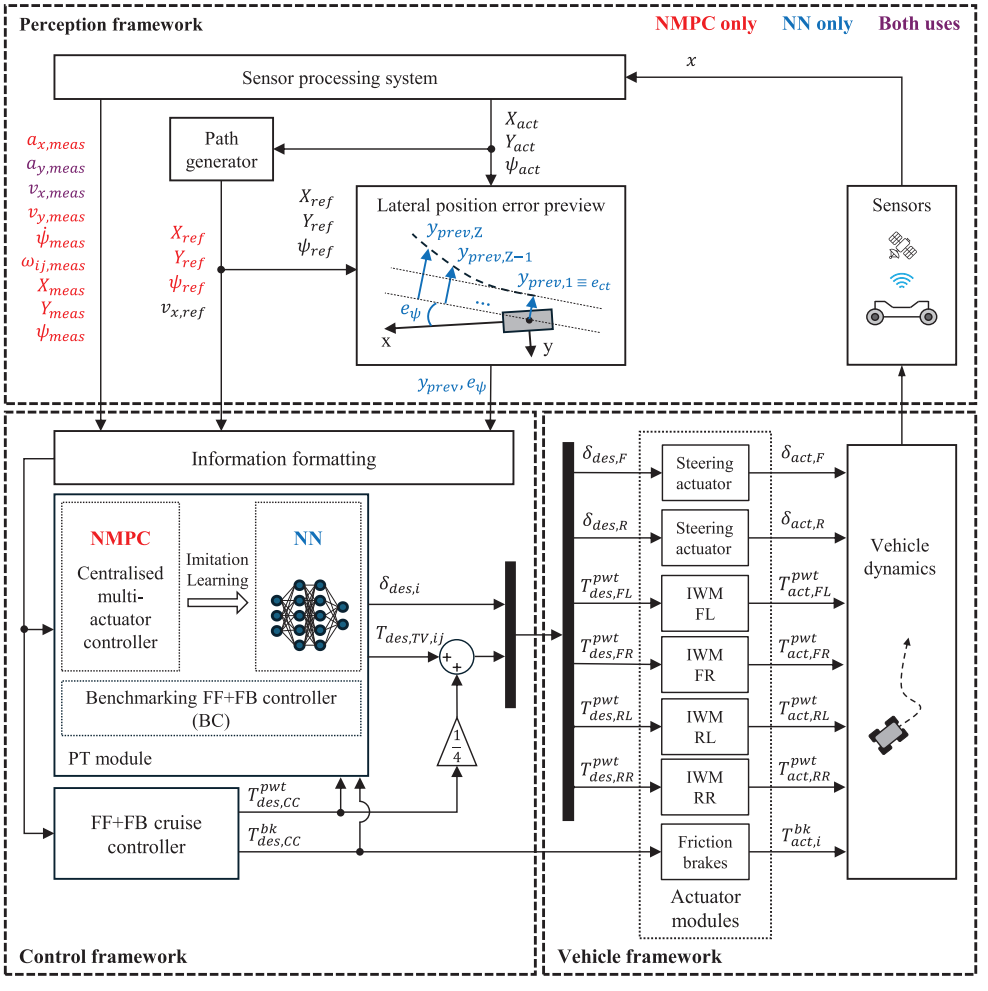


Figure 2. Simplified schematic of the PT development environment, where the indicated colour coding is used to distinguish between the variables that are used as inputs for the NMPC PT, NN-based PT, or both algorithms (for simplicity, no colour coding is used for the benchmarking FF + FB PT controller).

(c) The path generator, which stores the entire reference trajectory (RT), and produces, at each control step, the RT portion – where each point is expressed in terms of X_{ref}, Y_{ref}, ψ_{ref}, and $v_{x,ref}$ – to be used by the PT algorithms, as a function of the current vehicle position.

(d) The lateral position error preview module, which – starting from the current vehicle position and reference trajectory points – computes: (i) the sequence of lateral errors, $y_{prev,z}$, with $z = 1, \ldots, Z$, for multiple ($Z = 20$ in the simulations and experiments of this study) look-ahead points, for a total 10 m look-ahead distance. The $y_{prev,z}$ values are computed in the direction perpendicular to the tangent to the vehicle trajectory in the current EV position. The first element of the sequence, $y_{prev,1}$, represents the lateral error at the centre of gravity, i.e. the cross-track error e_{ct}, hence it is $y_{prev,1} = e_{ct}$. The $y_{prev,z}$ sequence at each time step forms the y_{prev} vector that is

used by the NN-based PT algorithm; and (ii) the vehicle orientation error, e_ψ, for the current EV position.

(e) The information formatting block, which receives the signals from (b) - (d), and arranges them into a control-suitable format, e.g. the features must be scaled to $[-1, 1]$ for the NN controllers.

(f) The cruise control module, based on the combination of feedforward and feedback formulations (hence, the FF + FB notation), which generates the vehicle-level reference powertrain and friction brake torque values, $T^{pwt}_{des,CC}$ and $T^{bk}_{des,CC}$, to track the reference speed, $v_{x,ref}$. The $T^{pwt}_{des,CC}$ contribution is then evenly split among the respective actuators at the corners, while $T^{bk}_{des,CC}$ is divided according to the front and rear friction brake sizes.

(g) The PT module, which targets the RT tracking, and includes the following alternative formulations: (1) the NN-based PT that is the focus of this study; (2) the NMPC for PT that is used for training the NN-based PT in (1); and (3) a benchmarking PT algorithm based on an FF + FB architecture. Each one among (1)−(3) can make the EV operate in the following configurations: 2WS, in which the PT controller only outputs the front steering angle, $\delta_{des,F}$; 4WS, in which the selected algorithm also computes the rear steering angle, $\delta_{des,R}$; and TV, in which – in addition to $\delta_{des,F}$ – the controller generates the reference IWM torque contributions, $T_{des,TV,ij}$ ($i = F, R$ denotes the axle, and $j = L, R$ denotes the EV side), which are used to correct the values from the cruise control module, thus enhancing the path following performance.

(h) The actuator modules, referring to the steering, powertrain, and friction brake actuations, which apply the commands generated by the controllers, introducing the corresponding dynamics.

(i) The vehicle dynamics module, which provides the relevant cornering response variables.

In summary, (a)–(d) represent the perception framework; (e)–(g) the control framework; and (h)–(i) the vehicle framework, which is the selected high-fidelity and experimentally validated model used for the simulations, or the real vehicle prototype in Figure 1 adopted for the experiments.

2.3. Vehicle model validation

The vehicle dynamics model for control system assessment is implemented in the MATLAB/Simulink environment. Given the adoption of an experimental plant with limited data availability, which is typical of real-world implementation scenarios, a relatively simple model configuration was selected. From a mechanical viewpoint, the model includes seven degrees of freedom, three of which are associated with the vehicle body dynamics (longitudinal, lateral, and yaw motions), and four with the rotational wheel dynamics. The tyre model is implemented through the 2002 version of the Pacejka magic formula including relaxation length. The actuator dynamics are considered through pure time delays and first order transfer functions (see Table 1 for parameters), which were identified through dedicated experiments.

The unavailable model parameters, e.g. the tyre model coefficients, were derived through a genetic algorithm optimisation with a large population – containing at least ten individuals per parameter – but a small (three) number of generations, to limit the computational time. The genetic algorithm optimisation was refined through a sequential quadratic programming method. In both optimisation phases, the cost function is the weighted average of the root mean square error values between experiments and simulations – along time t – of the longitudinal speed v_x, vehicle sideslip angle β, yaw rate $\dot{\psi}$, and lateral acceleration a_y.

Figure 3 reports examples of experimental model validation results along four quasi-steady-state and transient manoeuvres, including different tests from those adopted for parameter identification, and covering the longitudinal response as well as the whole lateral acceleration range. The good match between simulations and experiments makes the model a reliable tool for control system design and assessment.

3. Nonlinear model predictive and benchmarking path tracking controllers

3.1. Nonlinear model predictive controller formulations

In this study, the NMPC is used for the offline generation of the optimal control actions to be reproduced by the NNs. Since the computational efficiency of the resulting NMPC algorithm is not the priority (only the NNs need to be real-time capable), the formulation of its internal model can be especially detailed and accurate, e.g. with the inclusion of a double track setup and individual wheel dynamics. In particular, the internal model that is used to predict and optimise the future behaviour of the system is described by the following differential equations, see also the schematics in Figure 4(a):

- Computation of the speed components, $\dot{X}$ and $\dot{Y}$, of the vehicle centre of mass in the inertial reference frame:

$$\dot{X} = v_x \cos(\psi) - v_y \sin(\psi) \tag{1}$$

$$\dot{Y} = v_x \sin(\psi) + v_y \cos(\psi) \tag{2}$$

- Force balance along the longitudinal axis of the vehicle reference system:

$$\dot{v}_x = \frac{1}{m}\left\{\sum_{\substack{i=F,R\\ j=L,R}} [F_{x,ij}\cos\delta_{act,ij} - F_{y,ij}\sin\delta_{act,ij}] - 0.5\rho A C_d v_x^2\right\} + v_y\dot{\psi} \tag{3}$$

- Force balance along the lateral axis of the vehicle reference system:

$$\dot{v}_y = \frac{1}{m}\left\{\sum_{\substack{i=F,R\\ j=L,R}} [F_{x,ij}\sin\delta_{act,ij} + F_{y,ij}\cos\delta_{act,ij}]\right\} - v_x\dot{\psi} \tag{4}$$

- Yaw moment balance:

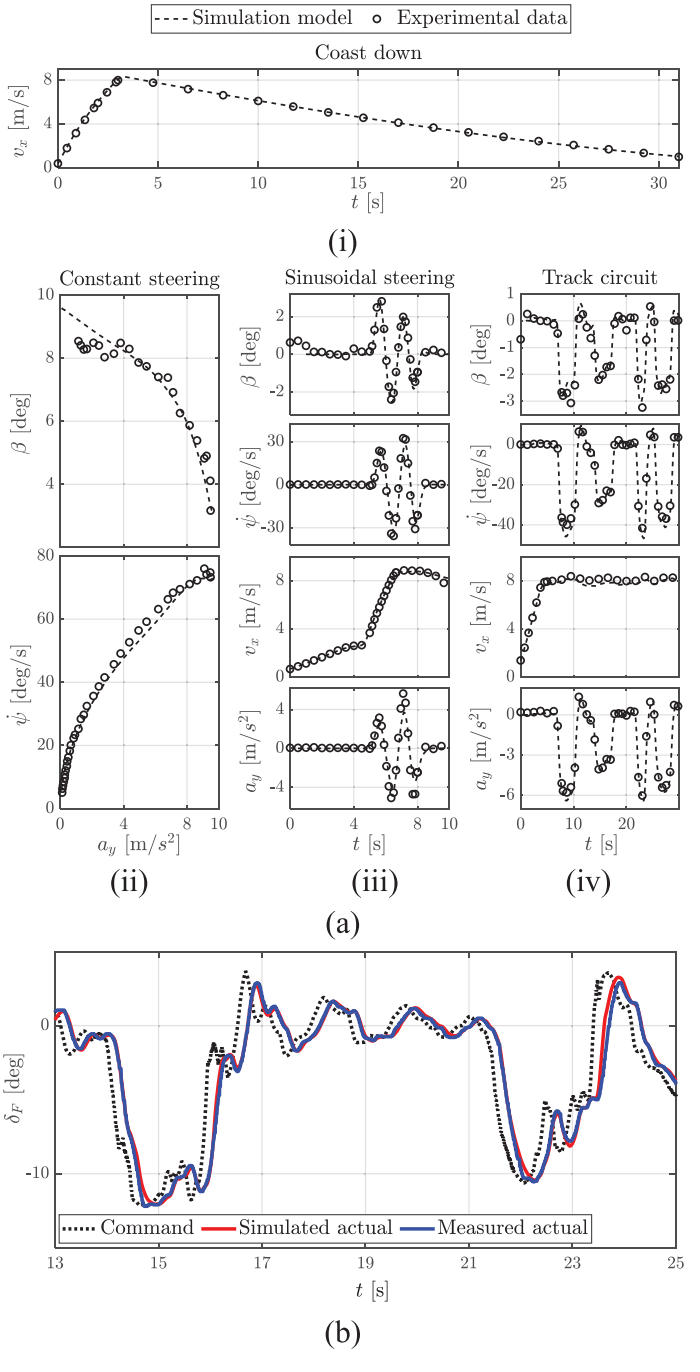


Figure 3. Examples of PixMoving model validation results, with experimental data obtained at the Torino Aero Club proving ground (Italy): (a) vehicle response along a coast down manoeuvre (i), a constant steering test with slowly increasing EV speed (ii), a sinusoidal steering test (iii), and a track lap (iv); and (b) reference and actual front steering angles for a section of the test in (iv).

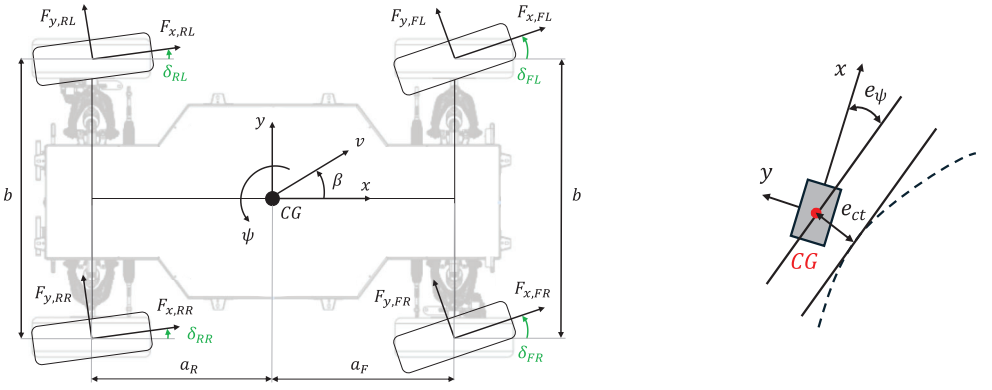


Figure 4. Top views of: (a) the vehicle with the main variables and parameters; and (b) the path tracking errors and vehicle coordinates.

$$\begin{aligned}\ddot{\psi} = \frac{1}{J_z}\Bigg\{ & a_F\left[\sum\nolimits_{j=R,L} F_{x,Fj}\sin\delta_{act,Fj} + F_{y,Fj}\cos\delta_{act,Fj}\right] \\ & - a_R\left[\sum\nolimits_{j=R,L} F_{x,Rj}\sin\delta_{act,Rj} + F_{y,Rj}\cos\delta_{act,Rj}\right] \\ & + \frac{b}{2}\Big\{-\left[F_{x,FL}\cos\delta_{act,FL} - F_{y,FL}\sin\delta_{act,FL}\right] + \left[F_{x,FR}\cos\delta_{act,FR} - F_{y,FR}\sin\delta_{act,FR}\right] \\ & - \left[F_{x,RL}\cos\delta_{act,RL} - F_{y,RL}\sin\delta_{act,RL}\right] + \left[F_{x,RR}\cos\delta_{act,RR} - F_{y,RR}\sin\delta_{act,RR}\right]\Big\}\Bigg\}\end{aligned} \tag{5}$$

- *ij* wheel moment balance:

$$\dot{\omega}_{ij} = \frac{1}{J_w}[T^{pwt}_{act,ij} - T^{bk}_{act,i} - F_{x,ij}R + M_{rr,ij}] \tag{6}$$

- Actuation dynamics:

$$\dot{v}_{act,ab} = \frac{v_{des,ab} - v_{act,ab}}{\tau_{ab}} \tag{7}$$

where v_y is the lateral component of the EV velocity at the centre of mass; $F_{x,ij}$ and $F_{y,ij}$ are the longitudinal and lateral tyre forces at the *ij* corner, which are computed from a simplified version of the Pacejka 2002 tyre model; $\delta_{act,i}$ is the steering angle of the *i* axle after considering the actuator dynamics; ρ is air density; A is the frontal area of the vehicle; C_d is the aerodynamic drag coefficient; ω_{ij} is the angular speed of the respective wheel; $T^{pwt}_{act,ij}$ and $T^{bk}_{act,i}$ are the resulting powertrain and friction brake torque levels at the *ij* corner, after considering the actuation dynamics; $M_{rr,ij}$ is the rolling resistance moment of the wheel; τ_{ab} is the time constant associated with the *ab* actuator, where the subscript *a* indicates the actuator typology – steering, IWM, or friction brake actuation – and *b* the location of the actuator within the EV, e.g. on the front or rear axle or one

of the corners; and $v_{des,ab}$ and $v_{act,ab}$ are the desired and actual values of the actuations. In the internal model, (1)-(7) are complemented by the kinematic formulations for the computation of the longitudinal tyre slips and slip angles, as well as the vertical tyre loads. In particular, the lateral acceleration used for the computation of the lateral load transfer is approximated as $a_y \approx \dot{\psi} v_x$, which enables the prediction of the vertical tyre load variation along the considered horizon.
The internal model is re-arranged in a nonlinear state-space formulation:

$$\begin{cases} \dot{x} = f(x, u, p) \\ z = g(x, u, p) \end{cases} \tag{8}$$

where f and g are nonlinear functions; and x, u, p and z are the state, control input, parameter and output vectors, defined as:

$$x = \begin{bmatrix} X \\ Y \\ \psi \\ v_x \\ v_y \\ \psi \\ \omega_{ij} \\ \delta_{act,i} \\ T^{pwt}_{act,ij} \\ T^{bk}_{act,i} \end{bmatrix}, u = \begin{bmatrix} \delta_{des,i} \\ T_{des,TV,ij} \end{bmatrix}, p = \begin{bmatrix} X_{ref} \\ Y_{ref} \\ \psi_{ref} \\ k_{ref} \\ T^{pwt}_{des,CC} \\ T^{bk}_{des,CC} \\ \lambda_\mu \end{bmatrix}, z = \begin{bmatrix} e_\psi \\ e_{ct} \\ \dot{e}_\psi \\ \dot{e}_{ct} \\ u \end{bmatrix} \tag{9}$$

with λ_μ being the scaling factor on the tyre-road friction level, which is imposed to be equal for the four corners. In (9), the cross-track and heading errors – and the respective time derivatives – are given by:

$$e_{ct} = [Y_{ref} - Y]\cos(\psi_{ref}) - [X_{ref} - X]\sin(\psi_{ref}) \tag{10}$$

$$e_\psi = \psi_{ref} - \psi \tag{11}$$

$$\dot{e}_{ct} = v_y \cos(e_\psi) - v_x \sin(e_\psi) \tag{12}$$

$$\dot{e}_\psi = k_{ref}\, v_x - \dot{\psi} \tag{13}$$

u includes a single reference average value of the steering angle for each axle, while (3)-(5) consider the individual angles for each corner, according to the Ackermann steering geometry of the specific suspension systems.

The NMPC optimal control problem (OCP) is formalised as the minimisation of a cost function J subject to (s.t.) a set of constraints:

$$\begin{aligned}
\min_{\{u_k\}_{k=0}^{N-1}} J &= l_0(x_0, u_0, p_0) + \sum_{k=1}^{N-1} \left[l(x_k, u_k, p_k) \right] + l_N(x_N, p_N) \\
&\text{s.t.} \\
x_{k+1} &= f_d(x_k, u_k, p_k), k = 0, \ldots, N-1 \\
z_k &= g_d(x_k, u_k, p_k), k = 0, \ldots, N-1 \\
z_N &= g_d(x_N, p_N) \\
x_{\min,q} &\leq x_{k,q} \leq x_{\max,q}, \forall q = 1, \ldots, n_q, k = 0, \ldots, N \\
u_{\min,r} &\leq u_{k,r} \leq u_{\max,r}, \forall r = 1, \ldots, n_r, k = 0, \ldots, N-1
\end{aligned} \tag{14}$$

where:

$$\begin{aligned}
l_0(x_0, u_0, p_0) &= \frac{1}{2}[z_0 - z_{ref,0}]^T W_0 [z_0 - z_{ref,0}] \\
l(x_k, u_k, p_k) &= \frac{1}{2}[z_k - z_{ref,k}]^T W [z_k - z_{ref,k}], k = 1, \ldots, N-1, \\
l_N(x_N, p_N) &= \frac{1}{2}[z_N - z_{ref,N}]^T W_N [z_N - z_{ref,N}]
\end{aligned}$$

in which:

$$z_0 = \begin{bmatrix} e_{\psi,0} \\ e_{ct,0} \\ \dot{e}_{\psi,0} \\ \dot{e}_{ct,0} \\ u_0 \\ \Delta u_0 \end{bmatrix}, z_k = \begin{bmatrix} e_{\psi,k} \\ e_{ct,k} \\ \dot{e}_{\psi,k} \\ \dot{e}_{ct,k} \\ u_k \end{bmatrix}, z_N = \begin{bmatrix} e_{\psi}(x_N, p_N) \\ e_{ct}(x_N, p_N) \end{bmatrix}$$

where k and N are indices defining the considered prediction step and total number of steps along the selected finite horizon; l_0, l, and l_N are the stage costs at the steps 0, k, and N, which focus on the minimisation of e_{ct}, e_{ψ}, and their time derivatives, with due consideration of the control effort aspects; f_d and g_d are the discretised versions of f and g; $x_{min,q}$ and $x_{max,q}$ are the elements of the vectors – with n_q elements – defining the minimum and maximum allowed values of the states; $u_{min,r}$ and $u_{max,r}$ are the entries of the vectors – with n_r elements – defining the control input constraints; z_0, z_k and z_N include the system outputs at the initial, intermediate and final prediction steps; $z_{ref,0}$, $z_{ref,k}$ and $z_{ref,N}$ are null vectors setting the reference values of the outputs; W_0, W, and W_N are diagonal weight matrices; and Δu_0 is the increment of the first control move with respect to the last one applied to the plant, which is considered for chattering prevention. The constraints in (14) are imposed as hard bounds in the NMPC OCP, i.e. they are not relaxed through slack variables. Therefore, the constraint violations are not penalised in the cost function, but are excluded from the feasible set, within the numerical tolerances of the solver. The primary

purpose of these bounds is to enforce the actuator-related limits, i.e. the admissible steering angle range and the selected electric machine torque range. Conversely, the bounds on the vehicle dynamics states are intentionally set to be permissive, so that they act as numerical guardrails without excessively restricting the predicted vehicle response. Stability, agility, and tracking behaviour are instead shaped through the calibration of the cost function weights, and then assessed a posteriori through the simulation and experimental results. λ_μ is considered constant along the horizon, not to involve any form of road preview in the implementation. (1)–(14) are reported for the comprehensive case in which all actuators are available. For achieving the individual 2WS, 4WS and TV actuation configurations, the control inputs corresponding to the unused actuators are set to zero.

3.2. NMPC implementation

The NMPC algorithms are implemented through the acados toolkit [31] a modular and computationally efficient framework for formulating and solving nonlinear OCPs. The OCP is discretised through a multiple shooting scheme with an implicit Runge–Kutta integrator. The prediction horizon ($T = 1$ s) is divided into $N = 20$ shooting intervals, $\Delta t_k = t_{k+1} - t_k$, with $k = 0, \dots, N-1$, according to a non-uniform distribution defined by $t_k = T(k/N)^2$. The setup quadratically clusters the steps towards the beginning of the prediction horizon, thus improving the short-term accuracy without increasing the computational effort. Given the level of detail and numerical stiffness of the prediction model, each shooting interval is divided into two internal integration steps. The average internal integration step is 25 ms, with the non-uniform grid leading to smaller steps at the beginning of the prediction horizon and larger ones (50 ms) towards its end. Although this average value may appear relatively coarse for the wheel dynamics, the reduced tyre model adopted for the NMPC prediction, the low longitudinal slip levels observed in the considered manoeuvres, and the use of an implicit Runge–Kutta integration scheme make this setting adequate for the presented application. For operating conditions involving higher slip levels or faster wheel speed transients, additional internal integration steps would become necessary. The resulting nonlinear program is solved by a sequential quadratic programming (SQP) scheme, with the HPIPM QP (high-performance interior-point-method quadratic programming) solver in partial condensing mode. Tight stopping tolerances (10^{-4}) are adopted to improve solution accuracy and convergence robustness. The controller is deployed in Simulink via the S-function generated by acados. The NMPC cost function weights were empirically calibrated to provide desirable performance across the range of operating conditions.

3.3. Benchmarking controllers

The proposed NMPCs and NN-based implementations are compared with benchmarking controllers (BCs), which – for the steering input computation – are inspired by the formulation in [6], combining FF and FB contributions. More specifically:

(a) In the 2WS configuration, the FF term, δ_F^{FF}, is obtained from: (i) the reference curvature k_{ref}, taken with a spatial forward shift, d_{shift}, to compensate for the delays from sensors and actuators; (ii) a term, $[1 + K_{US} v_x{}^2][a_F + a_R]$, including

the understeer coefficient K_{US}, to account for the marginally oversteering nature of the PixMoving platform; and (iii) an empirical calibration factor K_l. The FB term is a proportional derivative (PD) controller on the cross-track error:

$$\delta_F = \delta_F^{FF} + \delta_F^{FB} \tag{15}$$

$$\delta_F^{FF} = K_l k_{ref}(d_{shift})[1 + K_{US} {v_x}^2][a_F + a_R] \tag{16}$$

$$\delta_F^{FB} = K_P^F e_{ct} + K_D^F \dot{e}_{ct} \tag{17}$$

(b) In the 4WS configuration, the front steering terms are computed according to (15)-(17), while δ_R is obtained from a PD controller on the heading error e_ψ:

$$\delta_R = K_p^R e_\psi + K_d^R \dot{e}_\psi \tag{18}$$

(c) In the TV configuration, (15)-(17) are complemented by a PD formulation on the heading error to compute the reference direct yaw moment M_z:

$$M_z = K_p^{TV} e_\psi + K_d^{TV} \dot{e}_\psi \tag{19}$$

Starting from M_z, a lower-level controller based on heuristics outputs the individual reference torque levels at the corners, which are then split between IWMs and friction brakes.

4. The imitation learning toolchain

4.1. Neural network architecture and training

IL aims to replace a control expert with an NN capable of reproducing the same desirable behaviour at much lower computational cost. In typical IL formulations, the NN is trained from data generated through simulations with the control expert, e.g. represented by an NMPC in [22], and uses the complete input set required by the expert, including the full vehicle state vector. This conventional approach entails that all the state variables must be measured or estimated on the real vehicle, which increases complexity and sources of noise / disturbance. The control expert that generates the control actions for training the NN is usually run offline with simulated signals that do not include measurement or estimation errors, and – even if such errors are reproduced in the simulation environment – a typical advanced controller, such as an NMPC, has limited robustness with respect to the accuracy and latency of its input variables. As a consequence, when implemented online on the experimental vehicle, to achieve the expected performance, the resulting NN must be provided with signals at the same quality level as those used for the expert agent simulations. However, in real-world applications, it would be much more convenient to use a reduced or different set of readily available input variables for the vehicle-deployed NN, without having to implement advanced sensor fusion or delay compensation techniques.

To address the previous issues, in this study the NN is trained through the NMPC control actions: these are the ones entering the vehicle plant including the complete actuator models, but with a limited and different input variable set, given by $x_{NN} = [y_{prev}\ e_\psi\ a_y\ v_x]^T$, while the NMPC requires x_0, u_0, and p_0. x_{NN} only includes tracking error

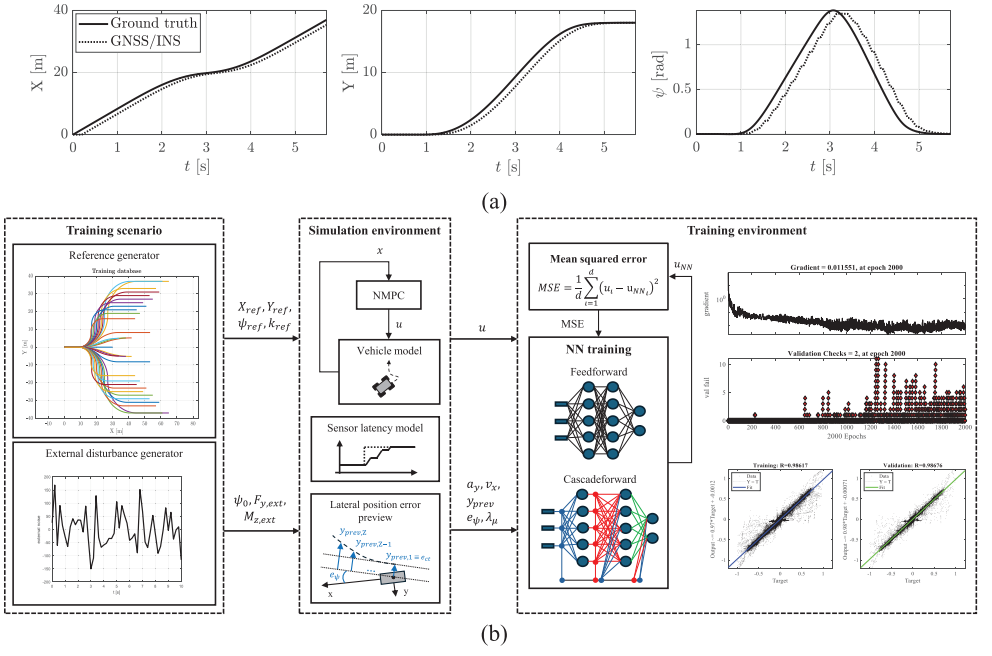


Figure 5. (a) Reconstructed GNSS / INS signals against the ground truth; and (b) Schematic of the simulation-based process for offline NN training.

and vehicle dynamics information that is available through the GNSS / INS unit, without any delay compensation. This is a crucial aspect, since in the specific application y_{prev} and e_ψ are purposely derived from GNSS signals that are affected by > 200 ms pure time delays, and thus are not directly usable for NMPC operation, see also Figure 5(a), reporting the X, Y and ψ coordinates as reconstructed from the GNSS / INS, together with the corresponding ground truth. Very importantly, it was verified that the NMPC generates an unacceptable oscillatory response if directly provided with its extensive input variable set with the GNSS / INS delays. In the proposed approach, the sensor non-idealities are reproduced in the simulation environment to generate x_{NN}, while the simulated NMPC is provided with the ideal input set. In this way, the NMPC operates at its best, while the NN learns from realistic signals and implicitly compensates for delays and inaccuracies.

The resulting NN controller imitates the NMPC policy, including its stability-oriented behaviour in critical and previously unseen operating conditions, as it will be shown in the following Figures 10–12. However, since the adopted feedforward NN does not solve a constrained OCP online, hard constraints on the vehicle dynamics states cannot be enforced in the same direct sense as in the NMPC formulation; their satisfaction is only indirectly promoted by the imitated policy and the actuator bounds applied at the implementation level. Nevertheless, if the network has successfully learned the NMPC behaviour across the state and parameter space, it is expected to behave similarly to the NMPC also in proximity of the constraint thresholds, and the satisfaction of the relevant operating limits can only be assessed a posteriori in simulation. In other words, the NN will not be able to enforce hard limits by architecture; it will only learn to behave like the NMPC, i.e., in expectation, the limits will be approximately respected. From the analysis of the training and testing data, in

the specific simulations and experiments of this study, both the NMPC and NN-generated trajectories remain within the desirable vehicle dynamics envelope in the considered scenarios, i.e., well inside the set constraints. As a future development, the NN can be trained on a database that is rich in examples where these limits are active. In this respect, preliminary analyses have been carried out on NNs that are the evolution of those of this study, and imitate NMPC path tracking algorithms targeting only road accident prevention scenarios. In these conditions, it was verified that: (i) a hard constraint on the sideslip angle magnitude during transient drifting is met across a set of emergency manoeuvres; and (ii) other constraints, e.g. on the minimum level of the exit speed, are generally matched very well, with a few exceptions in which the constraint is marginally violated. These very occasional and low-magnitude violations do not represent an issue, since also an advanced implicit NMPC for online operation, e.g. see the one in [32], would be characterised by hard constraints on the control actions, but not on the state variables, which would be covered by soft constraints.

The overall IL architecture is illustrated in Figure 5(b). In its baseline form, the control policy is realised through a fully connected feedforward NN consisting of two hidden layers with $h_1 = 32$ neurons and $h_2 = 16$ neurons for 2WS and 4WS, and 64 and 16 neurons for TV. Such a small architecture is easy to train and deploy, while helping generalisation and reducing the risk of overfitting. The hidden layers employ the tanh (hyperbolic tangent) activation function, enabling the NN to capture the nonlinear tyre and vehicle dynamics characteristics, while the output layer uses a linear activation function, ensuring that the steering angles are not artificially saturated:

$$\begin{aligned} &u_{NN} = \Gamma_3 \tanh(\Gamma_2 \tanh(\Gamma_1 x_{NN} + B_1) + B_2) + B_3, \\ &\text{with } x_{NN} \in \mathbb{R}^{\zeta} = [\mathrm{y}_{prev}\ e_{\psi}\ a_y\ v_x]^{\mathrm{T}} \\ &u_{NN} \in \mathbb{R}^{\varsigma} = [\delta_{des,i}\ T_{des,TV,ij}]^{\mathrm{T}} \\ &\Gamma_1 \in \mathbb{R}^{\mathrm{h}_1 \mathrm{x}\, \zeta}, B_1 \in \mathbb{R}^{\mathrm{h}_1}, \Gamma_2 \in \mathbb{R}^{\mathrm{h}_2 \mathrm{x}\, \mathrm{h}_1}, B_2 \in \mathbb{R}^{\mathrm{h}_2}, \Gamma_3 \in \mathbb{R}^{\varsigma\ \mathrm{x}\ \mathrm{h}_2}, B_3 \in \mathbb{R}^{\varsigma} \end{aligned} \tag{20}$$

where u_{NN} is the NN output vector; Γ_1, Γ_2 and Γ_3 are the weight matrices of the respective NN layers; and B_1, B_2 and B_3 are the NN biases.

The NN architecture and size were selected through an empirical design process. The initial model was intentionally oversized, with three hidden layers of 128, 64, and 32 neurons, and was then progressively reduced until satisfactory performance was retained with the smallest possible architecture. This choice was motivated by a parsimony criterion: although no clear overfitting was observed from the validation loss alone, unnecessarily large networks may still show poorer generalisation and reduced robustness outside the training conditions.

Regarding the network type, a fully connected feedforward neural network (FFNN) was preferred over more complex alternatives, such as long short-term memory (LSTM) networks. The rationale was to adopt the simplest architecture capable of reproducing the expert NMPC behaviour with satisfactory accuracy. A simpler structure reduces computational complexity and limits additional sources of design uncertainty. Since the FFNN was found to imitate the NMPC satisfactorily also under strongly nonlinear vehicle dynamics, as shown in Figure 9, it was considered adequate for this work.

The reference generator block outputs the reference trajectories of the training dataset. The 40 considered trajectories cover consecutive constant-curvature turns in opposite

directions, either separated by intermediate straight-line segments or involving abrupt steering transitions from a curve to the following one. The dataset is obtained by running the vehicle model with the NMPC algorithms (for the 2WS, 4WS, and TV configurations) tracking the defined RTs, which were designed to cover a broad range of lateral accelerations, up to limit handling conditions. To enhance robustness, the dataset was augmented with scenarios including lateral force ($F_{y,ext}$ in the figure) and yaw moment ($M_{z,ext}$) disturbances – with linearly growing values from zero to defined saturation levels – as well as initial pose misalignments (defined through the initial yaw angle ψ_0), exposing firstly the NMPCs and then the NNs to a wide range of cross-track and heading errors. The dataset was partitioned into training, validation and testing subsets, corresponding to 65, 25 and 10% of the total samples. The NNs were trained using a mean squared error (see Figure 5(b)) loss function on the control command vector, combined with L2 regularisation to promote generalisation [33]. The training was performed through the Levenberg–Marquardt second-order optimisation method, with a fixed epoch cap and patience-based early stopping on the validation set, allowing a maximum of 15 consecutive validation failures.

In a second stage, the current λ_μ value was explicitly introduced in the NN for the TV configuration, as an input feature to improve the NN ability to cope with extreme friction level variations. The expert was provided with the same variable friction-related input, maintained constant along the prediction horizon. However, it was verified that simply including λ_μ as an element of the feature vector of the fully connected feedforward NN is insufficient to achieve meaningful dependence of the NN outputs on the tyre-road friction level. Several strategies can be adopted to increase the influence of λ_μ, such as: (i) providing linear combinations of λ_μ and other features; (ii) repeating λ_μ within the input vector; or (iii) modifying the NN architecture. The last approach was adopted, by changing the NN layout from feedforward to cascade-forward [34], in which each layer receives inputs both from the preceding layers and directly from the feature vector:

$$\begin{aligned}
&u_{CF} = \Gamma_{3x}x_{CF} + \Gamma_{31}\tanh(\Gamma_{1x}x_{CF} + B_1) \\
&\quad + \Gamma_{32}\tanh\,(\Gamma_{2x}x_{CF} + \Gamma_{21}\tanh(\Gamma_{1x}x_{CF} + B_1) + B_2) + B_3, \\
&\text{with } x_{CF} = [x_{NN};\ \lambda_\mu], x_{CF} \in \mathbb{R}^{\zeta+1} \\
&\Gamma_{1x} \in \mathbb{R}^{h_1 \mathrm{x}\,(\zeta+1)}, B_1 \in \mathbb{R}^{h_1}, \\
&\Gamma_{2x} \in \mathbb{R}^{h_2 \mathrm{x}(\,\zeta+1)}, B_2 \in \mathbb{R}^{h_2}, \Gamma_{21} \in \mathbb{R}^{h_2 \mathrm{x}\, h_1}, \\
&\Gamma_{3x} \in \mathbb{R}^{\varsigma \mathrm{x}\,(\zeta+1)}, B_3 \in \mathbb{R}^{\varsigma}, \Gamma_{31} \in \mathbb{R}^{\varsigma \mathrm{x}\, h_1},\ \Gamma_{32} \in \mathbb{R}^{\varsigma \mathrm{x}\, h_2}
\end{aligned} \tag{21}$$

where the terms $\Gamma_{\varepsilon\theta}$ are the weight matrices, with the subscript ε indicating the number of the layer receiving the specific output, and θ the number of the layer providing the input (if $\theta = x$, the signal is from the input layer). The training database remained the same as for the feedforward NN, but the manoeuvres were repeated – including disturbances and pose misalignments – for different λ_μ levels, thus originating the so-called TV NN w λ_μ algorithm.

4.2. Implementation and computational cost aspects

The NNs were trained through the Deep Learning Toolbox of MATLAB, and were then deployed in Simulink via the gensim function, which generates a masked Simulink subsystem implementing the NN and supporting code generation. Figure 6 and Table 2 report

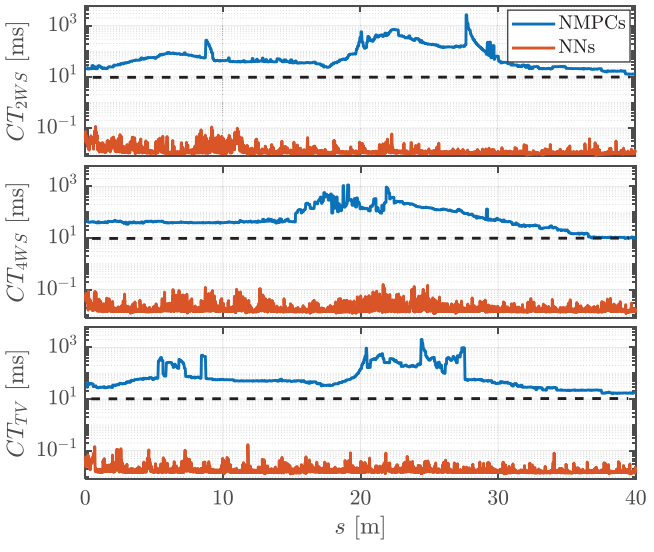


Figure 6. Comparison of the CPU times (*CT*) of the NMPCs and NNs along the spatial coordinate *s* of the Double turn manoeuvre in Figure 7. The dashed horizontal lines indicate the controller sampling time (10 ms).

Table 2. Example of *CT* comparisons between the considered NMPCs and feed-forward NNs, along the Double turn manoeuvre in Figure 7.

Configuration	mean(CT) [ms]	max(CT) [ms]
NMPCs		
2WS	101	2678
4WS	85	1129
TV	109	2046
NNs		
2WS	1e-2	1e-1
4WS	2e-2	2e-1
TV	2e-2	2e-1

the CPU times, referred to as *CT* (obtained on a laptop with an Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz), for the NMPCs and resulting NNs, highlighting the NN ability to achieve reductions of the mean and maximum *CT* values by five orders of magnitude. The dashed horizontal lines in Figure 6 represent the sampling time of the controllers, set to 10 ms: since the *CT* values for the NMPCs exceed this threshold, the NMPCs are not real-time implementable, whereas the NNs remain well below the threshold, and are therefore suitable for real-time operation on the specific CPU with very high safety margin.

5. Path tracking performance assessment

5.1. Reference trajectories and key performance indicators

Given the limitations of the specific EV platform, all simulations and experiments are covered at the constant vehicle speed of 30 km/h. For comprehensiveness, a total of six

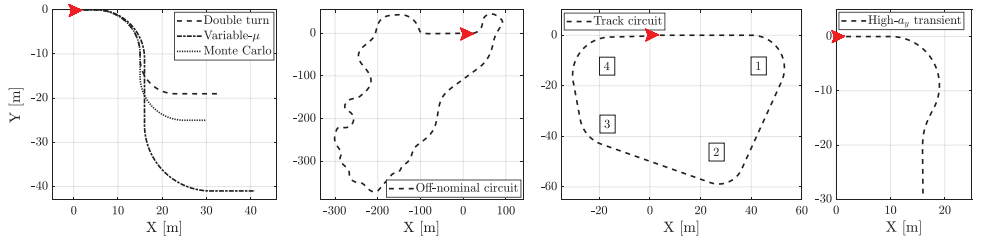


Figure 7. Reference trajectories for the implemented simulations and experiments. In all cases, the initial position of the vehicle is indicated by the red triangles.

reference trajectories are used (Figure 7). Three of them are S-shaped: (a) the 'Double turn', with two consecutive corners with 10 and 8 m radii, and a direct transition from the first to the second turn, bringing $|a_y|$ values up to ~8 m/s^2, i.e. involving limit handling operation; (b) the 'Variable-μ' trajectory, which is the least demanding path curvature-wise, but is tackled under major and sudden variations of the tyre-road friction factor, ranging from 0.3 to 1; and (c) the 'Monte Carlo' path, inspired by (a), but with a lower level of difficulty, i.e. with larger radii and the inclusion of a 5 m straight-line section between the two turns. The remaining trajectories are: (d) an 'Off-nominal circuit' with corners that are not part of the training database, which includes clothoidal curves, turns in which the curvature suddenly changes, e.g. with steps, and sequences of more than two consecutive turns; (e) the 'Track circuit', consisting of constant-radius turns of different lengths and straight segments, with moderate-to-high difficulty, i.e. corresponding to average and peak $|a_y|$ values of ~ 5.5 and 7 m/s^2; and (f) the 'High-a_y transient scenario', in which – following an initial 10 m long straight segment – the vehicle negotiates a first tight curve with a 9 m radius and $|a_y|$ peaks of ~8 m/s^2, then abruptly changes direction to continue in a short section with increasing curvature of opposite sign (clothoidal transition), before a final straight-line segment.

These trajectories are used for: (i) simulations in conditions close to those of the training dataset (Double turn); (ii) simulations in scenarios significantly different from those covered by the training phase, including various curvature profiles, variable tyre-road friction conditions, and Monte Carlo robustness analyses (Off-nominal circuit, Variable-μ, Monte Carlo, and Track circuit); and (iii) preliminary proof-of-concept experiments with the PixMoving platform (Track circuit and High-a_y transient scenario). Hence, the Track circuit RT was adopted for both the variable-μ simulations and high-friction experiments.

To evaluate the effectiveness of the PT algorithms, the following key performance indicators (KPIs) are used: (1) e_{ct}^{max}, i.e. the maximum absolute value of the cross-track error e_{ct} along the considered manoeuvre; (2) e_{ct}^{rms}, i.e. the root mean square (rms) value of e_{ct}; (3) e_{ψ}^{max}, i.e. the maximum absolute value of the heading error e_{ψ}; (4) e_{ψ}^{rms}, i.e. the rms value of e_{ψ}; (5) β^{max}, i.e. the maximum absolute value of the vehicle sideslip angle β; and (6) β^{rms}, i.e. the rms value of β.

5.2. Simulation results

5.2.1. Benefits of the nonlinear model predictive controllers over the benchmarking algorithms

This analysis – summarised in Figure 8 and Table 3 – compares the implemented NMPCs and BCs along the Double turn trajectory. During the simulations, it was observed that a constant parametrisation of the NMPCs robustly performs across scenarios corresponding to normal driving and limit handling conditions. On the contrary, the BC settings providing a good trade-off between PT performance and intervention smoothness in normal driving are not suitable for limit handling operation, and vice versa. Therefore, differently from the NMPC simulations, the main BC parameters were tuned to achieve the highest performance in the specific manoeuvre.

Although the BCs deliver comparable behaviour to the NMPCs in the initial turn, their non-predictive nature limits their ability to effectively manage the abrupt cornering direction reversal and the EV realignment in the final straight segment. The NMPC KPIs are consistently better than those for the BCs. Even more, the 2WS NMPC configuration – the most basic one actuator-wise – performs better – from every viewpoint – than the best BC configuration. The additional chassis actuators of the 4WS and TV configurations are mainly used to increase agility, e.g. the rear steering input is mostly applied in counterphase. The introduction of rear-wheel-steering and torque-vectoring significantly reduces the cross-track error for both BCs ($\sim$35 and $\sim$23% e_{ct}^{max} and e_{ct}^{rms} reductions brought by the 4WS configuration, and $\sim$49 and $\sim$42% by the TV one) and NMPCs ($\sim$39 and $\sim$22% e_{ct}^{max} and e_{ct}^{rms} reductions brought by the 4WS configuration, and $\sim$62 and $\sim$8% by the TV one). Obviously, since the 2WS NMPC already provides very good tracking, the absolute values of the reductions are lower for the multi-actuated NMPC cases, although the variations are similar percentage-wise. These important results justify the need to implement centralised multi-variable algorithms that combine the high PT performance capability of the NMPCs with the computational efficiency of the BCs, which is the objective of the developed NN-based PT implementations.3

5.2.2. Effectiveness of the IL toolchain in nominal conditions

To assess the effectiveness of the IL toolchain, the trained NNs are evaluated with the closed-loop simulation framework in Figure 2, along the Double turn scenario. The analysis focuses on the direct comparison between the NMPC- and NN-generated control inputs, together with the resulting NN-related PT performance. The IL objective is that the NNs reproduce the behaviour of the master controller as closely as possible, providing control actions with comparable amplitude, timing, and evolution over the manoeuvre. As shown in Figure 9 and Table 3, the target is achieved, since the NNs generate control signals that closely follow the NMPC ones also for demanding operating conditions, and the corresponding cross-track error range remains similar. As expected, the NMPCs still deliver slightly superior performance, particularly in the most critical phases. Nonetheless, the NNs significantly outperform the corresponding BCs, with $\sim$49 and $\sim$39% e_{ct}^{max} and e_{ct}^{rms} reductions for the 2WS configuration; $\sim$48 and $\sim$60% reductions for the 4WS configuration; and $\sim$23 and $\sim$13% reductions for the TV configuration. Under these circumstances, even small deviations in the NN decisions can drive the EV towards conditions

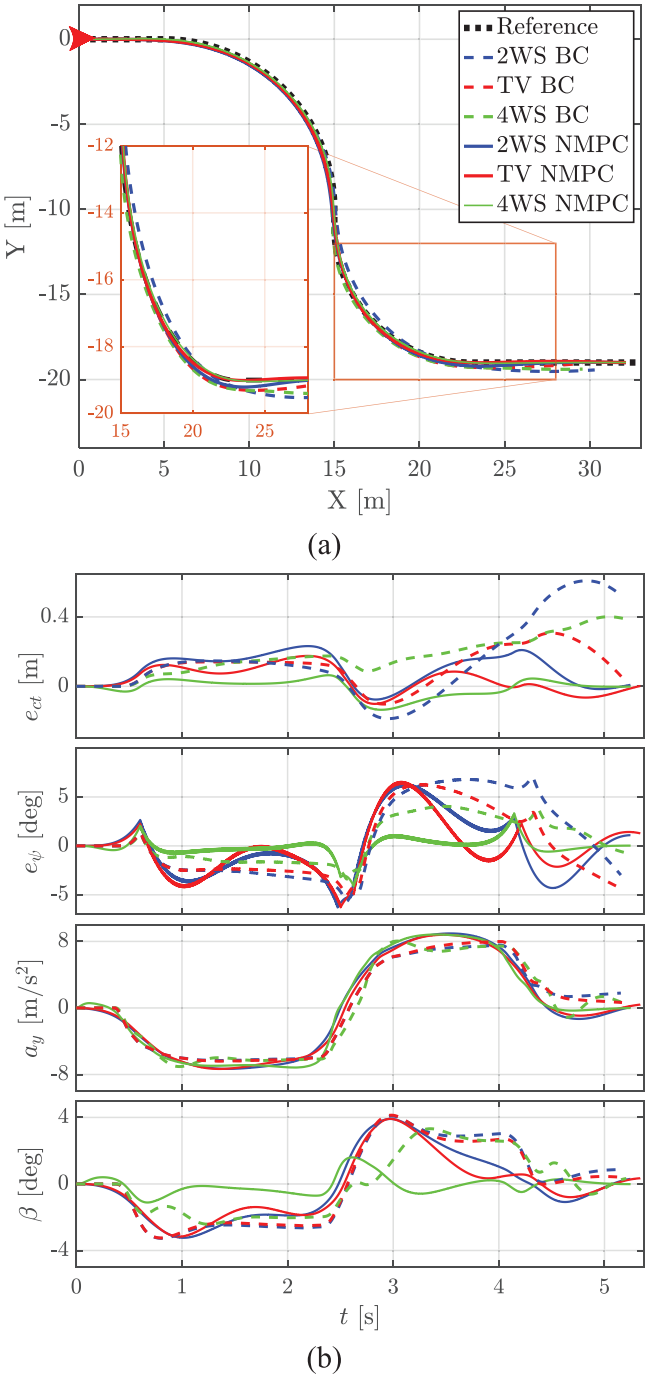


Figure 8. Double turn simulation results: (a) reference and resulting trajectories for the considered controller configurations; and (b) time (t) profiles of e_{ct}, e_{ψ}, a_y and β.

Table 3. Double turn simulation KPIs for the BCs, NMPCs and NNs applied to the considered actuation configurations.

Configuration	e_{ct}^{max} [m]	e_{ct}^{rms} [m]	e_{ψ}^{max} [deg]	e_{ψ}^{rms} [deg]	β^{max} [deg]	β^{rms} [deg]
BCs						
2WS	0.61	0.26	7.28	3.99	4.08	2.41
4WS	0.40	0.20	4.15	2.22	3.32	1.84
TV	0.31	0.15	6.36	3.32	4.13	2.30
NMPCs						
2WS	0.23	0.13	6.30	3.00	3.91	2.02
4WS	0.14	0.05	4.12	0.93	1.60	0.54
TV	0.18	0.08	6.55	2.69	3.91	1.81
NNs						
2WS	0.31	0.16	8.07	3.65	4.98	2.36
4WS	0.21	0.08	5.17	1.58	1.00	0.45
TV	0.24	0.13	6.65	2.83	4.05	1.72

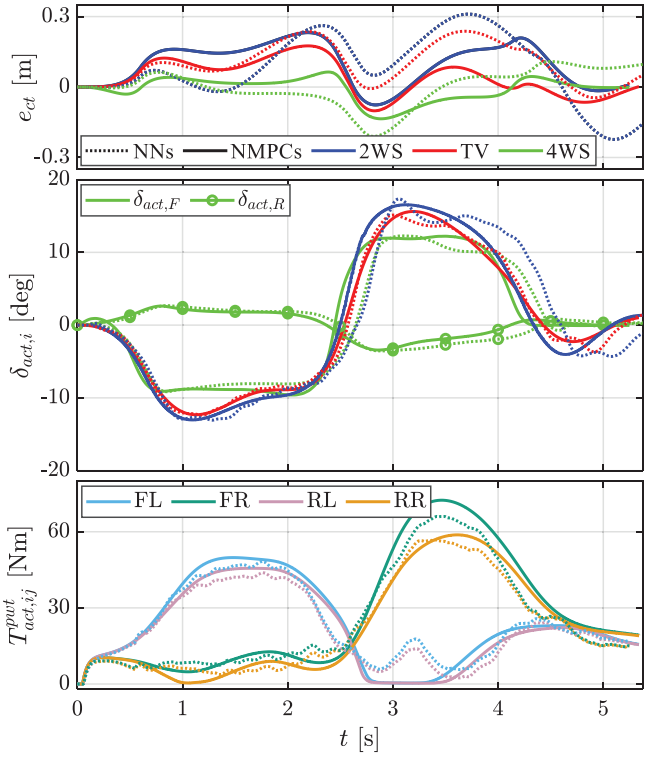


Figure 9. Double turn simulation results: comparison between NMPCs and NNs, in terms of time profiles of e_{ct}, $\delta_{act,F}$ (for all configurations), $\delta_{act,R}$ (for 4WS), and $T^{pwt}_{act,ij}$ (for TV). The dotted lines refer to the NN algorithms.

that are poorly represented in the training dataset. The ability of the NNs to maintain satisfactory performance in this out-of-distribution scenario provides evidence that the proposed IL framework achieves a good level of generalisation.

5.2.3. Robustness analyses of the neural network-based path tracking algorithms

The robustness analysis is based on three simulation typologies:

- Off-nominal circuit, which confirmed the generalisation capability of the offline-generated NN-based PT algorithms. In Figure 10, the 2WS NN completes the manoeuvre while maintaining a low cross-track error (with a 0.25 m peak value), comparable to the one in Section 5.2.2. Looking at the same figure, it is possible to appreciate the capability of the network to maintain accurate tracking capability spanning from smooth curvature transitions and low $|a_y|$ values, up to severe curvature transitions and high $|a_y|$ values in proximity of the handling limits. 2WS BC exhibits comparable tracking performance in the non-demanding segments, but shows clear limitations when the dynamics become fast and nonlinear.
- Monte Carlo analysis, performed by varying several vehicle and tyre parameters, namely: (i) the scaling factor, λ_μ, on the tyre-road friction level; (ii) and (iii) the scaling factors, k_x and k_y, on the longitudinal slip stiffness and cornering stiffness of the tyres; (iv) the vehicle mass, modified through its increment Δm; and (v) the time constants, $\tau_{\delta F}$ and $\tau_{\delta R}$, of the steering actuators. Except for the EV mass, the parameters are varied by $\pm 20\%$ with a normal distribution, see the subplots on the main diagonal of Figure 11(a), and are randomly combined for a set of 250 simulations, see the other subplots of the figure. The vehicle dynamics model is assigned the Monte Carlo-generated parameter combinations, while the NN parametrisation is not varied from the one used in Section 5.2.2. The BCs were recalibrated for the specific reference path and nominal EV parametrisation. In the plane reporting the normalised rms values of cross-track and heading errors, $\hat{e}_{ct}^{rms} = e_{ct}^{rms}/\max(e_{ct}^{rms})$ and $\hat{e}_{\psi}^{rms} = e_{\psi}^{rms}/\max(e_{\psi}^{rms})$, see Figure 11(b), the NN-related point clouds are generally smaller and better bounded than the BC ones. This is confirmed in Table 4, which – for each controller – reports the coordinates – $\xi_{\hat{e}_{ct}^{rms}}$ and $\xi_{\hat{e}_{\psi}^{rms}}$ – of the centroids of the cloud points, and the radius ϱ of the centroid-centred circumferences that include 90% of the points. For the NN-based PT algorithms, see the insets of Figure 11(b), ϱ is one order of magnitude smaller than for

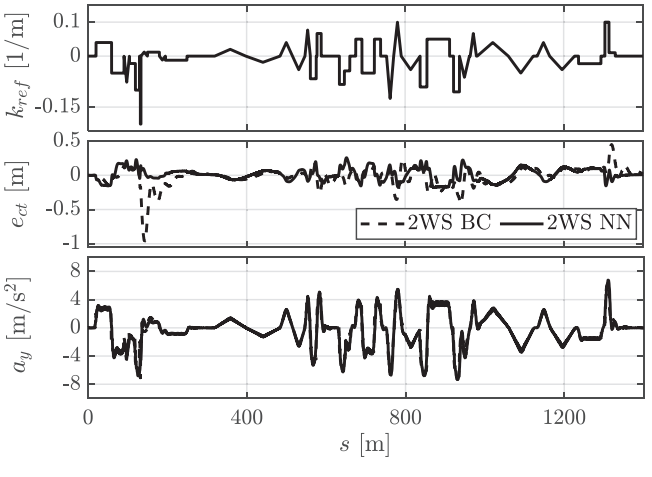


Figure 10. Off-nominal circuit simulation results: k_{ref}, e_{ct} and a_y as a function of s.

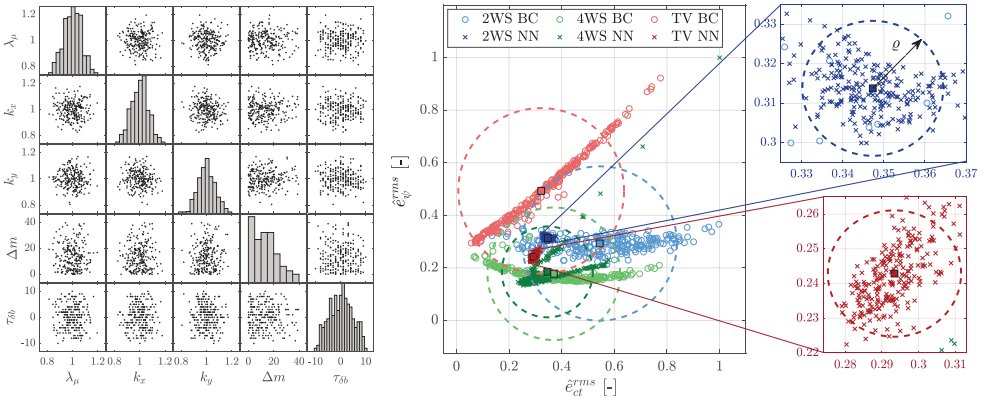


Figure 11. Monte Carlo simulation results: (a) parameter distributions; and (b) resulting normalised $\hat{e}_{ct}^{rms}$ and $\hat{e}_{\psi}^{rms}$ cloud points.

Table 4. Monte Carlo simulation results in terms of $\xi_{\hat{e}_{ct}^{rms}}$, $\xi_{\hat{e}_{\psi}^{rms}}$ and ϱ for the BCs and NNs applied to the considered actuation configurations.

Configuration	$\xi_{\hat{e}_{ct}^{rms}}$ **[-]**	$\xi_{\hat{e}_{\psi}^{rms}}$ **[-]**	ϱ **[-]**
BCs			
2WS	0.54	0.29	0.29
4WS	0.37	0.18	0.25
TV	0.32	0.49	0.32
NNs			
2WS	0.35	0.31	0.02
4WS	0.35	0.19	0.17
TV	0.29	0.24	0.02

the BCs, except for the 4WS NN, in which the radius is only slightly lower. Some BC samples may outperform the corresponding NNs, but this is due to the strong sensitivity of the BCs to parameter variations. In fact, in certain conditions, such sensitivity induces performance improvements, yet it highlights lack of robustness, since relatively small deviations from the nominal parameters can produce large performance changes.

- Variable-μ manoeuvres, with two objectives: (i) to evaluate how the controllers react to swift friction variations, especially reductions in the transition from straight-line to cornering, see the friction maps overlapped with the reference and actual trajectories in Figure 12(a); and (ii) to quantify the benefit of providing a varying friction factor input to the algorithms. Three TV configurations are tested, namely, in increasing order of complexity: (a) TV BC, which is used as term of reference; (b) TV NN, i.e. the same TV NN algorithm that was considered in the previous high-friction analyses; and (c) TV NN w λ_{μ}, including the additional friction factor input, see Section 4.1. Figure 12(b) highlights that both NNs outperform BC, i.e. they generate smoother control actions and prevent the vehicle from exhibiting the almost drifting behaviour and oscillatory

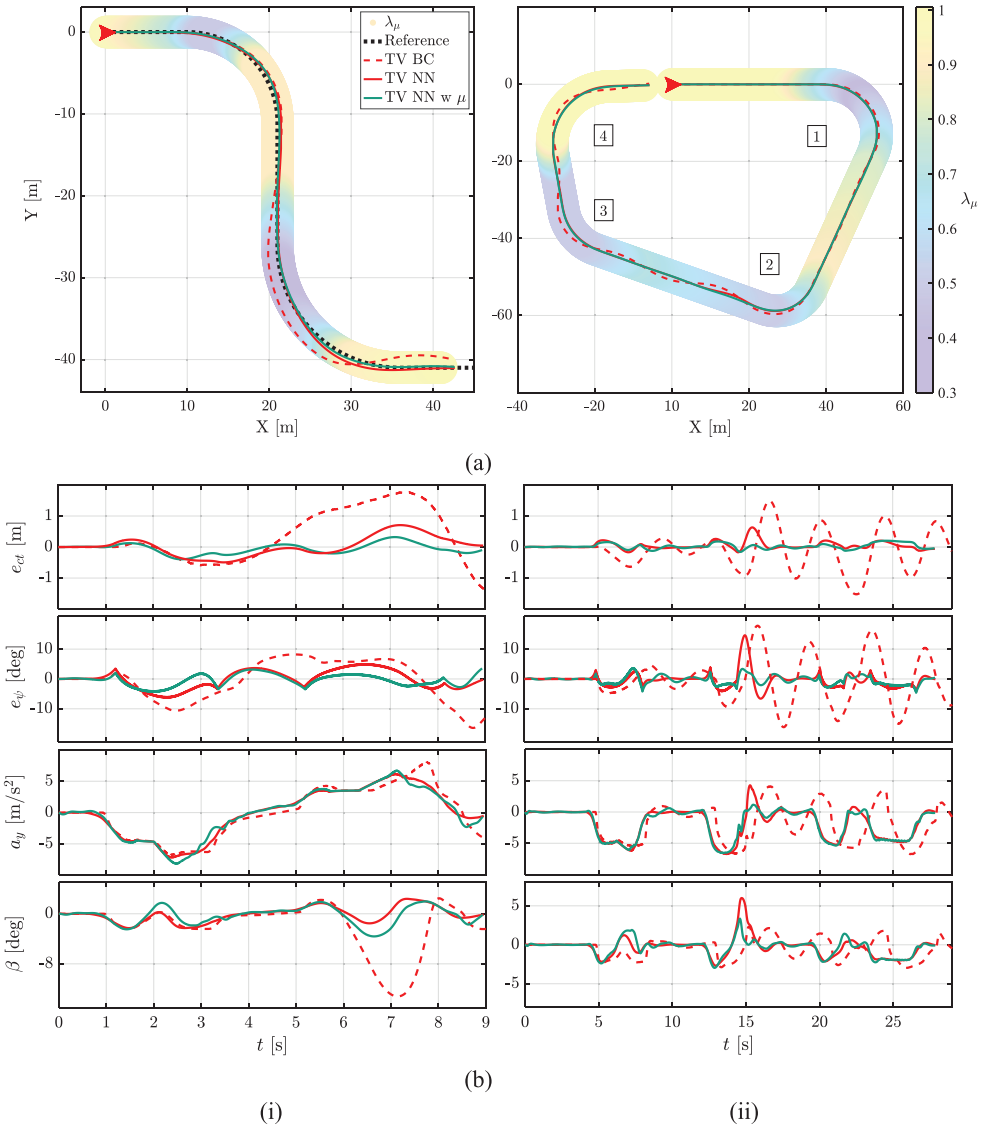


Figure 12. Simulated variable-μ manoeuvres: (i) S-shaped trajectory; and (ii) Track circuit, with plots of: (a) reference and actual trajectories; and (b) time profiles of e_{ct}, e_{ψ}, a_y, and β.

responses observed with BC, respectively in the second corner of the S-shaped trajectory and in the circuit, which make the BC performance unsatisfactory. TV NN w λ_μ achieves the lowest cross-track error in all conditions, especially in the lowest friction sections. Nonetheless, also the TV NN performance can be considered satisfactory. In fact, TV NN tends to select commands that maintain a sufficient stability margin, e.g. by progressively steering the vehicle back to the reference when the current tracking error is large, thanks to the inherently stability-oriented calibration of the NMPC-based expert.

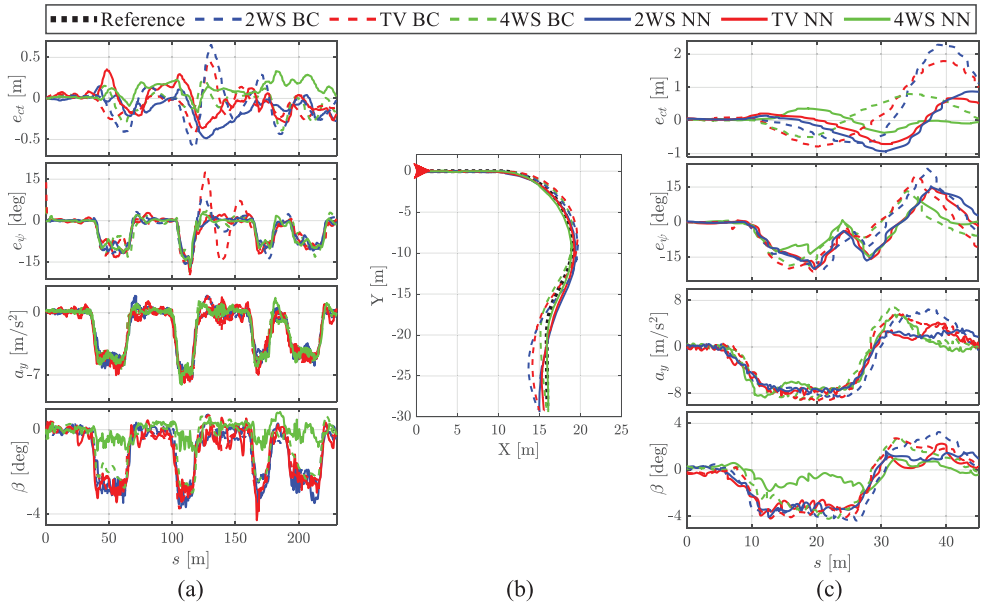


Figure 13. Experimental results: (a) time profiles of e_{ct}, e_ψ, a_y, and β along the high-friction Track circuit; (b) reference and actual trajectories along the High-a_y transient scenario; and (c) time profiles of e_{ct}, e_ψ, a_y, and β for the manoeuvre in (b).

Table 5. Experimental KPIs for the BCs and NNs applied to the considered actuation configurations.

Configuration	e_{ct}^{max} [m]	e_{ct}^{rms} [m]	e_ψ^{max} [deg]	e_ψ^{rms} [deg]	β^{max} [deg]	β^{rms} [deg]
Track circuit						
BCs						
2WS	0.66	0.23	18	6.1	3.7	1.5
4WS	0.40	0.12	16	5.5	2.8	1.1
TV	0.46	0.14	17	5.2	3.4	1.2
NNs						
2WS	0.49	0.17	16	5.4	3.7	1.6
4WS	0.33	0.13	19	5.4	1.1	0.4
TV	0.36	0.16	19	5.7	3.3	1.5
High-a_y transient scenario						
BCs						
2WS	2.29	0.82	23	9.2	4.4	2.2
4WS	0.80	0.35	19	7.0	4.2	1.8
TV	1.80	0.71	21	8.8	3.8	2.0
NNs						
2WS	0.94	0.39	20	8.2	4.1	1.8
4WS	0.37	0.15	15	5.6	1.9	0.7
TV	0.71	0.30	20	7.7	3.9	1.7

5.3. *Preliminary proof-of-concept experimental results*

The reported experimental PT tests – carried out at the Torino Aero Club proving ground – involved the EV operation along the high-friction Track circuit and the High-a_y transient scenario in Figure 7. The results are summarised in Figure 13 and Table 5.

In turns 1, 3 and 4 of the track, $|a_y|$ reaches values ranging from 4 to 5 m/s^2, see Figure 13(a), i.e. the vehicle dynamics remain in the linear region, and thus there are not

significant differences in terms of e_{ct} among the controllers and chassis actuation suites, which all achieve small errors. The situation changes in turn 2, with $|a_y|$ reaching ∼7 m/s^2. The involved vehicle dynamics become nonlinear, and the level of difficulty is significantly higher. In this case, the additional actuators and the more complex NN-based PT controllers bring clear benefits, e.g. see the e_{ct} profiles at $s = \sim 120$ m in Figure 13(a), where the NNs consistently manage to reduce the e_{ct} peaks by ∼20%. Moreover, the BCs tend to oscillate around the reference, e.g. see the e_{ct} profiles from $s = 100$–150 m, whereas the NNs progressively approach the RT. The β profile is similar among the controllers, with the exception of 4WS NN, which, although not receiving – differently from the corresponding NMPC – the sideslip angle information as an input, enables the desirable effect of nearly zero-sideslip angle operation.

In the second manoeuvre, i.e. the High-a_y transient scenario, see Figure 13(b,c) as well as Table 5, the NN superiority is more evident, i.e. all the tracking performance KPIs of the NN controllers are better than those for the corresponding BCs. This is due to the capability of the NNs to handle non-linearities, and their predictive nature enabled by the adoption of the look-ahead points, which can induce local performance sacrifices to obtain more relevant benefits later on.

In terms of e_{ct}^{max} and e_{ct}^{rms}, the results show: (i) ∼59 and ∼52% reductions from 2WS BC to 2WS NN; (ii) ∼54 and ∼57% reductions from 4WS BC to 4WS NN; and (iii) ∼61 and ∼58% reductions from TV BC to TV NN. On average, the e_ψ peaks are ∼15% lower for the NN implementations. While in the S-shaped trajectory the differences among the various actuation layouts were not particularly pronounced, in this more extreme scenario, the use of additional actuators significantly increases performance. In fact, e_{ct}^{max}- and e_{ct}^{rms}-wise, the experimental measurements highlight: (a) ∼65 and ∼57% reductions from 2WS BC to 4WS BC; (b) ∼61% reductions for the corresponding NNs; (c) ∼22 and ∼14% reductions from 2WS BC to TV BC; and (d) ∼24 and ∼23% reductions for the corresponding NNs.

6. Conclusions

The study presented an IL methodology for training centralised NN-based PT algorithms for EVs with multiple chassis actuators, using NMPCs as experts. The objective is to achieve computationally efficient non-task-specific controllers capable of high-performance operation across a wide range of scenarios, including limit handling conditions. The research led to NN-based PT algorithms that:

(a) Rely on a deliberately limited set of input variables; i.e. differently from NMPC implementations, the full vehicle state vector is not required, reducing the PT controller performance dependency on advanced sensor fusion systems.
(b) Implicitly handle realistic sensor characteristics, including significant measurement delays and low sampling frequencies without compromising tracking performance, thanks to their exposure to realistically emulated non-ideal sensor outputs in the training phase.
(c) Reduce the CPU time by 5 orders of magnitude with respect to the corresponding expert NMPC implementations, which enables the real-time operation of advanced algorithms with a high number of control actions.

(d) Provide comparable tracking performance to that of the advanced non-real-time-capable NMPC implementations, based on the considered set of simulations – carried out with an experimentally validated vehicle model – covering the whole range of achievable lateral accelerations.
(e) Achieved 54-to-61% and 52-to-58% reductions of the maximum and rms values of the cross-track error e_{ct} in the experimentally implemented transient scenario involving high lateral acceleration operation, with respect to the considered benchmarking controllers, across the range of available chassis actuators.
(f) Highlighted the benefit of the integrated control of multiple chassis actuators during limit handling operation, e.g. with $\sim$61 and $\sim$24% e_{ct}^{max} reductions brought by 4WS and TV compared with 2WS, in the same experimental scenario as in (e).
(g) Were comprehensively tested for robustness, retaining satisfactory performance for variations of key vehicle parameters and operating scenarios, despite the NNs remaining unaltered. In particular, the Monte Carlo analysis results showed $>$ 70% reduction of the dispersion circle radii of the NN-based PTs compared to the BCs, averaged among the actuator configurations.

Future steps will include: (i) more comprehensive experimental validations of the proposed algorithms, including the extension of the method to a wider operating speed range, as well as conditions involving the activation of NMPC constraints; (ii) the assessment of the methodology for applications including broader actuator sets, e.g. active/semi-active suspensions and active camber systems; and (iii) online adaptation algorithms of the NN-based PTs.

Acknowledgements

The authors would like to thank Giulio Spuri from Politecnico di Torino for the valuable discussions to improve the content of this paper.

Author contributions

CRediT: **Cecilia Formento:** Data curation, Formal analysis, Investigation, Software, Validation, Visualization, Writing – original draft; **Salvatore Ingraffia:** Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft; **Luca Dimauro:** Conceptualization, Data curation, Investigation, Methodology, Supervision, Validation, Writing – review & editing; **Mauro Velardocchia:** Conceptualization, Methodology, Project administration, Resources, Supervision, Writing – review & editing; **Aldo Sorniotti:** Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Writing – review & editing

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Horizon Europe Programme of the European Commission under grant agreement no. 101138110 (SmartCorners project).

ORCID

Cecilia Formento http://orcid.org/0009-0000-1970-5076
Luca Dimauro http://orcid.org/0000-0003-0874-682X
Mauro Velardocchia http://orcid.org/0000-0003-0757-1626
Aldo Sorniotti http://orcid.org/0000-0002-4848-058X

References

[1] Stanton NA, Salmon PM. Human error taxonomies applied to driving: A generic driver error taxonomy and its implications for intelligent transport systems. Saf Sci. 2009;47:227–237. doi:10.1016/j.ssci.2008.03.006
[2] Yasak MF, Heerwan PM, Aparow VR. Collision avoidance strategies in autonomous vehicles and on-ramp scenario: A review. Annu Rev Control. 2025;59:100986. doi:10.1016/j.arcontrol.2025.100986
[3] SAE International. (2021). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, SAE J3016™ APR2021 (J3016_202104).
[4] Sorniotti A, Barber P, De Pinto S. Path tracking for automated driving: A tutorial on control system formulations and ongoing research. In: Watzenig D, Horn M, editors. Automated driving. Cham: Springer; 2017. p. 71–140. doi:10.1007/978-3-319-31895-0_5
[5] Hoffmann GM, Tomlin CJ, Montemerlo M, et al. Autonomous automobile trajectory tracking for off-road driving: controller design, experimental validation and racing. American Control Conference. 2007, 2296–2301. doi:10.1109/ACC.2007.4282788
[6] Kapania NR, Gerdes JC. Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. Veh Syst Dyn. 2015;53(12):1687–1704. doi:10.1080/00423114.2015.1055279
[7] Zeng D, Pan S, Yu Y, et al. A comparative study on trajectory tracking control methods for automated vehicles. Sci Rep. 2025;15(1). doi:10.1038/s41598-025-01365-9
[8] Rokonuzzaman M, Mohajer N, Nahavandi S, et al. Review and performance evaluation of path tracking controllers of autonomous vehicles. IET Intel Transport Syst. 2021;15(5):646–670. doi:10.1049/itr2.12051
[9] Stano P, Montanaro U, Tavernini D, et al. Model predictive path tracking control for automated road vehicles: A review. Annu Rev Control. 2023;55:194–236. doi:10.1016/j.arcontrol.2022.11.001
[10] Stano P, Tavernini D, Montanaro U, et al. Enhanced active safety through integrated autonomous drifting and direct yaw moment control via nonlinear model predictive control. IEEE Transactions on Intelligent Vehicles. 2024;9(2):4172–4190. doi:10.1109/TIV.2023.3340992
[11] Lee J, Choi S. Nonlinear model predictive control for path tracking in high-speed corner entry situations. Int J Automot Technol. 2022;23:1373–1381. doi:10.1007/s12239-022-0120-0
[12] Xu F, Zhang X, Chen H, et al. Parallel nonlinear model predictive controller for real-time path tracking of autonomous vehicle. IEEE Trans Ind Electron. 2024;71(12):16503–16513. doi:10.1109/TIE.2024.3390738
[13] Goh JYM, Thompson M, Dallas J, et al. Beyond the stable handling limits: nonlinear model predictive control for highly transient autonomous drifting. Veh Syst Dyn. 2024;62(10):2590–2613. doi:10.1080/00423114.2023.2297799
[14] Acerbo FS, Van der Auweraer H, Duy Son T. Safe and computational efficient imitation learning for autonomous vehicle driving. In: 2020 American Control Conference (ACC). Denver, CO, USA: 2020. p. 647–652. doi:10.23919/ACC45564.2020.9147256
[15] Spielberg NA, Brown M, Gerdes JC. Neural network model predictive motion control applied to automated driving with unknown friction. IEEE Trans Control Syst Technol. 2022;30(5):1934–1945. doi:10.1109/TCST.2021.3130225

[16] Chemin J, Hill A, Lucet E, et al. A study of reinforcement learning techniques for path tracking in autonomous vehicles. In: 2024 IEEE Intelligent Vehicles Symposium (IV). Jeju Island, Korea: 2024. p. 1442–1449. doi:10.1109/IV55156.2024.10588521
[17] Caponio C, Stano P, Carli R, et al. Modelling, positioning, and deep reinforcement learning path following control of scaled robotic vehicles: design and experimental validation. IEEE Trans Autom Sci Eng. 2025;22:9856–9871. doi:10.1109/TASE.2024.3513701
[18] Liu S, Müller S. Reliability of deep neural networks for an end-to-end imitation learning-based lane keeping. IEEE Trans Intell Transp Syst. 2023;24(12):13768–13786. doi:10.1109/TITS.2023.3299229
[19] Gong C, Lu C, Li Z, et al. Beyond imitation: A life-long policy learning framework for path tracking control of autonomous driving. IEEE Trans Veh Technol. 2024;73(7):9786–9799. doi:10.1109/TVT.2024.3382309
[20] Hawke J, Shen R, Gurau C, et al. Urban driving with conditional imitation learning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris, France: 2020. p. 251–257. doi:10.1109/ICRA40945.2020.9197408
[21] Liu Q, Xiong P, Zhu Q, et al. A DDPG-based path following control strategy for autonomous vehicles by integrated imitation learning and feedforward exploration. Chin J Mech Eng. 2025;38(1):174. doi:10.1186/s10033-025-01336-1
[22] Kim S, Han K, Choi SB. Imitation learning of nonlinear model predictive control for emergency collision avoidance. IEEE Trans Intell Veh. 2024;9(1):2908–2922. doi:10.1109/TIV.2023.3309962
[23] Pan Y, Cheng CA, Saigol K, et al. Imitation learning for agile autonomous driving. Int J Rob Res. 2020;39(2-3):286–302. doi:10.1177/0278364919880273
[24] Lee T, Kang Y. Performance analysis of deep neural network controller for autonomous driving learning from a nonlinear model predictive control method. Electronics (Basel). 2021;10(7):767. doi:10.3390/electronics10070767
[25] Xu S, Peng H, Tang Y. Preview path tracking control with delay compensation for autonomous vehicles. IEEE Trans Intell Transp Syst. 2021;22(5):2979–2989. doi:10.1109/TITS.2020.2978417
[26] Mazzilli V, De Pinto S, Pascali L, et al. Integrated chassis control: classification, analysis and future trends. Annu Rev Control. 2021;51:172–205. doi:10.1016/j.arcontrol.2021.01.005
[27] Li Y, Cai Y, Sun X, et al. Trajectory tracking of four-wheel driving and steering autonomous vehicle under extreme obstacle avoidance condition. Veh Syst Dyn. 2024;62(3):601–622. doi:10.1080/00423114.2023.2186249
[28] Lin C, Siampis E, Velenis E. Path-tracking control at the limits of handling of a prototype over-actuated autonomous vehicle. Veh Syst Dyn. 2025;63(5):853–875. doi:10.1080/00423114.2024.2361726
[29] Tavolo G, Stano P, Tavernini D, et al. Nonlinear model predictive control for enhanced path tracking and autonomous drifting through direct yaw moment control and rear-wheel-steering. In: 16th International Symposium on Advanced Vehicle Control. Lecture Notes in Mechanical Engineering. Cham: Springer; 2024. doi:10.1007/978-3-031-70392-8_120
[30] Le Mero L, Yi D, Dianati M, et al. A survey on imitation learning techniques for end-to-end autonomous vehicles. IEEE Trans Intell Transp Syst. 2022;23(9):14128–14147. doi:10.1109/TITS.2022.3144867
[31] Verschueren R, Frison G, Kouzoupis D, et al. Acados – a modular open-source framework for fast embedded optimal control. Math Program Comput. 2022;14(1):147–183. doi:10.1007/s12532-021-00208-8
[32] Parra A, Tavernini D, Gruber P, et al. On nonlinear model predictive control for energy-efficient torque-vectoring. IEEE Trans Veh Technol. 2021;70(1):173–188. doi:10.1109/TVT.2020.3022022
[33] Bishop CM. Pattern recognition and machine learning. New York: Springer Nature; 2006.
[34] Cheng Y. Backpropagation for fully connected cascade networks. Neural Process Lett. 2017;46(1):293–311. doi:10.1007/s11063-017-9588-4