2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Coupled Longitudinal and Lateral Control of a Vehicle using Deep Learning*

Guillaume Devineau[1,⋆], Philip Polack[1,⋆], Florent Altché[1,2], and Fabien Moutarde[1]

Abstract— **This paper explores the capability of deep neural networks to capture key characteristics of vehicle dynamics, and their ability to perform coupled longitudinal and lateral control of a vehicle. To this extent, two different artificial neural networks are trained to compute vehicle controls corresponding to a reference trajectory, using a dataset based on high-fidelity simulations of vehicle dynamics. In this study, control inputs are chosen as the steering angle of the front wheels, and the applied torque on each wheel. The performance of both models, namely a Multi-Layer Perceptron (MLP) and a Convolutional Neural Network (CNN), is evaluated based on their ability to drive the vehicle on a challenging test track, shifting between long straight lines and tight curves. A comparison to conventional decoupled controllers on the same track is also provided.**

I. INTRODUCTION

The recent development of deep learning has led to dramatic progress in multiple research fields, and this technique has naturally found applications in autonomous vehicles. The use of deep learning to perform perceptive tasks such as image segmentation has been widely researched in the last few years, and highly efficient neural network architectures are now available for such tasks. More recently, several teams have proposed taking deep learning a step further, by training so-called "end-to-end" algorithms to directly output vehicle controls from raw sensor data (see, in particular, the seminal work in [1]).

Although end-to-end driving is highly appealing, as it removes the need to design motion planning and control algorithms by hand, handing the safety of the car occupants to a software operating as a black box seems problematic. A possible workaround to this downside is to use "forensics" techniques that can, to a certain extent, help understand the behavior of deep neural networks [2].

We choose a different approach consisting in breaking down complexity by training simpler, mono-task neural networks to solve specific problems arising in autonomous driving; we argue that the reduced complexity of individual tasks allows much easier testing and validation.

In this article, we focus on the problem of controlling a car-like vehicle in highly dynamic situations, for instance to perform evasive maneuvers in face of an obstacle. A particular challenge in such scenarios is the important coupling between longitudinal and lateral dynamics when nearing the vehicle's handling limits, which requires highly detailed models to properly take into account [3]. However, precisely modeling this coupling involves complex non-linear relations between state variables, and using the resulting model is usually too costly for real-time applications. For this reason, most references in the field of motion planning mainly focus on simpler models, such as point-mass or kinematic bicycle (single track), which are constrained to avoid highly coupled dynamics [4]. Similarly, research on automotive control usually treats the longitudinal and lateral dynamics separately in order to simplify the problem [5].

Although these simplifications can yield good results in standard on-road driving situations, they may be problematic for vehicle safety when driving near its handling limits, for instance at high speed or on slippery roads. To handle such situations, some authors have proposed using Model Predictive Control (MPC) with a simplified, coupled dynamic model [6] which is limited to extremely short time horizons (a few dozen milliseconds) to allow real-time computation. Other authors have proposed to model the coupling between longitudinal and lateral motions using the concept of "friction circle" [7], which allows precisely stabilizing a vehicle in circular drifts [8]. However, the transition towards the stabilized drifting phase – which is critical in the ability, *e.g.*, to perform evasive maneuvers – remains problematic with this framework.

In this article, we propose to use deep neural networks to implicitly model highly coupled vehicular dynamics, and perform low-level control in real-time. In order to do so, we train a deep neural network to output low-level controls (wheels torque and steering angle) corresponding to a given initial vehicle state and target trajectory. Compared to classical MPC frameworks which require integrating dynamic equations on-line, this approach allows to perform this task off-line and use only simple mathematical operations on-line, leading to much faster computations.

Several authors have already proposed a divide-and-conquer approach by using machine learning on specific subtasks instead of performing end-to-end computations, and in particular on the case of motion planning and control. For instance, reference [9] used a Convolutional Neural Network (CNN) to generate a cost function from input images, which is then used inside an MPC framework for high-speed autonomous driving; however, this approach has the same limitations as model predictive control. Other approaches,

*This work was supported by the international Chair MINES ParisTech - Peugeot-Citroën - Safran - Valeo on ground vehicle automation

⋆ Guillaume Devineau and Philip Polack contributed equally to this work.

[1]Guillaume Devineau, Philip Polack, Florent Altché and Fabien Moutarde are with Center for Robotics, MINES ParisTech, PSL Research University, 60 boulevard Saint-Michel, 75006 Paris, France {guillaume.devineau, philip.polack, florent.altche, fabien.moutarde}@mines-paristech.fr

[2]Florent Altché is also with École des Ponts ParisTech, Cité Descartes, 6-8 Av Blaise Pascal, 77455 Champs-sur-Marne, France.

such as [10], used reinforcement learning to output steering controls for a vehicle, but were limited to low-speed applications. Reference [11] used a Rectified Linear Unit (ReLU) network model to identify the dynamics of a helicopter in order to predict its future accelerations, but this model has not been used for control.

Closer to our work, reference [12] trained neural networks integrating a priori knowledge of the bicycle model for decoupled longitudinal and lateral control of a vehicle; in [13], authors used supervised learning to generate lateral controls for truck and integrated a control barrier function to ensure the safety of the system. Reference [14] coupled a standard control and an adaptive neural network to compensate for unknown perturbations in order to perform trajectory tracking for autonomous underwater vehicle. To the best of our knowledge, deep neural networks have not been used in the literature for the coupled control of wheeled vehicles.

The rest of this article is organized as follows: Section II presents the vehicle model used to generate the training dataset and to simulate the vehicle dynamics on a test track. Section III introduces two artificial neural networks architectures used to generate the control signals for a given target trajectory, and describes the training procedure used in this article. Section IV compares the performance of these two networks, using simulation on a challenging test track. A comparison to conventional decoupled controllers is also provided. Finally, Section V concludes this study.

II. THE 9 DoF VEHICLE MODEL

In this section, we present the 9 Degrees of Freedom (9 DoF) vehicle model which is used both to generate the training and testing dataset, and as a simulation model to evaluate the performance of the deep-learning-based controllers.

The Degrees of Freedom comprise 3 DoF for the vehicle's motion in a plane $(V_x, V_y, \dot{\psi})$, 2 DoF for the carbody's rotation $(\dot{\theta}, \dot{\phi})$ and 4 DoF for the rotational speed of each wheel $(\omega_{fl}, \omega_{fr}, \omega_{rl}, \omega_{rr})$. The model takes into account both the coupling of longitudinal and lateral slips and the load transfer between tires. The control inputs of the model are the torques T_{ω_i} applied at each wheel i and the steering angle δ of the front wheel. The low-level dynamics of the engine and brakes are not considered here. The notations are given in Table I and illustrated in Figure 1.

Remark: the subscript $i = 1..4$ refers respectively to the front left (fl), front right (fr), rear left (rl) and rear right (rr) wheels.

Several assumptions were made for the model:

- Only the front wheels are steerable.
- The roll and pitch rotations happen around the center of gravity.
- The aerodynamic force is applied at the height of the center of gravity. Therefore, it does not involve any moment on the vehicle.
- The slope and road-bank angle of the road are not taken into account.

TABLE I: Notations

X, Y	Position of the vehicle in the ground frame
θ, ϕ, ψ	Roll, pitch and yaw angles of the carbody
V_x, V_y	Longitudinal and lateral speed of the vehicle in its inertial frame
M_T	Total mass of the vehicle
I_x, I_y, I_z	Inertia of the vehicle around its roll, pitch and yaw axis
I_{r_i}	Inertia of the wheel i
T_{ω_i}	Total torque applied to the wheel i
F_{xp_i}, F_{yp_i}	Longitudinal and lateral tire forces generated by the road on the wheel i expressed in the tire frame
F_{x_i}, F_{y_i}	Longitudinal and lateral tire forces generated by the road on the wheel i expressed in the vehicle frame (x, y)
F_{z_i}	Normal reaction forces on wheel i
l_f, l_r	Distance between the front (resp. rear) axle and the center of gravity
l_w	Half-track of the vehicle
h	Height of the center of gravity
r_{eff}	Effective radius of the wheel
ω_i	Angular velocity of the wheel i
V_{xp_i}	Longitudinal speed of the center of rotation of wheel i expressed in the tire frame

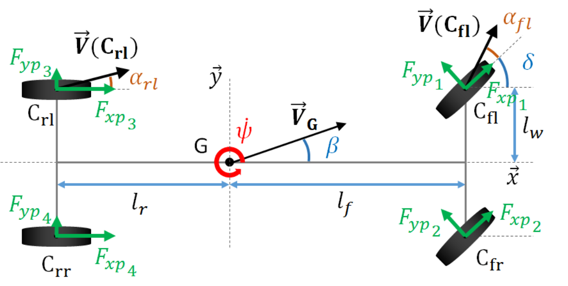

Fig. 1: Vehicle model and notations.

A. Vehicle dynamics

Equations (1a-1e) give the expression of the vehicle dynamics:

$$M_T \dot{V}_x = M_T \dot{\psi} V_y + \sum_{i=1}^{4} F_{x_i} - F_{aero} \tag{1a}$$

$$M_T \dot{V}_y = -M_T \dot{\psi} V_x + \sum_{i=1}^{4} F_{y_i} \tag{1b}$$

$$I_x \ddot{\theta} = l_w (F_{z_1} + F_{z_3} - F_{z_2} - F_{z_4}) + h \sum_{i=1}^{4} F_{y_i} \tag{1c}$$

$$I_y \ddot{\phi} = l_r (F_{z_3} + F_{z_4}) - l_f (F_{z_1} + F_{z_2}) - h \sum_{i=1}^{4} F_{x_i} \tag{1d}$$

$$\begin{aligned} I_z \ddot{\psi} &= l_f (F_{y_1} + F_{y_2}) - l_r (F_{y_3} + F_{y_4}) \\ &+ l_w (F_{x_2} + F_{x_4} - F_{x_1} - F_{x_3}) \end{aligned} \tag{1e}$$

F_{x_i} and F_{y_i} denote respectively the longitudinal and the lateral tire forces expressed in the vehicle frame; $F_{aero} = \frac{1}{2} \rho_{air} C_x S V_x^2$ denote the aerodynamic drag forces with ρ_{air} the mass density of air, C_x the aerodynamic drag coefficient and S the frontal area of the vehicle; F_{z_i} denote the damped mass/spring forces depending on the suspension travel ζ_i due to the roll θ and pitch ϕ angles according to Equation (1f). The parameters k_s and d_s are respectively the stiffness and the damping coefficients of the suspensions.

$$F_{z_i} = -k_s \zeta_i(\theta,\phi) - d_s \dot{\zeta}_i(\theta,\phi) \tag{1f}$$

The position (X,Y) of the vehicle in the ground frame can then be derived using Equations (1g) and (1h).

$$\dot{X} = V_x \cos\psi - V_y \sin\psi \tag{1g}$$
$$\dot{Y} = V_x \sin\psi + V_y \cos\psi \tag{1h}$$

B. Wheel dynamics

The dynamics of each wheel $i = 1..4$ expressed in the pneumatic frame is given by Equation (2):

$$I_r \dot{\omega}_i = T_{\omega_i} - r_{eff} F_{xp_i} \tag{2}$$

C. Tire dynamics

The longitudinal force F_{xp_i} and the lateral force F_{yp_i} applied by the road on each tire i and expressed in the pneumatic frame are functions of the longitudinal slip ratio τ_{x_i}, the side-slip angle α_i, the normal reaction force F_{z_i} and the road friction coefficient μ:

$$F_{xp_i} = f_x(\tau_{x_i}, \alpha_i, F_{z_i}, \mu) \tag{3a}$$
$$F_{yp_i} = f_y(\alpha_i, \tau_{x_i}, F_{z_i}, \mu) \tag{3b}$$

The longitudinal slip ratio of the wheel i is defined as following:

$$\tau_{x_i} = \begin{cases} \frac{r_{eff}\omega_i - V_{xpi}}{r_{eff}|\omega_i|} & \text{if } r_{eff}\omega_i \geq V_{xp_i} \text{ (Traction phase)} \\ \frac{r_{eff}\omega_i - V_{xpi}}{|V_{xpi}|} & \text{if } r_{eff}\omega_i < V_{xpi} \text{ (Braking phase)} \end{cases} \tag{4}$$

The lateral slip-angle α_i of tire i is the angle between the direction given by the orientation of the wheel and the direction of the velocity of the wheel (see Figure 1):

$$\alpha_f = \delta - \text{atan}\left(\frac{V_y + l_f \dot{\psi}}{V_x \pm l_w \dot{\psi}}\right);\ \alpha_r = -\text{atan}\left(\frac{V_y - l_r \dot{\psi}}{V_x \pm l_w \dot{\psi}}\right) \tag{5}$$

In order to model the functions f_x and f_y, we used the combined slip tire model presented by Pacejka in [15] (cf. Equations (4.E1) to (4.E67)) which takes into account the interaction between the longitudinal and lateral slips on the force generation. Therefore, the friction circle due to the laws of friction (see Equation (6)) is respected. Finally, the impact of load transfer between tires is also taken into account through F_z.

$$||\vec{F}_{xp} + \vec{F}_{yp}|| \leq \mu ||\vec{F}_z|| \tag{6}$$

Lastly, the relationships between the tire forces expressed in the vehicle frame F_x and F_y and the ones expressed in the pneumatic frame F_{xp} and F_{yp} are given in Equation (7):

$$F_{x_i} = (F_{xp_i}\cos\delta_i - F_{yp_i}\sin\delta_i)\cos\phi - F_{z_i}\sin\phi \tag{7a}$$
$$\begin{aligned} F_{y_i} &= (F_{xp_i}\cos\delta_i - F_{yp_i}\sin\delta_i)\sin\theta\sin\phi \\ &+ (F_{yp_i}\cos\delta_i + F_{xp_i}\sin\delta_i)\cos\theta + F_{z_i}\sin\theta\cos\phi \end{aligned} \tag{7b}$$

More details on vehicle dynamics can be found in [3] and [16].

III. DEEP LEARNING MODELS

We propose two different artificial neural network architectures to learn the inverse dynamics of a vehicle, in particular the coupled longitudinal and lateral dynamics. An artificial neural network is a network of simple functions called neurons. Each neuron computes an internal state (activation) depending on the input it receives and a set of trainable parameters, and returns an output depending on the input and the activation. Most neural networks are organized into groups of units called layers and arranged in a tree-like structure, where the output of a layer is used as input for the following one. The training of the neural network consists in finding the set of parameters (weights and biases) minimizing the error (or *loss*) between predicted and actual values on a training dataset. In this paper, this training dataset is computed using the 9 DoF vehicle model presented in Section II.

A. Dataset

The dataset generated by the 9DoF vehicle model has a total of 43241 instances: it is divided into a train set of 28539 instances and a test set of 14702 instances. The following procedure was used to generate each instance:

First, a control u to apply is generated randomly, as well as an initial state $\xi^{(0)}$ of the vehicle. More precisely, the vehicle is chosen to be either in an acceleration phase or in a deceleration phase with equiprobability. In the first case, the torques at the front wheels T_{ω_1} and T_{ω_2} are set equal to each other and drawn from a uniform distribution between 0Nm and 750Nm, while the torques at the rear wheels T_{ω_3} and T_{ω_4} are set equal to zero (the vehicle is assumed to be a front-wheel drive one). In the second case, the torques of each wheel are set equal to each other and drawn from a uniform distribution between -1250Nm and 0Nm. In both cases, the steering angle δ is drawn from a uniform distribution between -0.5 and $+0.5$rad. The initial state $\xi^{(0)}$ is composed of the initial position $(X^{(0)}, Y^{(0)})$ of the vehicle in the ground frame, the longitudinal and lateral velocities $V_x^{(0)}$ and $V_y^{(0)}$, the roll, pitch and yaw angles and their derivatives, and the rotational speed $\omega_i^{(0)}$ of the each wheels. The initial longitudinal speed $V_x^{(0)}$ is drawn from a uniform distribution between 5 and 40m.s^{-1}; the initial lateral speed $V_y^{(0)}$ is drawn from a uniform distribution whose parameters depend of $V_x^{(0)}$; the rotational speed $\omega_i^{(0)}$ is chosen such that the longitudinal slip ratio is zero. All the other initial states are set to zero.

Secondly, the 9 DoF vehicle model is run for 3s, starting from the initial state $\xi^{(0)}$ and keeping the control u constant during the whole simulation. The resulting trajectories are downsampled to 301 timesteps, corresponding to a sampling time of 10ms.

Consequently, each instance of the dataset consists in: an initial state $\xi^{(0)}$ of the vehicle, a control $(T_{\omega_1}, T_{\omega_2}, T_{\omega_3}, T_{\omega_4}, \delta)$ kept constant over time, and the associated trajectory obtained $(X^{(0)}, Y^{(0)}), \ldots, (X^{(300)}, Y^{(300)})$.

The dataset generation method is summarized in Algorithm 1.

Algorithm 1 Dataset Generation

```
 1: function GENERATE INSTANCE
 2:     is_accelerating $\sim \mathscr{B}(0,1)$                ▷ Coin flipping
 3:     if is_accelerating $= 1$ then
 4:         $u_1 \sim \mathscr{U}(0,750)$                ▷ uniform; in N.m
 5:         $\delta \sim \mathscr{U}(-0.5,+0.5)$              ▷ uniform; in rad
 6:         $u \leftarrow [u_1, u_1, 0, 0, \delta]$
 7:     else if is_accelerating $= 0$ then
 8:         $u_1 \sim \mathscr{U}(-1250,0)$              ▷ uniform; in N.m
 9:         $\delta \sim \mathscr{U}(-0.5,+0.5)$              ▷ uniform; in rad
10:         $u \leftarrow [u_1, u_1, u_1, u_1, \delta]$
11:     end if
12:     $V_x^{(0)} \sim \mathscr{U}(5,40)$                ▷ uniform; in m.s$^{-1}$
13:     $V_y^{(0)} \sim \mathscr{U}(a,b)$                 ▷ uniform; in m.s$^{-1}$
14:     where $a = \max\left(-1, -\frac{V_x^{(0)}}{3}\right)$
15:     and $b = \min\left(+1, +\frac{V_x^{(0)}}{3}\right)$
16:     trajectory $\leftarrow 9DoF(\xi^{(0)}, u, T_{sim} = 3s)\big|_{(X,Y)}$
17:     save $(\xi^{(0)}, u,$ trajectory$)$
18: end function
19: function GENERATE DATASET($n = 43241$)
20:     for $i \leftarrow 1 \ldots n$ do GENERATE INSTANCE()
21:     end for
22: end function
```

B. Model 1: Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP), or multi-layer feedforward neural network, is a neural network f whose equations are:

$$\mathbf{h}^{(0)} = \mathbf{x} \quad (8a)$$

$$\mathbf{h}^{(k)} = \sigma^{(k)}(\mathbf{W}^{(k)\top}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}), \text{ for } k = 1..L \quad (8b)$$

where $\mathbf{x}$ denotes the input vector, $\mathbf{h}^{(k)}$ the output of layer $k \in [\![1, L]\!]$, $L \in \mathbb{N}^*$ the number of layers of the MLP and $\sigma^{(k)}$ denotes the k-th activation function. $\mathbf{h}^{(L)} = f(\mathbf{x})$ denotes the output vector of the neural network.

The MLP, presented in Figure 2, is used to predict the constant control $(T_{\omega_1}, T_{\omega_2}, T_{\omega_3}, T_{\omega_4}, \delta)$ to apply given an initial state $\xi^{(0)}$ and a desired trajectory $(X^{(0)}, Y^{(0)}), \ldots, (X^{(300)}, Y^{(300)})$. It is trained on the dataset presented in subection III-A. It comprises $L = 5$ layers, respectively containing 32, 32, 128, 32 and 128 neurons. All the activations functions of the network are rectified linear units (ReLU): $\sigma(x) = \max(0, x)$. The loss function used, as well as weights initialization or regularization are discussed in the section III-D, as they are common for the two neural networks proposed. We performed a grid search to choose the sizes of the layers among $3^5 = 243$ possibilities by allowing each layer to have a size of either 32, 64, or 128 neurons, training the corresponding MLP for 200 epochs and evaluating its performance on the test dataset.

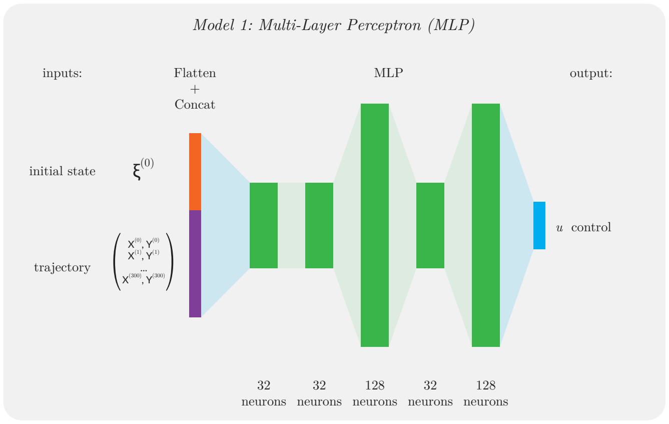
Model 1: Multi-Layer Perceptron (MLP)
inputs:
Flatten + Concat
MLP
output:
initial state
$\xi^{(0)}$
trajectory
$\begin{pmatrix} X^{(0)}, Y^{(0)} \\ X^{(1)}, Y^{(1)} \\ \ldots \\ X^{(300)}, Y^{(300)} \end{pmatrix}$
u control
32 neurons
32 neurons
128 neurons
32 neurons
128 neurons

Fig. 2: Multi-Layer Perceptron

C. Model 2: Convolutional Neural Network

Convolutional Neural Networks (CNN) are neural networks that use convolution in place of general matrix multiplication in at least one of their layers. A traditional CNN model almost always involves a sequence of convolution and pooling layers. CNNs have a proven history of being successful for processing data that has a known grid-like topology. For instance, numerous authors make use of CNNs for classification [17], or semantic segmentation [18] purposes.

We propose to use convolutions to pre-process the vehicle trajectory before feeding it to the MLP, as illustrated in Figure 3. Trajectories are time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and thus are very good inputs to process with a CNN. We decided to process the X and Y coordinates separately. For each channel $\mathbf{x}$ (either X or Y), we construct the following CNN module, which is depicted in Figure 4:

$$\mathbf{h}^{(0)} = \mathbf{x} \quad (9a)$$

$$\mathbf{h}^{(k)} = \sigma^{(k)}(\pi^{(k)}(\mathbf{W}^{(k)} * \mathbf{h}^{(k-1)} + \mathbf{b}^{(k)})), \text{ for } k = 1..L' \quad (9b)$$

where $\mathbf{h}^{(L')}$ is the output of the CNN module, $L' \in \mathbb{N}^*$ the number of layers, $\sigma^{(k)}$ the k-th activation function and $\pi^{(k)}$ the k-th pooling function.

The parameters of the CNN module are $L' = 3$, with a convolution kernel size of 3 for all convolutions. The activation functions are all ReLU and the pooling functions are all average-pooling of size 2. The first two convolutions have 4 feature maps while the last convolution has only 1 feature map.

As the longitudinal and lateral dynamics are quite different, distinct sets of weights are used for the X and Y convolutions. After processing the X and Y 1D-trajectories by their dedicated CNN module, their output are concatenated. This new output is then fed to the former MLP whose characteristics remain the same except from the dimension of its input. The whole model shown in Figure 3 is designated as the “CNN model” in the rest of this work.

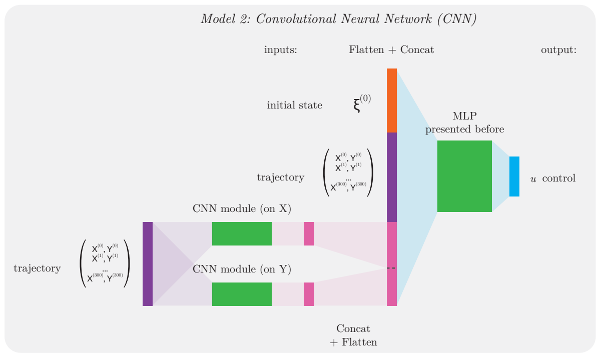

Fig. 3: Convolutional Neural Network

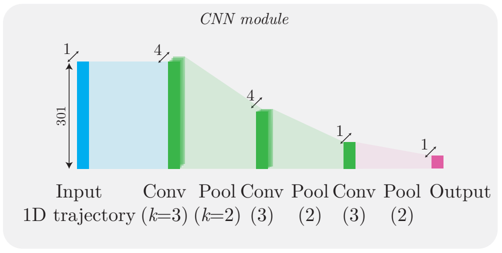

Fig. 4: CNN Module

D. Training procedure

The training procedure is the same for the two neural networks:

1) Weights Initialization & Batching: Each training batch is composed of 32 instances of the dataset. The Xavier initialization [19] (also known as GLOROT uniform initialization) is used to set the initial random weights for all the weights of our model.

2) Loss function, Regularization & Optimizer: The objective of the training is to reduce the mean square error (MSE) between the controls predicted u^{pred} by the neural network and the ones u^{real} that were really applied to obtain the given trajectory. The neural network is trained in order to minimize the loss function L defined by Equation (10) on the train dataset, before evaluation on the test dataset.

$$L = \gamma L_\delta + (1-\gamma) L_T + L_{reg} \tag{10}$$

where

$$L_\delta(\delta^{real}, \delta^{pred}) = \frac{1}{0.5} MSE(\delta^{real}, \delta^{pred}) \tag{11a}$$

$$L_T(T_{\omega_i}^{real}, T_{\omega_i}^{pred}) = \frac{1}{4 \times 2000} \sum_{i=1}^{4} MSE(T_{\omega_i}^{real}, T_{\omega_i}^{pred}) \tag{11b}$$

$$L_{reg}(W) = \gamma_{reg} ||W||_2^2 \tag{11c}$$

The scaling factors $1/0.5$ and $1/(4 \times 2000)$ were chosen in order to normalize the steering and the torques. The parameter $\gamma = 0.99$ was chosen in order to prioritize the lateral dynamics over the longitudinal one. Equation (11c) is an L2 regularization of our model, where W is the vector containing all the weights of the network. We set $\gamma_{reg} = 10^{-5}$.

To train our model, we used the Adam optimization algorithm [20]. It calculates an exponential moving average of the gradient and the squared gradient. For the decay rates of the moving averages, we used the parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$. The values of other parameters were $\alpha = 10^{-3}$ for the learning rate, and $\varepsilon = 10^{-8}$ to avoid singular values.

IV. RESULTS

In order to compare their ability to learn the vehicle dynamics, the two different artificial neural networks are used as "controllers"[1]. The reference track, presented in Figure 5, comprises both long straight lines and narrow curves. The reference speed is set to V_{ref} = 10m/s on the whole track.

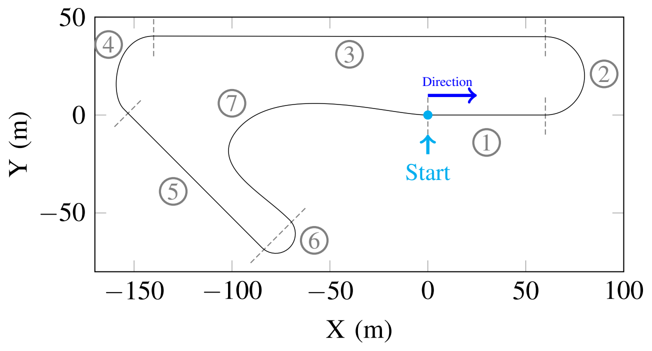

Fig. 5: Top view of the test track; numbers 1 to 7 refers to different road sections delimited by dashed lines in order to facilitate the matching with Figures 9 to 13.

A. Generating the control commands

In order to compute the control commands to be applied to the vehicle, the artificial neural network needs to know the trajectory the vehicle has to follow in the next 3s, as in the train dataset. One problem that arises is that it has only learned to follow trajectories starting from its actual position such as in Figure 6. However, in practice, the vehicle is almost never exactly on the reference path. Therefore, a path section starting from the actual position of the vehicle and bringing it back to the reference path is generated: for that purpose, cubic Bezier curves were chosen as illustrated in Figure 7. Thus, at each iteration, (i) a Bezier curve with length 3s is computed to link the actual position of the vehicle to the reference trajectory; (ii) a query comprising the previously computed Bezier curve is sent to the artificial neural network; (iii) the artificial neural network returns the torques at each wheel and the front steering angle to apply until the next control commands are obtained. The computation sequence is run every 300ms, even though the query takes less than 2ms.

[1]Properly speaking, they are not real controllers as they to not learn how to reject disturbances and modeling errors.

B. Comparison of the models

The results obtained for the MLP and the CNN models are displayed respectively in blue and in red in Figures 9 to 13. The resulting videos, obtained using the software PreScan [21], are available online[2]. Clearly, it appears that the results obtained using a CNN are better than a MLP. First, we observe that the control commands are smoother in curves with the CNN. There are steep steering (see Figure 9) and front torques (see Figure 10) variations for the MLP around $s = 360$m in road sections n°4 and around $s = 480$m in road sections n°6. In the latter case, the steering angle reaches its saturation value $+0.5$rad and the wheel torques change suddently from 1000Nm to -1000Nm and vice-versa, which is impossible in practice. On the contrary, the control signals of the CNN model remains always smooth and within a reasonable range of values. Secondly, both the longitudinal and lateral errors are smaller for the CNN than the MLP as shown respectively in Table II and III.

TABLE II: Comparison of the longitudinal performances of the MLP and CNN controllers (in m/s).

model	RMS	average	std. dev.	max
MLP	0.76	-0.29	0.70	-4.94
CNN	0.60	-0.39	0.46	-2.33

TABLE III: Comparison of the lateral performances of the MLP and CNN controllers (in m).

model	RMS	average	std. dev.	max
MLP	0.61	0.003	0.61	3.26
CNN	0.43	0.014	0.43	1.7

However, unlike classic controllers, stability cannot be ensured for these “controllers” as they are black boxes. In particular, for the CNN, we observe a lateral static error in straight lines. This static error is caused in fact by the Bezier curves which do not converge fast enough to the reference track on straight lines as only the first 300ms are really followed by the CNN model (see Figure 8). Moreover, Figure 9 shows that the steering angle applied during straight lines is the same for MLP and CNN.

C. Coupling between longitudinal and lateral dynamics

The speed limit a kinematic bicycle model can reach in a curve of radius R is given by Equation (12) where $\mu = 1$ is the road friction coefficient and g the gravity constant [4]. This corresponds to 9.9m/s ($R = 20$m) in road section n°2 and 7.0m/s ($R = 10$m) in road section n°6. As the reference speed is set to 10m/s throughout the track, conventional decoupled longitudinal and lateral controllers (based on a kinematic bicycle model) will not perform well in road section n°6.

$$V_{kbm_{lim}} = \sqrt{0.5\mu g R} \quad (12)$$

On the contrary, both models (especially the CNN) are able to pass this road section, showing the ability of artificial neural networks to handle coupled longitudinal and lateral dynamics. More precisely, we observe in Figure 12 that the speed is reduced in section n°6 because the artificial neural networks deliberately brake (see Figure 10 and 11), even though the speed of the vehicle is below the reference speed. This is due to the loss function used during training and given by Equation (10) that penalizes more steering angle errors than torque errors. Hence, the models prioritize the lateral over the longitudinal dynamics.

Therefore, such “controllers” are particularly interesting for highly dynamic maneuvers such as emergency situations or aggressive driving where the longitudinal and lateral dynamics are strongly coupled. However, they should be used sparingly as they are only black boxes, or should at least be supervised by model-based systems. Moreover, for normal driving situations, conventional decoupled longitudinal and lateral controller should be preferred.

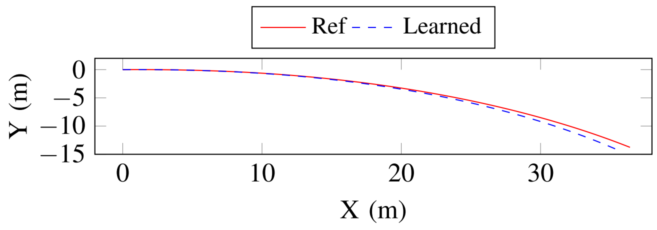

Fig. 6: Example of a training dataset instance: in red, the reference trajectory, in blue the one obtained from the control predicted by the CNN model.

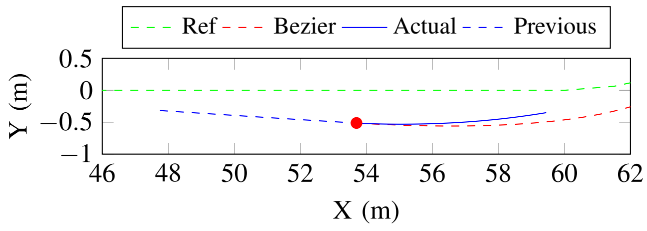

Fig. 7: Example of a Bezier curve (in red) joining the actual position of the vehicle (the red circle) to the reference trajectory (in green). The actual trajectory followed by the vehicle is shown in blue.

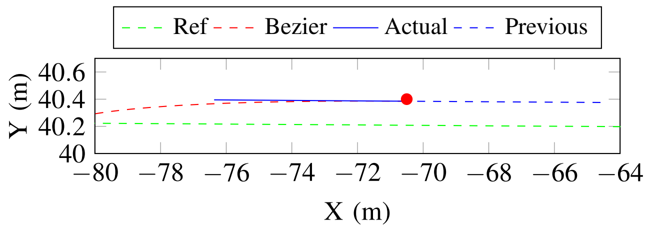

Fig. 8: Example of a Bezier curve on a straight line section of the reference trajectory. The lateral error is not corrected since the convergence of the Bezier curve to the reference trajectory is too slow.

[2]https://www.youtube.com/watch?v=yyWy1uavlXs

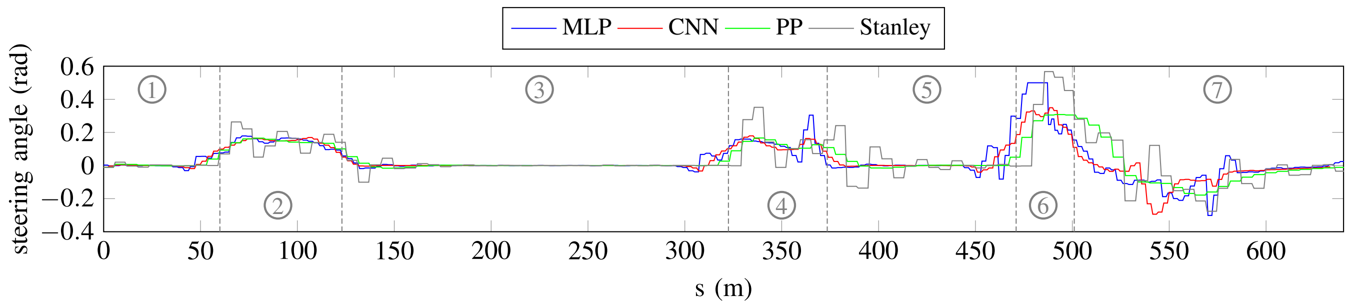

Fig. 9: Comparison of the steering command computed by the different controllers. The numbers 1 to 7 correspond to the different road sections presented in Figure 5.

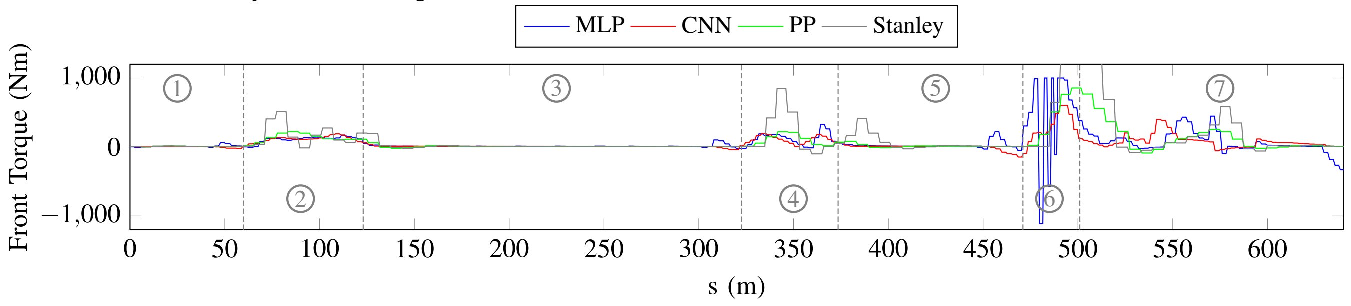

Fig. 10: Comparison of the torque applied at the front wheels computed by the different controllers.

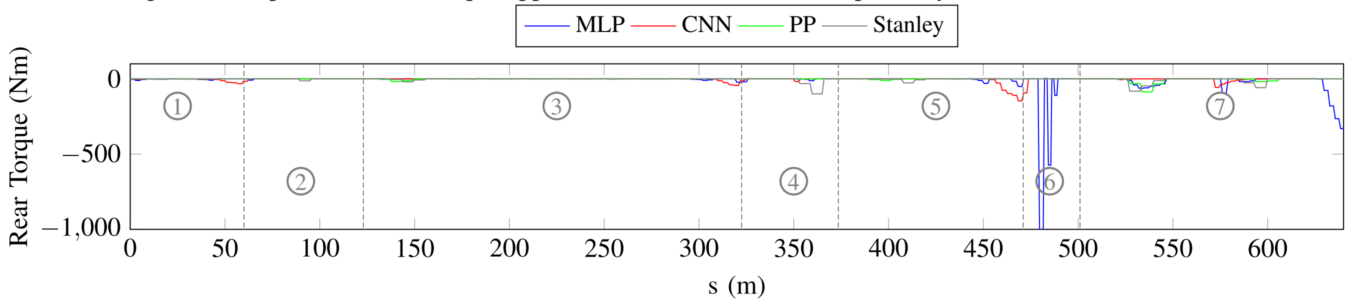

Fig. 11: Comparison of the torque applied at the rear wheels computed by the different controllers.

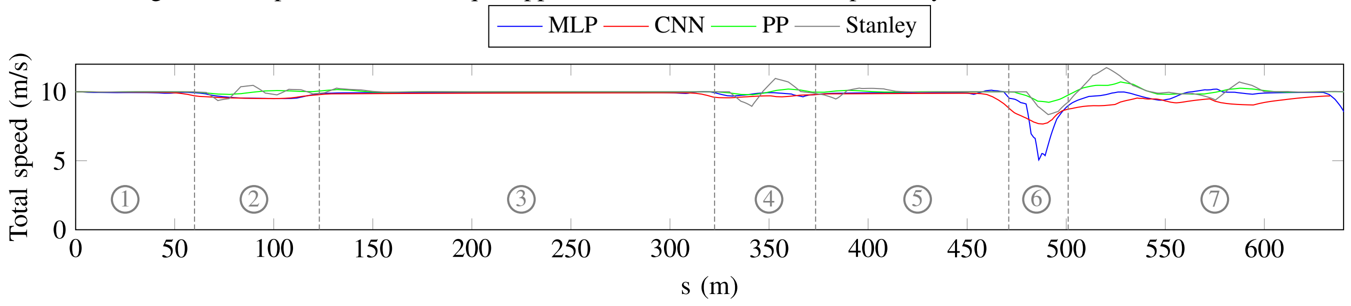

Fig. 12: Comparison of the total speed obtained with the different controllers.

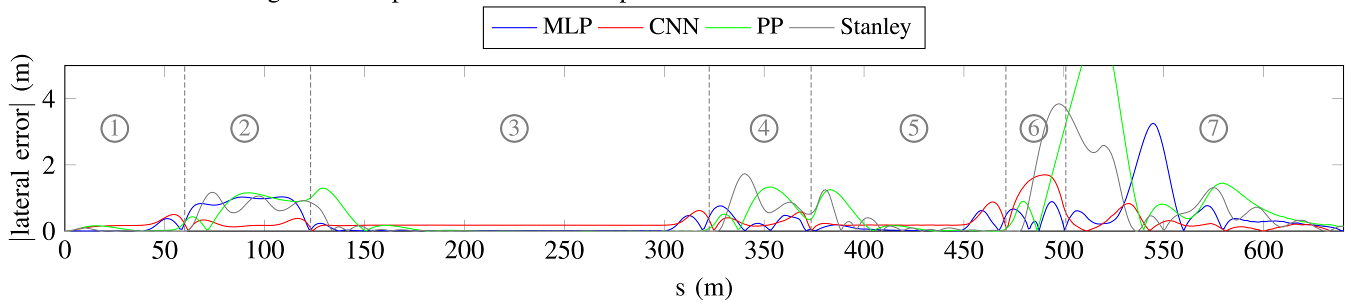

Fig. 13: Comparison of the absolute value of the lateral error obtained with the different controllers.

D. Comparison with decoupled controllers

Finally, the "controllers" obtained with the MLP and CNN models are compared with commonly used decoupled controllers: the lateral controller is either a pure-pursuit (PP) [22] or a Stanley [23] controller while in both cases, the longitudinal controller is ensured by a Proportional-Integral (PI) controller with gains $K_P = 600$ and $K_I = 10$. The gain for the front lateral error is 0.75 for the Stanley controller. The preview distance of the pure-pursuit controller is defined as a function of the total speed V_g at the center of gravity: $L_P = l_f + T_A V_g$ where $T_A = 1.5$s is the anticipation time. The results of the PP and the Stanley controllers are shown respectively in green and grey in Figures 9 to 13. Clearly, a decrease of performance can be observed when using these decoupled controllers in the challenging part of the track. In particular, the lateral error becomes huge in both cases during the sharp turn of road section n°6 while the CNN was able to perform reasonnably well.

V. CONCLUSIONS

This work presented some preliminary results on deep learning applied to trajectory tracking for autonomous vehicles. Two different approaches, namely a MLP and a CNN, were trained on a high-fidelity vehicle dynamics model in order to compute simultaneously the torque to apply on each wheel and the front steering angle from a given reference trajectory. It turns out that the CNN model provides better results, both in terms of accuracy and smoothness of the control commands. Moreover, compared to most of the existing controllers, it is able to handle situations with strongly coupled longitudinal and lateral dynamics in a very short time. However, the controller obtained is a black-box and should not be used in standalone.

The results proved the ability of deep learning algorithms to learn the vehicle dynamics characteristics. This opens a wide range of new possible applications of such techniques, for example for generating dynamically feasible trajectories. Future work will focus on (i) replacing the complex dynamics models by a learned off-line model in Model Predictive Control for motion planning, (ii) using Generative Adversarial Networks (GAN) to generate safe trajectories where the learned dynamics is used as constraint, and (iii) performing real-world experiments with our approach on a real car.

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv:1604*, pp. 1–9, apr 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[2] D. Castelvecchi, "Can we open the black box of ai?" *Nature News*, vol. 538, no. 7623, p. 20, 2016.

[3] T. D. Gillespie, "Vehicle dynamics," *Warren dale*, 1997.

[4] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "Guaranteeing Consistency in a Motion Planning and Control Architecture Using a Kinematic Bicycle Model," in *American Control Conference*, Milwaukee, WI, United-States, 2018 (accepted). [Online]. Available: https://arxiv.org/pdf/1804.08290.pdf

[5] A. Khodayari, A. Ghaffari, S. Ameli, and J. Flahatgar, "A historical review on lateral and longitudinal control of autonomous vehicle motions," *ICMET 2010 - 2010 International Conference on Mechanical and Electrical Technology, Proceedings*, no. Icmet, pp. 421–429, 2010.

[6] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 3, pp. 566–580, 2007.

[7] K. Kritayakirana and J. C. Gerdes, "Autonomous vehicle control at the limits of handling," *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 4, p. 271, 2012.

[8] J. Y. Goh and J. C. Gerdes, "Simultaneous stabilization and tracking of basic automobile drifting trajectories," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, no. Iv. IEEE, jun 2016, pp. 597–602. [Online]. Available: http://ieeexplore.ieee.org/document/7535448/

[9] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg, "Aggressive Deep Driving: Model Predictive Control with a CNN Cost Model," *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78, no. CoRL, pp. 133–142, jul 2017.

[10] Se-Young Oh, Jeong-Hoon Lee, and Doo-Hyun Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 3, pp. 997–1005, may 2000. [Online]. Available: http://ieeexplore.ieee.org/document/845116/

[11] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, no. June. IEEE, may 2015, pp. 3223–3230. [Online]. Available: http://ieeexplore.ieee.org/document/7139643/

[12] I. Rivals, D. Canas, L. Personnaz, and G. Dreyfus, "Modeling and control of mobile robots and intelligent vehicles by neural networks," in *Proceedings of the Intelligent Vehicles '94 Symposium*. IEEE, 1994, pp. 137–142. [Online]. Available: http://ieeexplore.ieee.org/document/639489/

[13] Y. Chen, A. Hereid, H. Peng, and J. Grizzle, "Synthesis of safe controller via supervised learning for truck lateral control," *arXiv*, no. December, pp. 1–13, dec 2017. [Online]. Available: http://arxiv.org/abs/1712.05506

[14] R. Cui, C. Yang, Y. Li, and S. Sharma, "Adaptive Neural Network Control of AUVs With Control Input Nonlinearities Using Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 6, pp. 1019–1029, jun 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7812772/

[15] H. B. Pacejka, *Tyre and Vehicle Dynamics*. Butterworth-Heinemann, 2002.

[16] R. Rajamani, *Vehicle Dynamics and Control*. Springer, 2012.

[17] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," *IEEE transactions on intelligent transportation systems*, vol. 16, no. 4, pp. 2247–2256, 2015.

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

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

[21] "Tass international," http://www.tassinternational.com/prescan.

[22] R. C. Coulter, *Implementation of the Pure Pursuit Path Tracking Algorithm*. Carnegie Mellon University, 1992.

[23] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge: Research articles," *J. Robot. Syst.*, vol. 23, no. 9, pp. 661–692, Sept. 2006.