Trajectory Tracking of Autonomous Vehicles Using Data-driven Modeling and Neural Predictive Control (NPC)

Mahrokh Hosseinkhani Hezaveh, Mehdi Foroozan Koodiyani, Howard Schwartz and, Ioannis Lambadaris

***Abstract*—The technology of autonomous vehicles is advancing rapidly, relying heavily on high-precision simulations and high-performance controllers. This study uses actual driving data to model coupled lateral and longitudinal vehicle dynamics with two neural network (NN) structures: a time-distributed multi-layer perceptron and a long short-term memory (LSTM). Inputs include accelerator pedal position, brake pressure, steering wheel angle, gear number, and road slope, while outputs are acceleration and yaw rate. Additionally, a bidirectional LSTM (Bi-LSTM) predicts vehicle control inputs based on expert driver data. A resolution of the path to be tracked is provided to the NN driver, which generates control commands that are subsequently fed into vehicle NN models. The vehicle's output states are used to compute the quadratic cost function for trajectory tracking. The minimum cost is identified, and the corresponding control inputs are selected as the optimal inputs for each prediction horizon. A comparative analysis was performed between the neural predictive controller (NPC) algorithm and the traditional nonlinear model predictive controller (NMPC), utilizing the interior point optimizer (IPOPT). Incorporating high-fidelity models within NMPC leads to substantial computational overhead. NPC method maintains both computational efficiency and is adaptive to road slope disturbances, without the need for a classical adaptive control design, making it suitable for real-time control applications.**

***Index Terms*—Data-driven modeling, Autonomous vehicle,Trajectory Tracking, Nonlinear model predictive controller(NMPC), Iterative learning control**

I. Introduction

Dynamic vehicle modeling in autonomous vehicles has evolved significantly, progressing from simple abstractions [1] to detailed data-driven representations [2]–[4] that capture complex real-world behaviors. Early models primarily used kinematic representations, considering the vehicle as a point mass with simplified motion constraints. These basic models were suitable for low-velocity scenarios but lacked the fidelity needed for more dynamic driving situations [1].

Machine learning improves the modeling of vehicle dynamics by eliminating the need for costly parameter estimation [5], while generating dynamic models that capture features often neglected in traditional mathematical approaches. This is achieved through the integration of sensory data and advancements in machine learning techniques [2]–[5]. Accurate modeling of dynamics is crucial for the vehicle trajectory tracking module [6]–[8], as it directly influences the ability to maintain stability, follow trajectories precisely, and ensure safe navigation under varying driving conditions. To achieve this, model predictive control (MPC) [9], [10] has been widely adopted for its ability to predict future system behavior and optimize control inputs to minimize predefined cost functions. The work in [11] introduces a data-driven MPC framework for linear time-invariant system, using measured input-output data instead of traditional system models. Researchers in [12] employ deep neural networks with Koopman operators to model and control autonomous vehicle, achieving accuracy and efficiency with deep extended dynamic mode decomposition MPC compared to traditional methods. However, the dataset utilized for validation is entirely simulated.

The key contributions of this paper are as follows: This paper introduces a novel neural predictive controller (NPC) that leverages a bidirectional long short-term memory (Bi-LSTM) to emulate driver behavior and generate optimal control commands for trajectory tracking. The NPC processes a batch of planned paths, evaluates them using a quadratic cost function accounting for trajectory errors and control effort, and selects the path with the minimum cost, further refining it through path augmentation. This iterative algorithm enables the NPC to handle dynamic disturbances, for instance road slopes, while maintaining real-time computational efficiency and robust tracking accuracy. Additionally, the paper employs data-driven modeling of coupled longitudinal and lateral vehicle dynamics using time-distributed multi-layer perceptron (TD-MLP) and long short-term memory (LSTM) architectures, trained on a real-world driving dataset. These models enhance the accuracy and adaptability of the NPC, ensuring precise trajectory tracking under complex driving conditions. We believe our research is the first in the literature to utilize actual driving data for modeling vehicle dynamics, capturing the complete process from pedal and steering wheel commands to vehicle position. Additionally, our proposed controller is uniquely inspired by real driver decision-making, offering a novel approach to trajectory tracking and control.

The remainder of this paper is structured in the following way. Section II details the dataset and the data-driven approach for modeling vehicle dynamics and a driver model. Section

This paragraph of the first footnote will contain the date on which you submitted your brief for review. It will also contain support

Mahrokh Hosseinkhani Hezaveh is with the Department of System and Computer Engineering, Carleton University, Ottawa, Ontario, Canada (email: MahrokhHosseinkhaniH@cmail.carleton.ca). Mehdi Foroozan Koodiyani is with the Automatic Control Laboratory, Department of Mechanical Engineering, K.N.Toosi University of Technology, Tehran, Iran (e-mail: Forouzan.mehdi73@gmail.com). Howard Schwartz is a Professor in the Department of System and Computer Engineering, at Carleton University, Ottawa, Ontario, Canada (email: schwartz@sce.carleton.ca). Ioannis Lambadaris is a Professor in the Department of System and Computer Engineering, at Carleton University, Ottawa, Ontario, Canada (email: ioannis@sce.carleton.ca).

III outlines the neural network (NN) models and training parameters. Section IV describes the NPC algorithm. Section V evaluates the data-driven vehicle dynamics model in comparison to ground truth data, comparing NPC with nonlinear model predictive control (NMPC) for a simulated cloverleaf trajectory. Section VI concludes the study.

II. Data-driven Modeling

This section introduces the dataset and preprocessing and then explains the relevance for modeling vehicle dynamics using NNs.

A. Dataset and preprocessing

In this work, the Inertial Odometry Vehicle Navigation Benchmark Dataset (IO-VNBD) [13] is used for training and testing NNs. From the dataset description, 40 hours of driving data were recorded using the Racelogic VBOX Video HD2, which captures data directly from the vehicle's CAN bus and GPS sensors with a 10 Hz sampling rate. This dataset includes driving data collected with a Ford Fiesta under various conditions, such as slippery roads, sharp turns, stationary motion, winding roads, inner cities, traffic congestion, town centers, and motorways, by eight drivers using both defensive and aggressive driving styles. The dataset includes over 20 minutes of stationary vehicle data, specifically recorded to estimate and correct sensor bias. Velocities below 2 m/s were excluded from the analysis. During preprocessing, intervals with GPS outages were manually identified and removed. The GPS, inertial measurement unit, and odometry data were fused using a Kalman filter [14]. Steering wheel angle biases were corrected using straight-path sections. Finally, the processed data were segmented into appropriate lengths for NN training and testing.

B. Modeling of vehicle dynamics

This section describes NNs designed to process inputs that represent steering wheel angle, accelerator pedal position, brake pedal pressure, gear number, and road slope as disturbances, producing vehicle states as outputs. A vehicle system can be expressed as follows:

$$x_{t+1} = f(x_t, u_t, w_\theta).$$

Here t represents the time step, $x \in R^n$ is the state vector, $u \in R^m$ stands for the inputs vector, w_θ describes the set of uncertain parametric variables, and f is the vehicle transition function. We define $x = [X, Y, \phi, v_x, v_y, a_x, a_y, \dot{\phi}]$ where, X and Y are the vehicle's center of gravity position in the global coordinates, ϕ is the yaw angle, v_x and v_y refer to the longitudinal and lateral velocities at the center of gravity, and a_x and a_y indicate the longitudinal and lateral accelerations at the center of gravity. Instead of estimating the complete state vector x in the output of the NNs, it is feasible to estimate solely the longitudinal acceleration on the rear axis $a_{x,r}$ and the yaw rate $\dot{\phi}$ of the vehicle. Consequently, the remaining state variables can be calculated using the following equations:

$$v_{x,r_{t+1}} = v_{x,r_t} + a_{x,r_t} dt, \tag{1}$$

$$\phi_{t+1} = \phi_t + \dot{\phi}_t dt, \tag{2}$$

$$X_{t+1} = X_t + v_{x,r_t} \cdot \cos(\phi_t) dt, \tag{3}$$

$$Y_{t+1} = Y_t + v_{x,r_t} \cdot \sin(\phi_t) dt. \tag{4}$$

where dt is the sampling time in seconds. Using this configuration, a significant portion of the NN operation, which integrates acceleration and yaw rate to reach other state variables, is reduced. Thus, the NN model is as follows:

$$[a_{x,r_t}, \dot{\phi}_t] = f_{\text{NN, model}}(v_h, u_h, \theta_h, W_{NN,m}). \tag{5}$$

where $f_{\text{NN, model}}(v_h, u_h, \theta_h, W_{NN}, m)$ is the NN model function. Where $v_h = [v_{x,r_t}, v_{x,r_{t-1}}, \ldots, v_{x,r_{t-N_h}}]$ is the history of rear axle longitudinal velocity, $u_h = [u_t, u_{t-1}, \ldots, u_{t-N_h}]$ is the history of control inputs of the system, $\theta_h = [\theta_t, \theta_{t-1}, \ldots, \theta_{t-N_h}]$ is the history of longitudinal road slope as a disturbance input, and $W_{NN,m}$ includes the weights and biases of the NN model. N_h is the history horizon. Control inputs in the model are aligned with real-world driving conditions, as defined in the benchmark dataset [13], where the control input is specified as:

$$u = [\delta, A_{\text{Pos.}}, B_{\text{Pre.}}, G]^T.$$

Here δ is the steering wheel angle of the vehicle, $A_{\text{Pos.}}$ is the position of the accelerator pedal, $B_{\text{Pre.}}$ is the brake pressure, and G is the gear number. Since acceleration and yaw rates depend on velocity, the velocity history must be included as part of the vehicle NN input.

C. Driver modeling

Driving involves continuous monitoring of road conditions and dynamic adjustments to vehicle control inputs, including steering, throttle, and braking, which are influenced by the vehicle's state, the reference path, and the driver's anticipated trajectory. For example, when approaching a turn, precise modulation of the steering and pedal inputs is required to ensure safe and efficient navigation. Inspired by this, this paper presents a novel data-driven model. This model is constructed to derive a function that maps reference path segments, coupled with a prediction horizon length, to the requisite control inputs. Therefore the NN function for modeling the driver's behavior can be expressed in the following form:

$$[M]^T = f_{\text{NN, driver}}(x_{\text{ref}_N}, y_{\text{ref}_N}, \theta_{\text{ref}_N}, v_{\text{ref}_N}, G_N, W_{NN_d}).$$

Where $f_{\text{NN, driver}}$ is the NN model of the driver, y_{ref_N} and x_{ref_N} are the latitude and longitude of the path in local coordinates from time step $t+1$ to the prediction horizon N. θ_{ref_N} represents the slope values of the reference path. v_{ref_N} consists of the current vehicle velocity at time step t and the reference velocity derived from the discrete derivative of the reference path–x_{ref_N} and y_{ref_N}. G_N refers to the gear number during the prediction horizon, and W_{NN_d} represents the weights and biases of the driver NN. M contains the predicted steering wheel angle, δ_N; accelerator position, A_{Pos_N}; and brake pressure, B_{Pre_N}. The subindex N represents the time steps from $t+1$ to the prediction horizon.

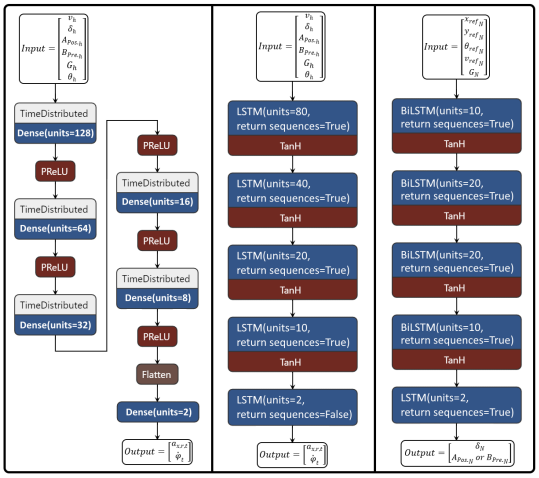

Fig. 1: The proposed architectures: (left) TD-MLP for the vehicle model, (middle) LSTM for the vehicle model, and (right) Bi-LSTM for the driver model.

III. NN ARCHITECTURES

This study proposes two NNs architectures, namely the TD-MLP and LSTM network, to model vehicle dynamics. Additionally, a Bi-LSTM architecture is introduced to represent driver behavior. These architectures provide enhanced flexibility by allowing the adjustment of prediction and control horizons post-training, a capability that is often lacking in conventional NN models.

A. Vehicle TD-MLP architecture

Fig. 1 (left) shows the proposed TD-MLP for vehicle dynamics modeling. This model is fed with a total of $N_h + 1$ time steps of inputs, and each step lasts 0.1 seconds. After the last TD layer, a flattening layer is used, and its output is fed to two neurons with linear activation functions to predict acceleration and yaw rate as outputs.

B. Vehicle LSTM architecture

Fig. 1 (middle) shows the LSTM NN for vehicle dynamics. The input of this structure is the same as the TD-MLP NN, a time span of N_h+1 steps of vehicle inputs, and each time step lasts 0.1 seconds. According to the proposed architecture in Fig. 1 (middle), the structure uses five LSTM layers with 80, 40, 20, 10, and 2 units, respectively. The final LSTM layer has a linear activation function to perform regression operations.

C. Driver Bi-LSTM architecture

The driver model architecture is illustrated in Fig. 1 (right). This structure consists of four Bi-LSTM layers and one final LSTM layer. Since only one of the brake and accelerator pedals has a non-zero value while driving, one unit with a linear activation function is used in the output layer to predict the values of these pedals' commands and a second output unit is used to predict the steering wheel angle. In this case, the positive values of this unit correspond to the position of the accelerator pedal, and the negative values represent the brake pressure.

TABLE I: NN training hyperparameters

Parameter	Value
Number of Epochs	50
Initial Learning Rate	0.01
Decay Steps	40
Decay Rate	0.998
Optimizer	ADAM

D. NNs training

In NNs, the selection of optimal hyperparameters is highly important, since it greatly impacts the final accuracy. Table I presents the hyperparameters associated with the NNs training. Table II presents the RMSE values for the predicted states. We utilized 20 hours of recorded driving data for training and testing purposes. Approximately 70 minutes of data, was reserved for testing. We ensured its selection based on its diverse content, including scenarios with sharp turns, to provide a comprehensive and rigorous evaluation of the model's performance. To train the NNs, N_h is set to 9 and N to 10.

IV. NPC DESIGN

Although the driver model is used to emulate driver behavior to generate the required control inputs for navigating a desired path, two significant challenges persist. Firstly, it is crucial to acknowledge that this NN at best approximates driver behavior. Given that drivers do not consistently provide optimal control commands, the output of the driver NN is not always optimal either. Secondly, the challenges extend where discrepancies between the controlled environment and real-world scenarios arise. During training, the vehicle consistently starts from the same initial position as the reference path. However, such ideal conditions rarely occur in actual situations, leading to differences between the vehicle's initial position and reference path during maneuvers. This underscores the controller's capability to manage scenarios that may deviate from those that driver NN had seen during training process. To design the NPC, it is assumed that the vehicle's position at the current time step, $k = 0$, differs from the starting point of the reference path. In other words, we have:

$$\sqrt{{x_{\text{ref}_k}}^2 + {y_{\text{ref}_k}}^2} > 0. \tag{6}$$

where, for $k = 0$, y_{ref} and x_{ref} are the coordinates of the first point on the reference path in the vehicle's local coordinates attached to the body. This means that y_{ref} and x_{ref} denote the location of the first point from the perspective of the vehicle itself, rather than in an absolute coordinate system. Because of inequality (6), a path planning method is required to guide the vehicle to the reference path. It is not the primary objective of this study to address the problem of path planning. Therefore, a simple path planning approach, as described in (7)-(10), is utilized to design the desired path components x_{pp_k} and y_{pp_k}, where "pp" denotes the planned path. The values y_{p_k} and x_{p_k} indicate the trajectory that the vehicle would follow, taking into account its current velocity and steering wheel angle, within the prediction horizon. These values are determined through calculations based on (1) to (5).

$$x_{\text{pp}_k} = x_{\text{p}_k} + (x_{\text{ref}_k} - x_{\text{p}_k}) \cdot \text{sigmoid}(k, C_a, C_c), \tag{7}$$

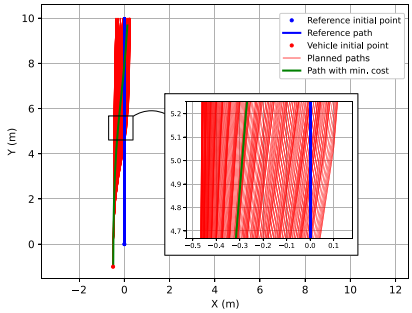

Fig. 2: Generated trajectories toward the reference path using different curvature.

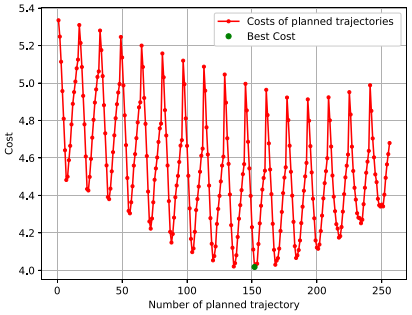

Fig. 3: The green dot indicates the minimum value of the costs associated with P^2=256 planned paths.

$$y_{\text{pp}_k} = y_{\text{p}_k} + (y_{\text{ref}_k} - y_{\text{p}_k}) \cdot \text{sigmoid}(k, C_a, C_c), \tag{8}$$

$$\text{sigmoid}(k, C_a, C_c) = \frac{1}{1 + e^{-C_a(k - C_c)}}, \tag{9}$$

$$k = \{1, \ldots, N\}. \tag{10}$$

Here, the sigmoid curve is shaped by parameters C_a and C_c. The parameter C_a represents the maximum slope in the sigmoid curve, while C_c represents a centrosymmetric point. In this work, $C_c = \frac{4.6}{C_a}$ is considered, indicating that:

$$\text{sigmoid}(k = 0, C_a, 4.6/C_a) = 0.01.$$

Accordingly, for all values of C_a, the sigmoid function always has a value close to zero at $k = 0$. Different values of C_a have been considered in the range $C_a \in [0.3, 1]$. Consequently, by considering P different number of C_a values and combining all $x_{\text{pp}} = [x_{\text{pp}_{k=1}}, \ldots, x_{\text{pp}_{k=N}}]$ and $y_{\text{pp}} = [y_{\text{pp}_{k=1}}, \ldots, y_{\text{pp}_{k=N}}]$, which are generated using (7) and (8), P^2 new paths with different curvature intensities are created towards the reference path. As depicted in Fig. 2, we generated a total of P^2 planned trajectories. Fig. 2 shows these generated paths in red. This quantity can vary depending on the computational resources available. In the next step, these P^2 paths are fed as a batch input to the driver NN. As a result, the driver NN generates a batch of P^2 control command vectors to navigate through these paths. Thus, the shape of the matrix is $P \times P \times N$. This implies that:

$$\begin{aligned}&[\delta_{[P^2 \times N]}, (A_{\text{Pos.}} \text{ or } B_{\text{Pre.}})_{[P^2 \times N]}]^T \\ &= f_{\text{NN,driver}}(x_{\text{pp}_{[P^2 \times N]}}, y_{\text{pp}_{[P^2 \times N]}}, \\ &\quad \theta_{\text{pp}_{[P^2 \times N]}}, v_{\text{pp}_{[P^2 \times N]}}, G_{[P^2 \times N]}, W_{NN_d}).\end{aligned} \tag{11}$$

Where θ_{pp} and v_{pp} are the road slope and velocity corresponding to the planned paths. We ignore any deviation in heading angle between the vehicle and the reference path at $k = 0$, assuming that the difference is negligible during the maneuver.

The generated control inputs are then fed into the vehicle model in (5), and model predicted paths are calculated by combining (1)-(4). By having the control inputs and their model predicted paths, (13) calculate the corresponding costs.

$$\begin{aligned}J^p = &\sum_{k=1}^{N} w_d (e^p_{d,k})^2 + w_\psi (e^p_{\phi,k})^2 + w_\delta (\Delta \delta^p_k)^2 \\ &+ w_{A_{\text{Pos.}}} (\Delta A^p_{\text{Pos.}k})^2 + w_{B_{\text{Pre.}}} (\Delta B^p_{\text{Pre.}k})^2\end{aligned} \tag{12}$$

Subject to:

$$-475^\circ \leq \delta_k \leq 475^\circ \tag{13}$$

$$0\% \leq A_{\text{Pos.}k} \leq 100\% \tag{14}$$

$$0\,\text{psi} \leq B_{\text{Pre.}k} \leq 190\,\text{psi} \tag{15}$$

And:

$$e^p_{d_k} = \sqrt{(\tilde{x}^p_k - x_{\text{ref}_k})^2 + (\tilde{y}^p_k - y_{\text{ref}_k})^2}$$

$$e^p_{\psi_k} = \sqrt{(\tilde{\psi}^p_p)_k - (\psi_{\text{ref}_k})^2}$$

$$p = 1, 2, \ldots, P^2$$

Here, J^p represents the cost associated with the pth control inputs and the corresponding model predicted path. In (13), the first two terms refer to the distance and heading error of the vehicle, respectively. Δ represents the change in the control inputs between consecutive time steps. The terms $\tilde{x}_{p,k}$ and $\tilde{y}_{p,k}$ represent the predicted position in local coordinates for the kth time step. Weighting coefficients w_d, w_ψ, w_δ, $w_{A_{\text{Pos.}}}$, and $w_{B_{\text{Pre.}}}$ are chosen based on experience and controller performance. When the values of J^p are computed for $p = 1, 2, \ldots, P^2$, the path with the minimal cost, represented by the green color in Fig.2, is determined among the P^2 generated paths. Fig. 3 shows the cost function values that correspond to the example shown in Fig. 2.

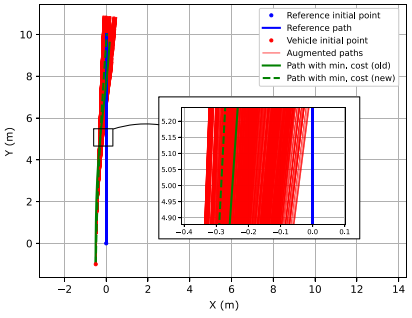

Fig. 4: Augmented paths and path with minimum cost found in the first iteration of path augmentation.

After identifying the lowest-cost path in the first iteration, the process transitions to the path augmentation phase. This phase refines the selected path by introducing random Gaussian noise to the velocity and yaw rate, enabling a higher-resolution search for potentially lower-cost paths. The new trajectories are computed using (2)-(4), and the noise variance

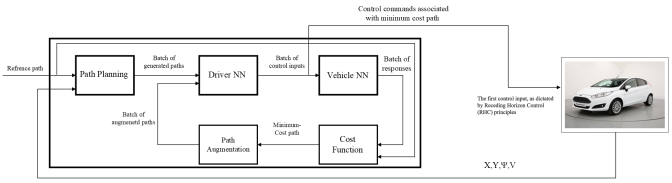

Fig. 5: Block diagram of the proposed NPC algorithm. The plant is the vehicle NN model, and the actual vehicle image is for visualization.

progressively decreases in successive iterations to improve resolution. Path augmentation continues until the stopping criteria are met, defined as follows:

$$\begin{cases} J_{\min,i} < \text{Th}_{\text{cost}} \\ \quad \text{or} \\ \dfrac{J_{\min,i} - J_{\min,i-1}}{J_{\min,i}} < \text{Th}_{\text{diff}_{\text{cost}}} : \ i > 1 \end{cases} \tag{16}$$

Here, $J_{\min,i}$ is the minimum cost at iteration i, Th_{cost} is the threshold for acceptable cost, and $\text{Th}_{\text{diff}_{\text{cost}}}$ is the threshold for acceptable cost difference. These thresholds depend on the cost scale. In the NPC algorithm, receding horizon control (RHC) is employed as the final step. Once the minimum-cost path is identified, the corresponding optimal control commands are derived from the driver NN. Following RHC principles, only the first control input is applied to the system, and the process repeats at the next time step for the updated state. This iterative optimization continues until the end of the path. Fig. 5 illustrates the block diagram of the proposed NPC controller.

Fig. 2 illustrates an example of planned paths (red) generated using (7) and (8), where the green path represents the one with the lowest cost. Fig. 4 then shows the augmented paths (red) generated around the best path (solid green) during the path augmentation phase. The dashed green line in Fig. 4 represents the newly optimized path after augmentation.

V. Results and Discussion

A. Vehicle modeling results

The initial evaluation of the NN model follows the standard validation approach used during training, where predictions are tested over short time horizons to asses immediate, step-by-step accuracy. The open-loop test uses 100 seconds of sequential control inputs to evaluate how well the model maintains accuracy over time, highlighting potential error accumulation or drift that may not appear in short-horizon tests.

- NN model evaluation

The test data is divided into segments of 10-time step length series, as mentioned in the methodology where this number is $N_h + 1$ and N_h is set to 9. According to (5), the vehicle model is utilized to predict the values of a_x, r_t, and ϕ_t. The performance results of the TD-MLP, LSTM and Bi-LSTM are presented in Table III. It is clear from the results that the TD-MLP architecture has performed slightly better than the LSTM.

- Open-loop test

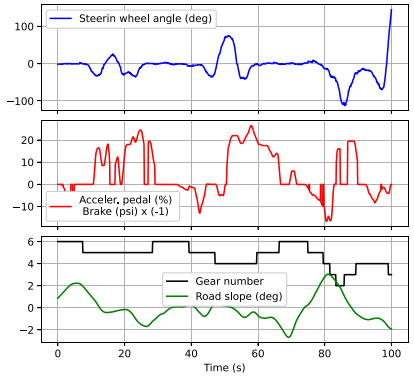

Fig. 6: A 100-second sample of control commands and road slope.

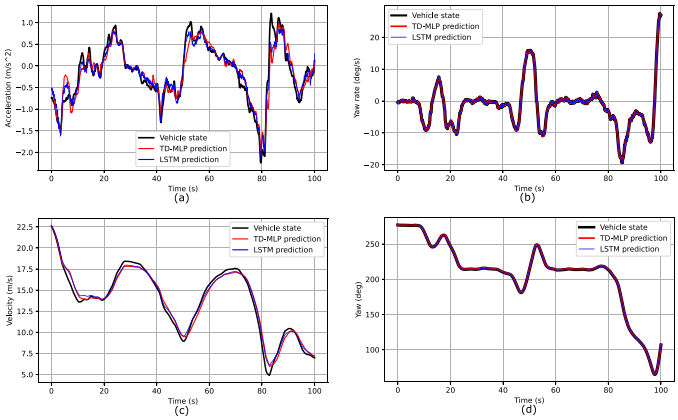

Fig. 7: Comparison of actual vehicle states with predicted states from the proposed NN models, including TD-MLP and LSTM.

The NN models, in an open-loop method, receive driver control inputs $u = [\delta, A_{\text{Pos.}}, B_{\text{Pre.}}, G]^T$, path slopes, and initial vehicle velocities from test data. Vehicle state values are then updated using (1)-(5). It is worth noting that, in the long-term open-loop test, numerical integration over an extended horizon can lead to the accumulation of integration errors. Therefore, the fourth-order Runge–Kutta method was employed to compute X and Y, instead of the simpler forward Euler approach used in (3) and (4).

Fig. 6 illustrates a 100-second sample of control inputs and road slope applied to the NN models for long-term prediction. This figure presents the steering wheel angle in degree units in the top plot, percentages of accelerator pedal position (values greater than zero) and brake pressure in psi (absolute values less than zero) in the middle plot, as well as gear numbers and road slope in the bottom plot.

Corresponding to the inputs shown in Fig. 6, Fig. 7 illustrates the long-term prediction of acceleration, yaw rate, velocity, and yaw angle in the (a), (b), (c), and (d) respectively. The predictions generated by the NN models for acceleration and yaw rate states demonstrate remarkable accuracy, closely matching the actual values observed in the test data.

As shown in Fig. 7(c), over the 100-second timeframe, the cumulative impact of predicted acceleration errors had minimal influence on the velocity output, which closely aligns with the actual data. Observing the predicted yaw angle in Fig. 7(d), the overlap between NN model predictions and the actual

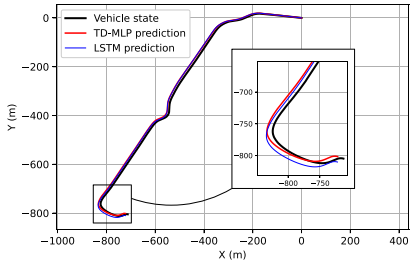

Fig. 8: Analyzing predicted trajectories in comparison to actual trajectories driven by the vehicle.

TABLE II: TD-MLP, LSTM, Bi-LSTM RMSE in train and test.

Model	Output	Unit	RMSE (train)	RMSE (test)
TD-MLP	$a_{(x,r)}$	m/s^2	0.0847	0.0995
	$\dot{\phi}$	deg/s	0.0307	0.0370
LSTM	$a_{(x,r)}$	m/s^2	0.0901	0.1135
	$\dot{\phi}$	deg/s	2.008	1.862
Bi-LSTM	$A_{\text{Pos.}}$	%	0.0901	0.1135
	$B_{\text{Pre.}}$	psi	1.691	1.933
	δ	deg	1.777	2.174

TABLE III: RMSE for NN models in the open-loop test.

Model	$a_{(x,r)}$ (m/s^2)	$\dot{\phi}$ (deg/s)	v (m/s)	ϕ (deg)
TD-MLP	0.2141	0.0261	0.5804	0.2364
LSTM	0.1528	0.0426	0.4940	0.1195

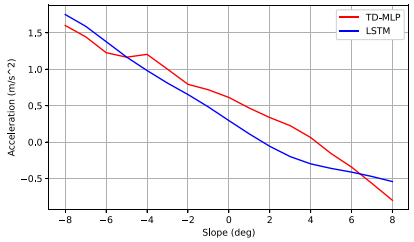

Fig. 9: The effects of road slope on the predicted acceleration by NN models.

vehicle values indicates the effectiveness of the NN models in capturing lateral vehicle dynamics.

The predicted and ground truth trajectory, covering more than one kilometer, is shown in Fig. 8. During this 100-second interval, the input conditions pose significant challenges, with continuous variations in throttle and brake inputs leading to fluctuations in vehicle speed within the range of 7 m/s to 22.5 m/s. Additionally, the gear selection transitions frequently, and the road gradient exhibits a wide range of values. Despite these challenges, the NN models demonstrate robustness in accurately predicting the trajectory under diverse conditions.

One of the primary goal of NN modeling is to analyze the relationship between road slope and vehicle dynamics, emphasizing its role in controller design and highlighting variables often overlooked in analytical models. For instance, Fig. 9 shows that traveling uphill requires more gas pedal pressure to maintain constant velocity. To evaluate the NN models' response to slope changes, a test was conducted with fixed inputs: steering wheel angle ($\delta = 0°$), accelerator position ($A_{\text{Pos.}} = 25\%$), brake pressure ($B_{\text{Pre.}} = 0$), gear number ($G = 3$), and an initial velocity of 10 m/s. The slope was varied from -8 to 8 degrees. Results showed that acceleration was positive downhill and turned negative uphill, confirming that NN models accurately capture the slope-acceleration relationship. This test highlights the adaptability of NN models to real-world vehicle dynamics by isolating the slope's impact on acceleration.

TABLE IV: Controller Parameters

Control Parameter	Value
History horizon, N_h	9
Prediction horizon, N	10
Time step, t	0.1 s
Distance error weight, w_d	0.5
Heading error weight, w_ψ	0.5
Steering wheel angle weight, w_δ	0.0001
Pedal input weight, $w_{A_{\text{Pos.}}}, w_{B_{\text{Pre.}}}$	0.001
NMPC termination tolerance	1×10^{-6}
Minimum cost threshold, Th_{cost}	0.1
Cost difference threshold, $\text{Th}_{\text{diff}_{\text{cost}}}$	5%

B. NPC tracking results

This section compares NPC with two NMPC variants, both based on solving a finite-horizon optimal control problem with the same quadratic cost function and the same weights, system dynamics, and constraints, as in NPC (12)-(16). Two NMPC versions were considered: NMPC, with less strict stopping conditions, and NMPC-2, which used stopping conditions similar to NPC. Both NMPC variants utilized the interior point optimizer (IPOPT) algorithm [15], a robust method for solving nonlinear optimization problems. The NN vehicle model served as the prediction model for both NMPC and NPC, with the reference trajectory being a cloverleaf interchange, a non-level intersection. For all trajectory-tracking experiments, a constant reference velocity of $v_{\text{ref}} = 10$ m/s was assumed. This constant-speed assumption directly determines the temporal spacing of the reference path $(x_{\text{ref}}, y_{\text{ref}})$, ensuring uniform sampling of the desired trajectory in time. As shown in the slope profile of the reference trajectory visible in Fig. 11(a), this scenario includes both positive and negative road slopes. The gear number was fixed to 4. This choice reflects the typical operating condition observed in the dataset, where speeds of 9–11 m/s and slopes between -6° and +6° were predominantly driven in 4th gear. Gear control is outside the scope of this study but could be integrated into the NPC without modifying the core method. Although the vehicle and driver NN models, and consequently the optimization algorithm, are capable of receiving slope information, the road slope was intentionally set to zero in NPC and NMPC to introduce uncertainty during operation. This setup allows us to evaluate the robustness of both controllers in the presence of a slope disturbance. The value of controller parameters are listed in Table IV. As shown in Fig. 10, positive values represent accelerator pedal position, while the absolute values of negative data indicate brake pressure. NMPC-2 exhibited poor control outcomes, including significant oscillations in control values. Unlike NMPC, which allows more iterations, NMPC-2's limited optimization steps make it more sensitive to prediction errors and disturbances, amplifying high-frequency variations. The key difference is the

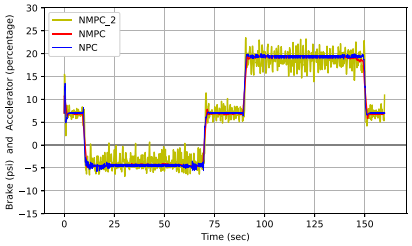

Fig. 10: Pedal inputs generated by NMPC-2, NMPC, and NPC.

truncated optimization process in NMPC-2, which limits its ability to optimize smooth control commands effectively. Due to its subpar performance, subsequent analyses focus solely on NPC and NMPC.

Fig. 11 should be observed sequentially: (a)-(c)-(e) and (b)-(d)-(f). Fig. 11(a) illustrates the reference trajectory slope profile with zero, positive, and negative values, enabling an analysis of pedal control responses to slope changes. In Fig. 11(c), negative slopes (downhill) prompt the controller to reduce accelerator input and engage the brakes, while positive slopes (uphill) increase accelerator input. As shown in Fig. 11(e), the vehicle closely follows the reference path with a velocity near 10 m/s. Initially, a one-meter position offset causes a slight velocity increase, corrected by reducing accelerator input to maintain a constant 10 m/s.

In Fig. 11(d), the controllers initially applied a steering wheel angle of about 90 degrees to account for the vehicle's offset from the reference path in Fig. 12. This caused a slight heading angle deviation in Fig. 11(f), which the controllers promptly corrected. Figure 11(b) illustrates the road curvature, a primary factor governing vehicle steering. Variations in curvature, shown in Fig. 11(b), lead the controllers to adjust the steering wheel angle (Fig. 11(d)), which subsequently drives the evolution of the vehicle's heading angle over time (Fig. 11(f)). Throughout the scenario, the controllers consistently generated steering commands that corresponded to the curvature of the reference path.

At the 10th and 150th seconds, significant changes occurred in the road gradient, shifting from 0 degrees to -5 and from 5 degrees to 0 degrees, respectively. As a result of these abrupt changes in slope, the vehicle experienced a momentary increase in velocity, exceeding the 10 m/s target. At the 10th second, the control system reduced the accelerator pedal input and applied the brakes, while at the 150th second, the controllers decreased the accelerator input to maintain the reference velocity. Furthermore, at the 70th and 90th seconds, when the road incline steepened by 5 degrees, the vehicle's velocity decreased. In these instances, the controllers responded by reducing brake pressure and increasing accelerator pedal input to restore the velocity to the desired reference value 10 m/s. It is worth noting that no input saturation was observed in the conducted simulation.

As depicted in Fig. 12, the NMPC algorithm demonstrates a slight performance advantage over NPC, particularly within the initial 10 meters of the path, as it reaches the reference trajectory slightly ahead of NPC. This improved performance can be attributed to the looser stopping conditions employed

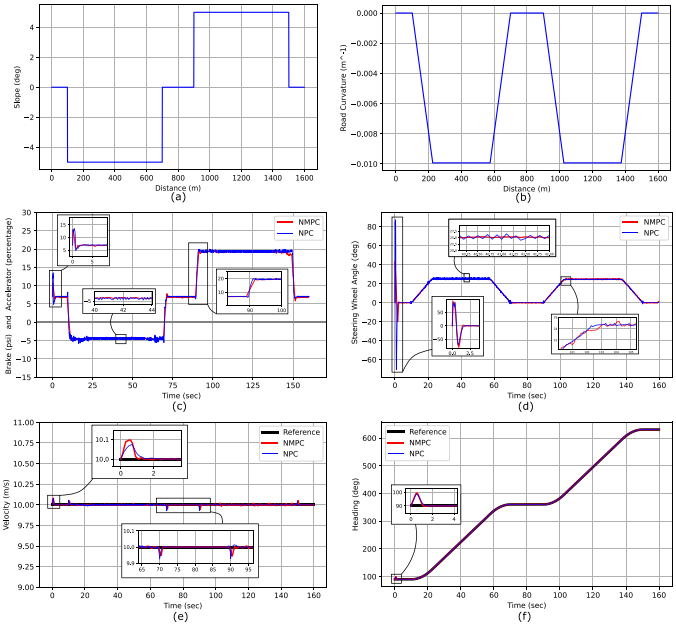

Fig. 11: Longitudinal slope of reference path (a), curvature of reference path (b), accelerator and brake pedal inputs (c), steering wheel angle input (d), velocity (e), and heading (f).

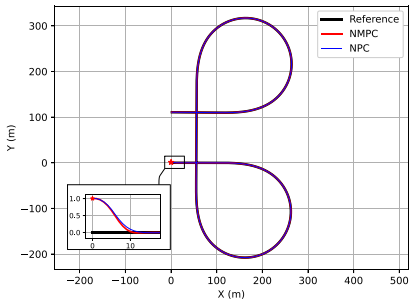

Fig. 12: Trajectory tracking performance of NPC and NMPC.

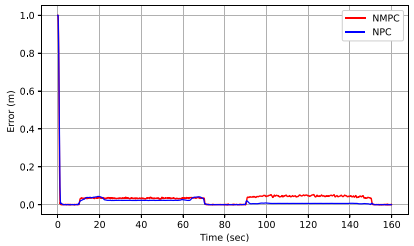

Fig. 13: Distance error for NMPC and NPC.

in IPOPT, which lies at the heart of the NMPC algorithm.

Fig. 13 reveals that NPC outperformed NMPC during downhill segments (10th to 70th second, -5-degree slope) and was even more effective during uphill segments (90th to 150th second, 5-degree slope). This demonstrates NPC's reliability in handling road slope disturbances. The path augmentation method consistently reduced the optimal control cost by 4–20% across all scenarios. Smaller improvements ($\approx$ 4–8%) occurred when the road slope was zero, since the initially planned paths were already close to optimal. Larger reductions ($\approx$ 8–20%) appeared in uphill and downhill segments, where position errors were greater.

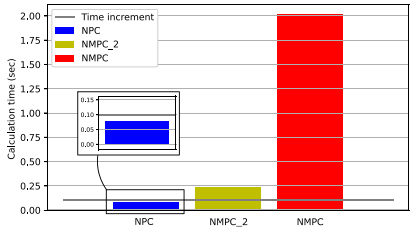

Fig. 14: Calculation time comparison.

In this research, conducted on an Intel Core i7, 8 processor (1.90 GHz) and 16 GB DDR4 RAM computer, we found that $P = 16$ was the optimal value for achieving real-time performance of NPC on our hardware. Increasing the number of generated paths beyond 256 could impact real-time performance, especially on less powerful hardware. However, more capable systems can handle a higher number of paths without affecting performance. The simulations are conducted at 0.1-second intervals. As depicted in Fig. 14, a standard Core i7 computer completes each time step in approximately 0.075 seconds for the NPC method, 2 seconds for the NMPC method, and 0.2 seconds for the NMPC-2 method. The longer computation time for NMPC results from the complexity of optimizing NN dynamics.

NPC benefits from prior knowledge through the driver NN, allowing efficient optimization by predicting control commands within the horizon. Its batch processing feature enables simultaneous evaluation of cost functions, speeding up the search for optimal inputs.

In contrast, NMPC uses the IPOPT algorithm, a gradient-based method that iteratively computes gradients to find optimal solutions. Using offline trained NN models in NMPC's online optimization increases computation time, making it less efficient for real-time applications. Additionally, unlike NMPC, which requires solving a constrained nonlinear program at each time step, NPC enforces input limits through a lightweight clipping mechanism. This eliminates the need for constraint gradients [15] and the overhead associated with iterative solvers, thereby significantly reducing computational complexity while preserving tracking performance and demonstrating adaptive behavior under road slope disturbances.

VI. Conclusion

This article presents a coupled longitudinal and lateral dynamic modeling of a vehicle using NNs. The NN models map real control inputs to acceleration and yaw rate. The section IV introduces the NPC technique, where a predefined path horizon is provided to driver NN models, generating control inputs. These commands are applied to the vehicle model to produce vehicle states and calculate a batch of cost functions. The algorithm identifies the path with the minimum cost, then augments new paths around it to seek further cost reductions. This iterative process continues until cost changes become negligible. The process repeats for subsequent control horizons until the path is complete. The study also considers road slope as a disturbance factor, with controllers receiving a constant slope value of zero during the simulation while the test scenario has -5 to +5 degrees road slope. NPC effectively generates control inputs to maintain constant velocity without classic adaptive designs.

Although the path planning method in this study was intentionally simplified, as it was not the primary focus, future work could explore more advanced path planning techniques to further enhance trajectory tracking performance. Moreover, NPC could be extended to autonomous aerial vehicles, necessitating the collection of sensory data tailored to the unique dynamics and control requirements of aerial systems.

References

[1] D. Schramm, M. Hiller, and R. Bardini, "Vehicle dynamics: Modeling and simulation," 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:60211154

[2] Y. U. Yim and S.-Y. Oh, "Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 4, pp. 1076–1084, 2004.

[3] G. Devineau, P. Polack, F. Altché, and F. Moutarde, "Coupled longitudinal and lateral control of a vehicle using deep learning," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 642–649, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:53046755

[4] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, p. eaaw1975, 2019. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.aaw1975

[5] L. Hermansdorfer, R. Trauth, J. Betz, and M. Lienkamp, "End-to-end neural network for vehicle dynamics modeling," in *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, 2020, pp. 407–412.

[6] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[7] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, "Review and performance evaluation of path tracking controllers of autonomous vehicles," *IET Intelligent Transport Systems*, vol. 15, no. 5, pp. 646–670, 2021.

[8] Q. Yao, Y. Tian, Q. Wang, and S. Wang, "Control strategies on path tracking for autonomous vehicle: State of the art and future challenges," *IEEE Access*, vol. 8, pp. 161 211–161 222, 2020.

[9] 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.

[10] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "Mpc-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, pp. 265–291, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:31658345

[11] J. Berberich, J. Köhler, M. A. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Transactions on Automatic Control*, vol. 66, no. 4, pp. 1702–1717, 2021.

[12] Y. Xiao, X. Zhang, X. Xu, X. Liu, and J. Liu, "Deep neural networks with koopman operators for modeling and control of autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 135–146, 2023.

[13] U. Onyekpe, V. Palade, S. Kanarachos, and A. Szkolnik, "Io-vnbd: Inertial and odometry benchmark dataset for ground vehicle positioning," *Data in Brief*, vol. 35, p. 106885, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340921001694

[14] V. Girbés-Juan, L. Armesto, D. Hernández-Ferrándiz, J. F. Dols, and A. Sala, "Asynchronous sensor fusion of gps, imu and can-based odometry for heavy-duty vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8617–8626, 2021.

[15] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.