2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

A Hybrid Approach Combining Upper-Level Policy Search and MPC for Lane Change of Autonomous Vehicles

Francesco Laneve[1], Alessandro Rucco[2], and Massimo Bertozzi[3]

***Abstract*— In this paper we present a novel approach to address the lane change maneuver for Autonomous Vehicles (AVs). We frame this challenge as a parametric Model Predictive Control (MPC) problem, recognizing that the successful execution of lane changes requires two critical components: the decision on when to initiate these maneuvers and the actual generation of the maneuvers. Unlike existing approaches that often decouple decision-making and planning tasks, leading to performance bottlenecks and conservative solutions, our approach adopts a hybrid perspective. Specifically, we tackle the problem of capturing the lane change decision-making through an upper-level policy search, which, in turn, guides an MPC policy in generating the maneuver. The proposed approach leverages a weighted maximum likelihood technique for policy learning, effectively optimizing the lane change strategy. Furthermore, we incorporate self-supervised learning techniques to adapt to dynamic and online scenarios, ensuring that the AV can handle unexpected changes in its environment. We provide numerical results demonstrating the effectiveness of the proposed approach, highlighting its potential for improving AV maneuvering in dynamic environments.**

***Index Terms*— MPC, reinforcement learning, autonomous driving.**

I. Introduction

Autonomous Vehicles (AVs) play a crucial role in the future of transportation, offering the potential to enhance safety and efficiency. Typically, in order to navigate in complex environments, AVs need to execute various maneuvers, including lane changes [1], overtaking [2], [3], and merging maneuvers [4], [5]. These tasks require advanced algorithms that enable an AV to perceive its surroundings, reason in a timely manner (to reach human-level reliability) and execute maneuvers safely. Lane change, in particular, has gained significant research attention, evident in a growing body of literature (see, e.g. [6]–[8]). However, successfully executing a lane change is challenging due to the need to generate collision-free trajectories and the complex decision of when to initiate the maneuver. Typical approaches to address this challenge have often decoupled decision-making and planning tasks. The most common solution employs rule-based lane change models for decision-making and then integrates the decision logic into a trajectory planning module.

[1]Francesco Laneve is with the Dipartimento di Ingegneria e Architettura, Università di Parma and VisLab srl, an Ambarella Inc. company - Parma, Italy, francesco.laneve@unipr.it.

[2]Alessandro Rucco was with VisLab srl, an Ambarella Inc. company - Parma, Italy.

[3]Massimo Bertozzi is with the Dipartimento di Ingegneria e Architettura, Università di Parma, Parma, Italy, massimo.bertozzi@unipr.it.

In rule-based models, AVs determine lane change decisions based on predefined rules, such as lane preference or the feasibility of the maneuver (see, e.g., [9]). However, they may lack accuracy and fail to capture all relevant influencing factors when applied only through threshold-based conditions to assess driving intentions. Other approaches address the problem of capturing the lane change decision as a classification problem. In [10], a data-driven approach is used to "mimic" human driver behavior during lane changes. Real-world data are collected in typical lane change scenarios, and an SVM classifier is employed to predict when a lane change should be initiated based on a specific driver's preferences. However, feature selection and transformation are required to achieve good performance.

Once an AV has determined its driving intention, generating a safe trajectory becomes crucial. Among various control techniques, Model Predictive Control (MPC) has gained popularity due to its ability to handle nonlinear dynamics and state-input constraints, see e.g. [11], [12]. For example, in [13], an MPC-based lane change algorithm is proposed to integrate path planning and path following layers with a utility function to automatically determine the target lane. However, the performance of the MPC in closed-loop operation depends on the design choices of the heuristic decision function, leading to approximations that can result in conservative solutions. On the other hand, Reinforcement Learning (RL), specifically policy search approaches [14], has recently emerged as an innovative method for learning driving policies, as seen in e.g.and [15]. RL involves training a policy through iterative trial and error to maximize a performance function, referred to as the "return", directly translating sensor inputs into actuation commands for the AV. However, it is important to note that RL-based methods typically require large amounts of training data to achieve satisfactory results and may struggle to adapt to unseen situations. Another major concern is the limited safety and stability guarantees provided by RL methods, especially when used in the context of AVs.

In this paper, we adopt a hybrid perspective: we assume that a perception system provides information about the future intentions of other moving vehicles, and we focus on the task of deciding and generating a lane change maneuver through an upper-level policy search within a parametric MPC framework. Various approaches have been explored to combine learning and control in autonomous systems. In [16], a sampling-based MPC approach was developed for AV, with a focus on obstacle avoidance. However, a notable challenge arises from the need to generate a large

Copyright ©2024 EUCA

amount of samples in real-time, often accomplished through computationally expensive parallel processing. In [17], an approach was introduced that integrates machine learning and MPC within an imitation learning framework, applied to lane-keeping maneuvers. Nonetheless, this method involves training deep neural network policies using supervised learning, which relies on ground-truth labels and may have limitations in certain scenarios.

The contributions of this paper are as follows. First, we propose a novel approach to address the lane change problem as a parametric model predictive control problem. The parameter within this framework determines whether to remain in the current lane or transition to an adjacent one. Second, we address the challenge of finding the parameter to execute the lane change maneuver by formulating it as a probabilistic policy search problem. Our approach employs a weighted maximum likelihood method for learning the policy parameter, offering a closed-form solution for policy updates. Third, inspired by the work outlined in [18], [19], which focused on maneuvering a quadrotor through the center of a rapidly moving gate, we employ a self-supervised learning approach. This enables to generate the parameter online based on observations of the surrounding environment. Finally, through numerical simulations, we demonstrate the effectiveness of our proposed approach and discuss interesting features related to the generated maneuvers.

The rest of the paper is organized as follows. In Section II, we describe the lane change problem and formulate the parametic MPC framework. In Section III, we describe the strategy for effectively solving the lane change problem. This strategy is evaluated through numerical computations and illustrated in Section IV. The conclusions are given in Section V.

II. Problem Formulation

Let us consider the scenario depicted in Fig. 1. The road is composed of two parallel lanes, called the *ego lane* and the *target lane*. The AV (from now on, referred as the ego-vehicle) is traveling along the ego lane, while a *front vehicle*, indicated as the FV, is moving on the same lane. We assume that the FV is traveling slower than the ego-vehicle, which motivates a lane change maneuver. In such a scenario, we are interested in generating a lane change maneuver by designing an MPC with an upper-level decision variable. This variable selects the "right time" to execute the maneuver while avoiding the *lateral vehicle*, indicated as the LV, moving on the target lane.

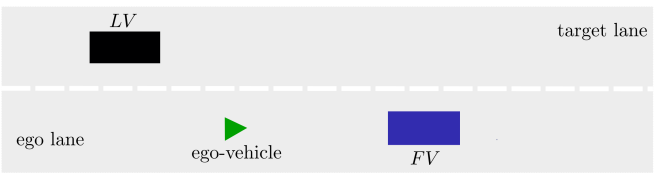

Fig. 1: 2D representation of lane change scenario.

A. Ego-Vehicle Motion

In the given scenario, we describe the 2D motion of a car-like vehicle with the following equations:

$$\begin{aligned} \dot{x} &= v\cos\psi\,, \\ \dot{y} &= v\sin\psi\,, \\ \dot{\psi} &= v\kappa\,, \\ \dot{v} &= a\,, \end{aligned} \tag{1}$$

where (x, y) represent the longitudinal and lateral coordinates in the inertial frame, ψ is the heading angle, and v is the velocity. Control over the ego-vehicle's motion is achieved by manipulating the curvature κ, where $\kappa = \tan\delta/L$ (with δ as the steering angle and L denoting the wheelbase), and the acceleration a. This simplified model is particularly suitable for scenarios with relatively low acceleration inputs and closely approximates the behavior of more complex dynamic models, as detailed in [20]. In the next section, we introduce a new coordinate system defined by the longitudinal and lateral coordinates (s, e_y). Here, s represents the position along the center-line of the ego lane, while e_y indicates the displacement transverse to this center-line, see in Fig. 2. This coordinate system is particularly useful for lane changes, where it is more intuitive to consider the lateral distance from the desired lane position rather than Cartesian coordinates.

B. Longitudinal and Transverse Coordinates

We assume that the ego lane has a reasonably smooth (at least C^2) arc-length parametrized center-line, denoted as $(\bar{x}(s), \bar{y}(s))$. The course heading $\bar{\psi}(s)$ and the curvature $\bar{\kappa}_{cl}(s)$ are related through differentiation:

$$\begin{aligned} \frac{d\bar{x}(s)}{ds} &= \cos\bar{\psi}(s)\,, \\ \frac{d\bar{y}(s)}{ds} &= \sin\bar{\psi}(s)\,, \\ \frac{d\bar{\psi}(s)}{ds} &= \bar{\kappa}_{cl}(s)\,. \end{aligned} \tag{2}$$

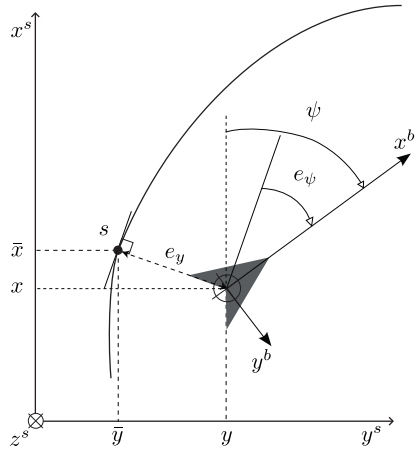

Fig. 2: Local coordinates around the ego lane. The bold triangle indicates the ego vehicle. The solid lines indicate the center-line of the ego lane.

Using the arc-length parametrization, the ego-vehicle's coordinates can be expressed as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \bar{x}(s) \\ \bar{y}(s) \end{bmatrix} + R_z(\bar{\psi}(s)) \begin{bmatrix} 0 \\ e_y \end{bmatrix}, \tag{3}$$

where

$$R_z(\bar{\psi}(s)) = \begin{bmatrix} \cos\bar{\psi}(s) & -\sin\bar{\psi}(s) \\ \sin\bar{\psi}(s) & \cos\bar{\psi}(s) \end{bmatrix}$$

is the rotation matrix transforming vectors from the velocity frame into the inertial frame.

Next, we describe the ego-vehicle position with respect to the (s, e_y) coordinates. Following the calculations in [21], see also [22], we differentiate (3) with respect to time. By employing equations (1) and (2), we obtain:

$$\begin{aligned} \dot{s} &= \frac{v\cos e_\psi}{1 - e_y\bar{\kappa}_{cl}(s)}, \\ \dot{e}_y &= v\sin e_\psi, \\ \dot{e}_\psi &= v\kappa - \bar{\kappa}_{cl}(s)\dot{s}, \\ \dot{v} &= a, \end{aligned} \tag{4}$$

where $e_\psi = \psi - \bar{\psi}$ represents the local heading error. In equation (4) the state and the control vector are denotes as $\boldsymbol{x} = [s, e_y, e_\psi, v]$ and $\boldsymbol{u} = [\kappa, a]$, respectively.

Remark II.1: The inverse mapping $(s, e_y) \mapsto (x, y)$ is well-defined when the ego-vehicle position lies within a tube around the center-line of the ego lane, specifically for $1 - e_y\bar{\kappa}_{cl}(s) > 0$.

C. Model Predictive Control Formulation

In order to address the lane change problem, we set up a parametric nonlinear MPC problem. In this section, we define state-input constraints and formulate the cost function for optimization.

First, we define constraints. The ego-vehicle is required to satisfy the road boundaries, which, in the new coordinate system, take a simple form:

$$e_{y_{min}} \leq e_y \leq e_{y_{max}}. \tag{5}$$

Additionally, we account for the operational limits of the kinematics model and the passenger comfort by imposing state and input constraints on (4) as follows. The velocity is bounded by two constants, i.e.,

$$v_{min} \leq v \leq v_{max}, \tag{6}$$

while the longitudinal acceleration is bounded as follows,

$$a_{min} \leq a \leq a_{max}. \tag{7}$$

Moreover, in order to take into account the limited wheel steer angle, the curvature is bounded in module as follows,

$$|\kappa| \leq \kappa_{max}. \tag{8}$$

We design the cost function J as follows. The ego-vehicle is supposed to travel on the ego lane. In order to execute the lane change maneuver, it needs to minimize the distance from the target lane. Hence, the ego-vehicle needs to stop following the ego lane and start following the target lane. Thus, we design the cost function as a sum of three components as follows:

$$J(\mathbf{x}, \mathbf{u}) = J_{el}(\mathbf{x}, \mathbf{u}) + J_{tl}(\mathbf{x}, \mathbf{u}) + J_u(\mathbf{u}), \tag{9}$$

where J_{el} penalizes deviance from the ego lane and a desired velocity, v^{el}:

$$J_{el} = \gamma(\theta)(q_1 s^2 + q_2 e_y^2 + q_3(v - v^{el})^2);$$

J_{tl} penalizes the deviance from the target lane and its assigned velocity v^{tl}:

$$J_{tl} = (1 - \gamma(\theta))(q_4 s^2 + q_5(e_y - e_y^{tl})^2 + q_6(v - v^{tl})^2);$$

and J_u penalizes the control effort:

$$J_u = r_1\kappa^2 + r_2 a^2.$$

The time-varying switch term $\gamma(\theta)$ is defined as:

$$\gamma(\theta) = \frac{1}{1 + \exp(\alpha(t - \theta))},$$

where $\alpha \in \mathbb{R}^+$ controls the temporal spread, and θ controls the transition between following the ego lane and following the target lane. For $t \leq \theta$, $\gamma \approx 0$, indicating the ego-vehicle should follow the ego lane, while for $t > \theta$, $\gamma \approx 1$, which implies following the target lane.

We are ready to formulate the MPC$_\theta(\boldsymbol{x}, \boldsymbol{u})$ problem:

$$\begin{aligned} \min_{\mathbf{x}(\cdot)\,\mathbf{u}(\cdot)} & \int_0^{t_f} J(\mathbf{x}(\tau), \mathbf{u}(\tau), \theta)\, d\tau + m(\mathbf{x}(t_f)) \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \ \mathbf{x}(0) = \mathbf{x}_0 \\ & h(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \end{aligned} \tag{10}$$

where $t_f > 0$ is the time horizon, $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ represents the nonlinear dynamics (4), $h(\mathbf{x}, \mathbf{u})$ denotes state and input constraints (5), (6), (7), (8), $J(\mathbf{x}, \mathbf{u}, \theta)$ is the stage cost as in (9), and $m(\mathbf{x}(s_f))$ minimizes the square distance between the ego-vehicle state and the desired end state $\mathbf{x}(t_f)$.

It is important to highlight that a key requirement to solve the problem is to determine optimal θ in advance. We determine θ through a probabilistic policy search approach, which will be detailed in the following section.

III. Upper-level Policy Learning

We formalize the problem of selecting the parameter θ of the MPC$_\theta(\boldsymbol{x}, \boldsymbol{u})$ problem, (10), by introducing the concept of learning an upper-level policy $\pi_{\boldsymbol{\omega}}(\theta)$. This selection process can be captured by modeling $\pi_{\boldsymbol{\omega}}(\theta)$ as a Gaussian distribution:

$$\pi_{\boldsymbol{\omega}}(\theta) = \mathcal{N}(\theta|\boldsymbol{\omega}),$$

where $\boldsymbol{\omega} = \left[\mu, \sigma^2\right]$ represent the mean μ and the variance σ^2 of the distribution. In order to address the upper-level policy search problem, we approach it as a probabilistic inference problem, see Fig. 3. To this end, we introduce the "return event" as the observable variable. The probability of observing the return event is expressed as $p(R = 1|\theta) = p(R|\theta)$. Our objective is to find the optimal $\boldsymbol{\omega}$ that maximizes

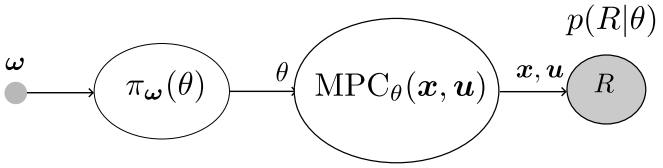

Fig. 3: Graphical representation of upper-lever policy search as a probabilistic inference problem.

the probability of this return event. In other words, we aim to solve the following maximum likelihood problem:

$$\max_{\boldsymbol{\omega}} \ \log p_{\boldsymbol{\omega}}(R) = \log \int_\theta p(R|\theta)\pi_{\boldsymbol{\omega}}(\theta)\ d\theta\,. \tag{11}$$

We solve (11) by using the Monte Carlo Expectation Maximization (MC-EM) algorithm [23], a well-known technique for finding maximum likelihood solutions. Before introducing the algorithm, we need to define the return R by specifying the different contributions that concur to define the task of performing a lane change when necessary.

First, the ego-vehicle needs to avoid collision with obstacles. In order to penalize collisions, we employ a trajectory evaluation mechanism in which we assign negative rewards to collision states. This evaluation is expressed as follows:

$$N_c(\boldsymbol{x}) = \begin{cases} -p_1, & \text{if } c(\mathbf{x}, \mathbf{x}^V) > 0,\ V \in \{FV, LV\} \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where p_1 is a positive constant that weighs the penalty. By assigning negative values to collision states, the ego-vehicle is effectively discouraged from entering such states and encouraged to prioritize safe trajectories that avoid collisions. In (12), the function $c(\mathbf{x}, \mathbf{x}^V)$ evaluates whether a collision occurs between the ego-vehicle and the obstacles. This function is defined by considering an ellipse centered at the axes of the obstacle relative to the new set of coordinates (s, e_y). Specifically, given the positions and future predictions of the obstacles, $\mathbf{x}^V = [s^V, e_y^V]$, we evaluate the collision avoidance as follows,

$$c(\mathbf{x}, \mathbf{x}_o^V) = -1 + \left(\frac{s - s^V}{\bar{s}}\right)^2 + \left(\frac{e_y - e_y^V}{\bar{e}_y}\right)^2, \tag{13}$$

where $\bar{s}$ and $\bar{e}_y$ represents the longitudinal and lateral safety distance, respectively.

Second, in order to discourage unnecessary lane changes, we introduce a contribution penalizing trajectories in which a change in the lateral coordinate e_y occurs. The formulation of this contribution is presented below:

$$N_{lc}(\boldsymbol{x}) = \begin{cases} -p_2, & \text{if } |e_y(t) - e_y(t+1)| > 0 \\ 0, & \text{otherwise} \end{cases}, \tag{14}$$

where p_2 is a positive constant.

Third, in order to discourage trajectories that anticipate changing the lane to avoid the FL, we penalize lateral coordinates e_y that deviate from the ego lane center-line. The lateral displacement penalty is defined as:

$$N_{e_y}(\boldsymbol{x}) = \begin{cases} -p_3, & \text{if } e_y < 0 \\ 0, & \text{otherwise} \end{cases}, \tag{15}$$

where p_3 is a positive constant.

Finally, we can formulate our return as the sum of (12),(14),(15), resulting in the following expression:

$$R(\boldsymbol{x}) = \int_0^{t_f} \left(N_c(\boldsymbol{x}) + N_{lc}(\boldsymbol{x}) + N_{e_y}(\boldsymbol{x})\right) dt\,. \tag{16}$$

Maximizing this return guides the upper-level policy $\pi_{\boldsymbol{\omega}}(\theta)$ to select parameter θ that enable the ego-vehicle to successfully avoid the FV and the LV by performing lane changes when necessary.

We are now ready to describe the MC-EM algorithm. Similar to the standard EM algorithm, we begin by decomposing (11) into two terms with the introduction of a variational distribution $q(\theta)$:

$$\log p_{\boldsymbol{\omega}}(R) = \mathcal{L}_{\boldsymbol{\omega}}(q(\theta)) + KL(q(\theta)||\pi_{\boldsymbol{\omega}}(\theta))\,,$$

where with $KL(\cdot)$ we denote the Kullback-Leibler divergence. Since the KL-divergence is always larger or equal to zero, the term $\mathcal{L}_{\boldsymbol{\omega}}(q(\theta))$ is a lower bound of the log marginal-likelihood $\log p_{\boldsymbol{\omega}}(R)$. The two update steps in EM correspond to maximizing the lower bound $\mathcal{L}$ and minimizing the KL-divergence term. In the MC version of EM algorithm we use a sample-based approximation for the variational distribution $q(\theta)$, i.e., in the E-step, we minimizes the KL-divergence by using samples

$$\theta_i \sim \pi(\theta_i|\boldsymbol{\omega}_k)\,.$$

Then, these samples θ_i are used in the M-step to estimate the expectation of the complete data log-likelihood by maximizing the following objective:

$$\boldsymbol{\omega}_{k+1} = \arg\max_{\boldsymbol{\omega}} \sum_i w_i \log \pi(\theta_i|\boldsymbol{\omega}_k)\,, \tag{17}$$

where $w_i = f(R_i)$ are the wights. In order to transform the return R into an improper probability distribution, we employ the exponential transformation [24]:

$$w_i = \exp(\beta_k(R_i - \max \boldsymbol{R}_k))\,.$$

The parameter β_k serves as the "temperature" of the distribution and can be determined using the following heuristic [25]:

$$\beta_k = \frac{\beta_0}{\max \boldsymbol{R}_k - \min \boldsymbol{R}_k}\,.$$

It is worth noting that higher values of β lead to more greedy policy updates. The MC-EM algorithm iteratively refines the upper-level policy, until convergence is achieved. Convergence is typically indicated when parameter estimates stabilize. The MC-EM algorithm has a closed-form solution for a Gaussian policy. Algorithm 1 gives a pseudocode description of the strategy.

Algorithm 1 Upper-level Policy Learning

Input: initial policy params $\boldsymbol{\omega}_0$,
initial temperature param β_0,
initial traj $(\boldsymbol{x}_0, \boldsymbol{u}_0)$, $k = 0$
repeat
for $i = 1$ to I **do**
Sample θ_i from $\pi(\theta_i|\boldsymbol{\omega}_k)$
Solve $(\boldsymbol{x}, \boldsymbol{u})_i = \text{MPC}_{\theta_i}(\boldsymbol{x}_0, \boldsymbol{u}_0)$
Evaluate $R_i = R(\boldsymbol{x}_i)$
end for
Construct $\boldsymbol{R}_k = [R_1, ..., R_I]$
Calculate weights:
$w_i = \exp(\beta_k(R_i - \max \boldsymbol{R}_k))$, $i = 1$ to I ▷ E-step
Maximize objective:
$\boldsymbol{\omega}_{k+1} \leftarrow \arg\max_{\boldsymbol{\omega}} \sum_i w_i \log \pi(\theta_i|\boldsymbol{\omega}_k)$ ▷ M-step
Update temperature: $\beta_{k+1} \leftarrow \frac{\beta_0}{\max \boldsymbol{R}_k - \min \boldsymbol{R}_k}$
until Convergence criteria met
Output: Optimized policy parameters $\boldsymbol{\omega}^*$

A. Deep Upper-Level Policy

In order to address dynamic scenarios where the environment rapidly changes, we combine the MC-EM algorithm with a self-supervised learning approach to adaptively select the parameter θ based on the environment's observation.

First, we characterize the observation vector. At any given time t, the observation vector, $\mathbf{o}_t$, captures relevant information about the ego-vehicle and surrounding obstacles. Specifically, it includes the ego-vehicle's position and velocity, $\mathbf{o}_t^{ego} = [s, e_y, e_\psi, v]_t$, as well as the positions and velocities of obstacles, $\mathbf{o}_t^V = [s, e_y, e_\psi, v]_t^V$. Formally, the observation $\mathbf{o}_t$ is defined as:

$$\mathbf{o}_t = [\mathbf{o}_t^{ego} - \mathbf{o}_t^{FV}, \mathbf{o}_t^{ego} - \mathbf{o}_t^{LV}] .$$

Second, we collect a dataset $\boldsymbol{D}$ by simulating various scenarios as follows. Each scenario begins with random initial states for the ego-vehicle and the two obstacles. This randomization is crucial as it allows us to explore a wide range of possible situations. Then, we record the observation vector $\boldsymbol{o}_t$. In order to identify the optimal parameter θ_t^* for executing a lane change, we employ the MC-EM algorithm. With the optimal parameter θ_t^* in hand, we solve (10), yielding the first optimal control input, applied to the ego-vehicle. After executing the first optimal control input, we record the next observation $\boldsymbol{o}_{t+1}$. We repeat this process until either the scenario reaches its maximum simulation steps or no collision-free lane change maneuver is found. This systematic approach constructs our dataset $\boldsymbol{D}$, which includes observations and corresponding optimal parameters ($\boldsymbol{o}_t$, θ_t^*) from different scenarios.

Third, we use dataset $\boldsymbol{D}$ to train a general-purpose neural network, denoted as $f_{\boldsymbol{\phi}}$, with $\boldsymbol{\phi}$ representing the network's parameters. We optimize $\boldsymbol{\phi}$ using the following Mean Squared Error (MSE) loss function:

$$\arg\min_{\boldsymbol{\phi}} |f_{\boldsymbol{\phi}}(\boldsymbol{o}_t) - \theta_t^*|^2 ,$$

which minimizes the difference between the neural network's predictions, $f_{\boldsymbol{\phi}}(\boldsymbol{o}_t)$, and the optimal parameter θ_t^* for a given observation $\boldsymbol{o}_t$.

Finally, once the neural network is trained, it can be employed online in the inference phase to handle unseen situations. Given the current observations, the model predicts the optimal parameter to execute a lane change. It is important to note that the MPC control policy (10), used to construct dataset $\boldsymbol{D}$, do not take into account explicitly avoidance constraints. As a consequence, the neural network is trained to provide an upper-level parameter θ even when generating a collision-free lane change is impossible. In order to ensure collision-free maneuvers in the inference phase, the (10) is augmented with avoidance constraints (13), providing safety if the lane change maneuver cannot be executed.

IV. Numerical Computations

In this section, we present numerical computations that demonstrate the effectiveness of the proposed approach.

First, we consider a setup where all vehicles involved start from fixed initial conditions. Given the vehicle's dynamic, we want to plan a trajectory over a fixed time horizon, such that the planned lane-change maneuver is collision-free. To achieve this, we leverage CasADi, an open-source tool for nonlinear optimization and algorithm differentiation, for implementing problem (10). We employ a discretization step of $dt = 0.1\ s$ and a planning horizon $t_f = 10\ s$.

Second, we incorporate observation vectors and train an upper-level policy capable of adaptively generating a lane change maneuver when feasible. We use a simulation time of $t_s = 20\ s$ and apply the same planning settings as in the first setup.

The constraint parameters of problem (10) are based on [20] and on practical driving experience: $v_{min} = 0\ m/s$, $v_{max} = 19.5\ m/s$, $a_{min} = -2.0\ m/s^2$, $a_{max} = 1.5\ m/s^2$, $\kappa_{max} = 0.02\ m^{-1}$, $e_{y_{min}} = -3.75\ m$, $e_{y_{max}} = 1.25\ m$, $\bar{s} = 10\ m$ and $\bar{e}_y = 0.5\ m$.

A. Upper-level Policy for Lane Change Trajectory Generation

The ego-vehicle's initial position is set at $(x_0, y_0) = (80, 0)$, with heading angle $\psi_0 = 0$, and a velocity of $v_0 = 35\ km/h$ (i.e., almost $9.7\ m/s$). The FV obstacle starts at the position $(x^{FV}(0), y^{FV}(0)) = (130, 0)$ and moves along the ego lane at a constant velocity of $v^{FV} = 11\ km/h$ ($3\ m/s$), while the LV obstacle starts at position $(x^{LV}(0), y^{LV}(0)) = (37, 2.5)$ and travels along the target lane with at a constant velocity of $v^{LV} = 30\ km/h$ ($8.3\ m/s$).

In order to provide an example of finding the parameter θ using the MC-EM algorithm, let us consider the learning progress of the upper-level policy $\pi_{\boldsymbol{\omega}}$ using a fixed value of $\beta = 3$, see Fig. 4. As discussed before, the algorithm starts by randomly generating a list of I samples of θ_i. In the example, I is set to 20. Each θ_i is drawn from the upper-level policy $\pi_{\boldsymbol{\omega}}(\theta)$, which is modeled as Gaussian distribution with parameters $\boldsymbol{\omega} = (\mu, \sigma^2)$. In the first iteration, see Fig. 4b,

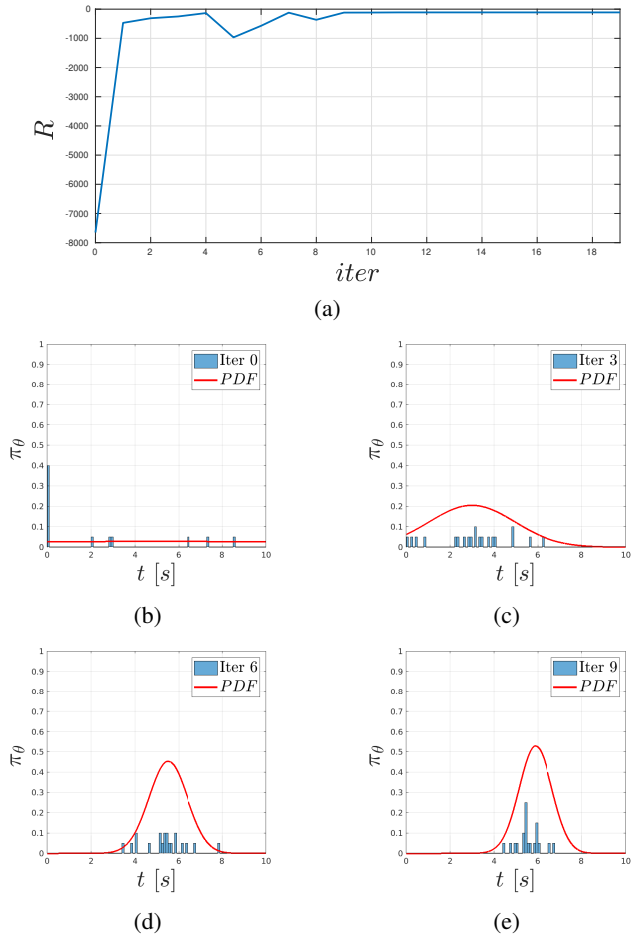

Fig. 4: Learning progress of the upper-level policy. The top sub-figure depicts the return curve with a temperature parameter $\beta = 3.0$, while the bottom sub-figures illustrate the policy distribution at various iteration stages $(0, 3, 6, \text{and } 9)$.

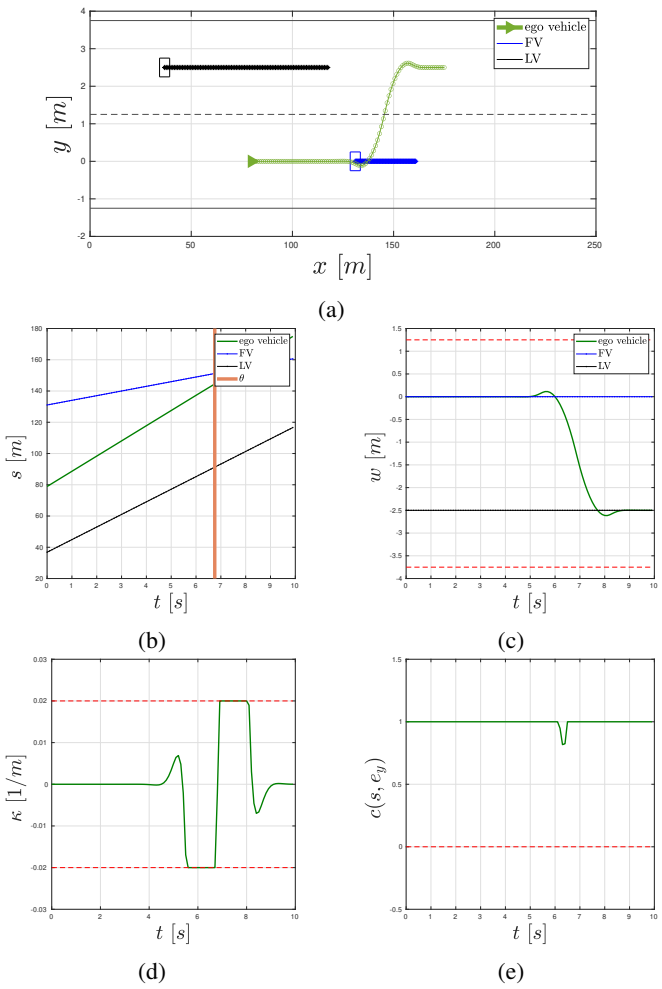

Fig. 5: Lane change maneuver. The ego-vehicle's optimal maneuver (solid green line) is shown, while the maneuvers of the FV and the LV are depicted in dash-dotted blue and black lines, respectively. The optimal switching parameter $\theta^* = 6.7\ s$ is highlighted by the vertical orange line.

the variance σ^2 is set to a large value. This high variance encourages a wide exploration of the θ domain. Then, a set of predicted trajectories $(\boldsymbol{x}, \boldsymbol{u})$ are obtained by solving I optimization problems (10). These trajectories represent different potential lane change maneuvers based on the sampled values of θ_i. The quality of these sampled trajectories is evaluated using the return (16). Next, the parameters $\boldsymbol{\omega}$ (mean and variance) are updated solving (17), see Fig 4c and 4d. This update process is repeated until the return no longer converges, see Fig. 4a.Once convergence is achieved, the policy can be represented by the bell-shaped distribution as the one shown in Fig. 4e (in this case, convergence is achieved after 9 iterations).

The optimal trajectory is depicted in Fig. 5. Next, we highlight some interesting features of the generated trajectory. At first glance, we can identify the planning of a lane change maneuver. The ego-vehicle trajectory transitions from its current lane to the target lane, in order to avoid the FV obstacle, which is moving at a slower velocity. At the same time, it avoids a potential collision with the LV, which is travelling on the target lane (it is worth noting that the collision avoidance constraint is always satisfied, see Fig. 5e). Next, we highlight three distinct phases. Initially, at a considerable distance from the FV, the ego-vehicle maintains alignment with its current lane's center-line, aiming to minimize the cost function J_{el}. During this phase, lateral displacement from the center-line remains nearly zero (see Fig. 5c). Then, at around $t = 5.7\ s$, the cost function J_{tl} is minimized. During this phase, the ego-vehicle steers by applying a negative curvature, see Fig. 5d, and moves towards the center-line of the target lane, see Fig. 5a. Finally, the ego-vehicle approaches the center-line of the target lane and proceeds along it, thus confirming the successful execution of the lane change. We highlight that θ, depicted by the vertical line in Fig. 5b, does not denote the timing for executing a lane change. Instead, it serves as a critical time parameter governing the convex combination of the ego and target cost function.

B. Deep Upper-level Policy for Online Scenarios

Next, we want to find an upper-level policy that enables to select online the parameter for executing a lane change based on the environment observations. To do this, we leverage a combination of MC-EM algorithm and self-supervised learning technique as follows.

In order to construct $\boldsymbol{D}$, we collect $100K$ samples of $(\boldsymbol{o}_t, \theta_t^*)$ pairs. Then, we use $\boldsymbol{D}$ to train a MultiLayer Perceptron (MLP) model. We employ TensorFlow [26], a versatile machine learning framework, to implement the MLP. We chose an architecture that consists of 32 hidden layers, each comprising 32 units. Rectified Linear Unit $(ReLU)$ nonlinearities are applied to these layers to enhance the model's capacity to learn and generalize from the data. Finally, ADAM optimization is then used to update the weights during the backpropagation phase.

We encourage the reader to refer to the video attachment[1] related to the execution of 20 random maneuvers. Next, we highlight some interesting features related to the first two maneuvers of the attached video.

In the first example, depicted in Fig. 6a, we can identify two consecutive lane change maneuvers. The scenario begins with both the ego-vehicle and the FV positioned on the same lane, as illustrated in Fig. 6b. However, there is a significant difference in their velocities: the ego-vehicle and the FV are traveling at $9.7\ m/s$ and at $3.8\ m/s$, respectively, see Fig. 6c. As a result, a decision is made by the deep upper-level policy at time $t = 0\ s$, leading to a lane change maneuver, as depicted in Fig. 6a. After such a lane change, the ego-vehicle is traveling along the target lane. However, the LV is also traveling on the same lane (i.e., the target lane) with a velocity of $6.5\ m/s$. In this situation, the deep upper-level policy once again intervenes by selecting an appropriate θ parameter for the MPC. This decision allows the ego-vehicle to perform another lane change maneuver at about $t = 10\ s$, effectively avoiding a collision with the LV. It is important to highlight that during the execution of this maneuver, the avoidance constraint is always satisfied, see Fig. 6e.

In the second example, illustrated in Fig. 7, we can observe a different behavior compared to the previous case. In this case, the initial positions (see Fig. 7b) and velocities (see Fig. 7c) of the ego-vehicle and the two obstacles do not allow the generation of a collision-free lane change trajectory. Indeed, there is no sufficient time-space gap available in order to perform a safe lane change maneuver. In such a case, in order to keep a safety distance from the FV, the ego-vehicle decreases its velocity by applying a negative acceleration, see Fig. 7d. As expected, also in this case, the safety distance imposed by the avoidance constraint is always satisfied, see Fig.7e.

This last scenario highlights the essential requirement to include hard collision avoidance constraints in our proposed formulation. This constraint becomes particularly crucial in situations where the upper-level policy cannot feasibly determine an appropriate θ to ensure collision-free maneuver.

[1] https://youtu.be/oJjOMCAav7I

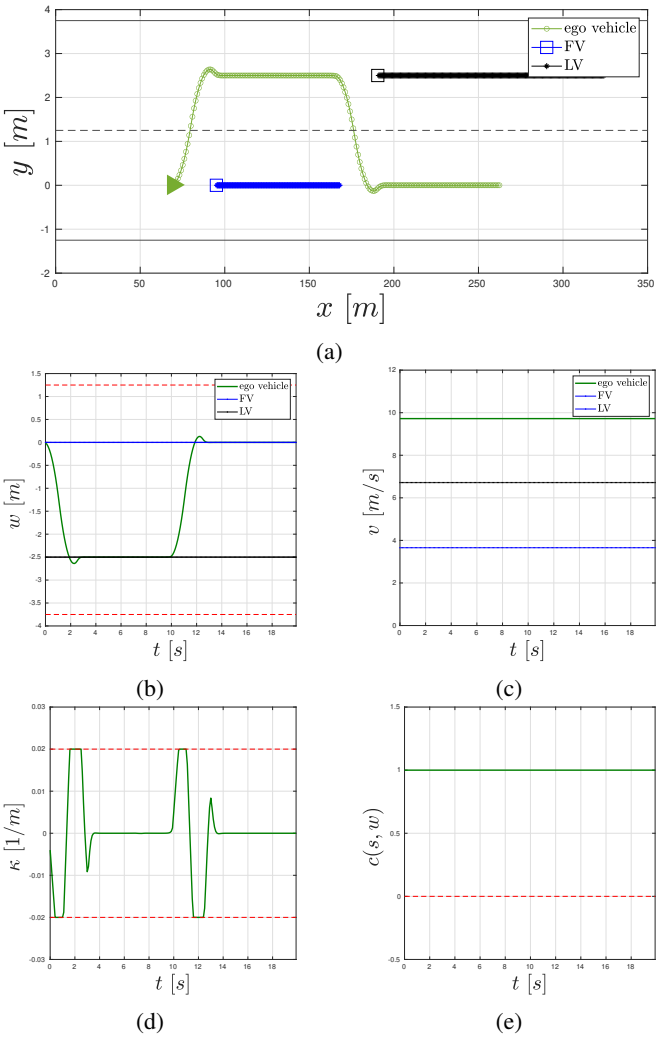

Fig. 6: Double lane change maneuver. The ego-vehicle's optimal maneuver is shown in solid green line. The maneuvers of the FV and the LV are in dash-dotted blue and black lines, respectively.

V. Conclusion

In this paper we have presented a novel approach to tackle the lane change maneuver challenge for Autonomous Vehicles (AVs) by framing it as a parametric Model Predictive Control (MPC) problem. Unlike conventional methods that decouple decision-making and planning, our hybrid approach integrates upper-level policy search with MPC-based low-level policy generation to optimize the lane change strategy effectively. We employed a weighted maximum likelihood approach for policy learning and incorporated self-supervised learning techniques to adapt to dynamic online scenarios, ensuring adaptability to unexpected environmental changes.

While the numerical results presented focus on a straight-line scenario for the sake of presentation, it is important to note that our approach is easily extendable to different types of road layouts. The effectiveness of our method are showcased through these results, indicating its potential to enhance AV maneuvering in dynamic environments.

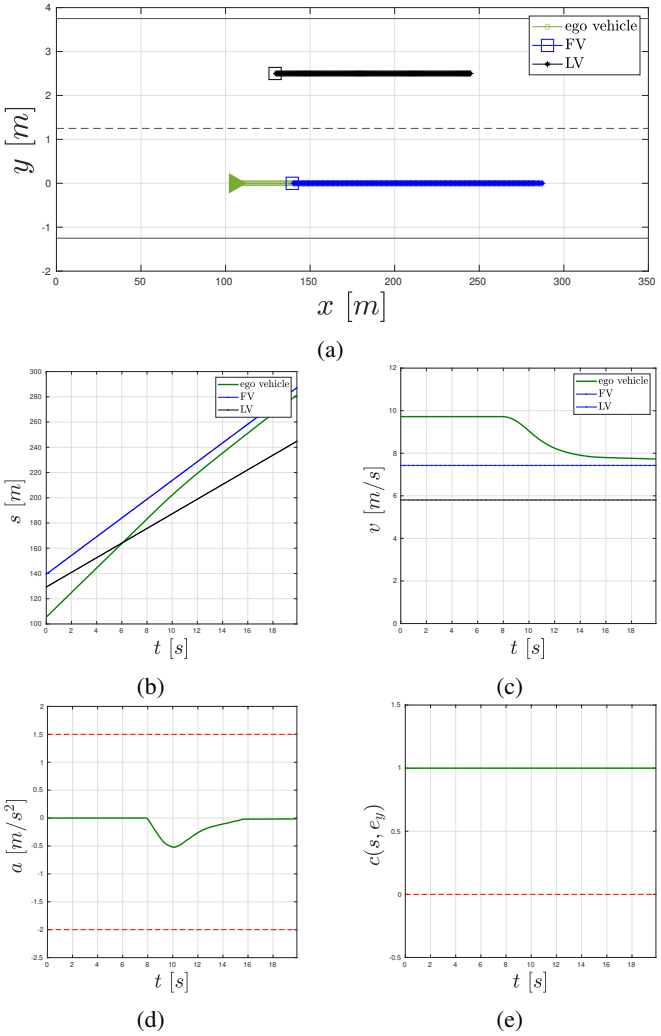

Fig. 7: Longitudinal avoidance maneuver. The ego-vehicle's optimal maneuver is shown in solid green line. The maneuvers of the FV and the LV are in dash-dotted blue and black lines, respectively.

REFERENCES

[1] H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, Y. Yamamoto, A. Yamashita, and H. Asama, "Lane-change detection based on vehicle-trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1109–1116, 2017.

[2] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2016.

[3] F. Laneve, A. Rucco, and M. Bertozzi, "A real-time collision-free maneuver generation algorithm for autonomous driving," *European Journal of Control*, p. 100865, 2023.

[4] ——, "A trajectory optimization strategy for merging maneuvers of autonomous vehicles," in *APCA International Conference on Automatic Control and Soft Computing*. Springer, 2022, pp. 3–14.

[5] C. Hidalgo, R. Lattarulo, J. Pérez, and E. Asua, "Hybrid trajectory planning approach for roundabout merging scenarios," in *2019 IEEE International conference on connected vehicles and expo (ICCVE)*. IEEE, 2019, pp. 1–6.

[6] W. Hu, Z. Deng, D. Cao, B. Zhang, A. Khajepour, L. Zeng, and Y. Wu, "Probabilistic lane-change decision-making and planning for autonomous heavy vehicles," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 12, pp. 2161–2173, 2022.

[7] X. Duan, C. Sun, D. Tian, J. Zhou, and D. Cao, "Cooperative lane-change motion planning for connected and automated vehicle platoons in multi-lane scenarios," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[8] J. Hu, Y. Zhang, and S. Rakheja, "Adaptive lane change trajectory planning scheme for autonomous vehicles under various road frictions and vehicle speeds," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1252–1265, 2022.

[9] S. He, J. Zeng, B. Zhang, and K. Sreenath, "Rule-based safety-critical control design using control barrier functions with application to autonomous lane change," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 178–185.

[10] C. Vallon, Z. Ercan, A. Carvalho, and F. Borrelli, "A machine learning approach for personalized autonomous lane change initiation and control," in *2017 IEEE Intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 1590–1595.

[11] A. P. Aguiar, F. A. Bayer, J. Hauser, A. J. Häusler, G. Notarstefano, A. M. Pascoal, A. Rucco, and A. Saccon, "Constrained optimal motion planning for autonomous vehicles using pronto," *Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles*, pp. 207–226, 2017.

[12] I. Batkovic, M. Zanon, M. Ali, and P. Falcone, "Real-time constrained trajectory planning and vehicle control for proactive autonomous driving with road users," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 256–262.

[13] Z. Li, J. Jiang, and W.-H. Chen, "Automatic lane change maneuver in dynamic environment using model predictive control method," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2384–2389.

[14] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[15] J. Chen, X. Meng, and Z. Li, "Reinforcement learning-based event-triggered model predictive control for autonomous vehicle path following," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3342–3347.

[16] A. Muraleedharan, H. Okuda, and T. Suzuki, "Real-time implementation of randomized model predictive control for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 11–20, 2021.

[17] F. S. Acerbo, H. Van der Auweraer, and T. D. Son, "Safe and computational efficient imitation learning for autonomous vehicle driving," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 647–652.

[18] Y. Song and D. Scaramuzza, "Learning high-level policies for model predictive control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7629–7636.

[19] ——, "Policy search for model predictive control with application to agile drone flight," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2114–2130, 2022.

[20] 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.

[21] F. Bayer and J. Hauser, "Trajectory optimization for vehicles in a constrained environment," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 5625–5630.

[22] A. Rucco, G. Notarstefano, and J. Hauser, "An efficient minimum-time trajectory generation strategy for two-track car vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1505–1519, 2015.

[23] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Advances in neural information processing systems*, vol. 21, 2008.

[24] J. Peters and S. Schaal, "Applying the episodic natural actor-critic architecture to motor primitive learning." in *ESANN*, 2007, pp. 295–300.

[25] G. Neumann and J. Peters, "Fitted q-iteration by advantage weighted regression," *Advances in neural information processing systems*, vol. 21, 2008.

[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.