Self-Adaptive Driving in Nonstationary Environments through Conjectural Online Lookahead Adaptation

Tao Li,1∗ Haozhe Lei,1∗ and Quanyan Zhu1

Abstract—Powered by deep representation learning, reinforcement learning (RL) provides an end-to-end learning framework capable of solving self-driving (SD) tasks without manual designs. However, time-varying nonstationary environments cause proficient but specialized RL policies to fail at execution time. For example, an RL-based SD policy trained under sunny days does not generalize well to rainy weather. Even though meta learning enables the RL agent to adapt to new tasks/environments, its offline operation fails to equip the agent with online adaptation ability when facing nonstationary environments. This work proposes an online meta reinforcement learning algorithm based on the conjectural online lookahead adaptation (COLA). COLA determines the online adaptation at every step by maximizing the agent’s conjecture of the future performance in a lookahead horizon. Experimental results demonstrate that under dynamically changing weather and lighting conditions, the COLA-based self-adaptive driving outperforms the baseline policies regarding online adaptability. A demo video, source code, and appendixes are available at https://github.com/Panshark/COLA

I. INTRODUCTION

Recent breakthroughs from machine learning [1]–[3] have spurred wide interest and explorations in learning-based self driving (SD) [4]. Among all the endeavors, end-to-end reinforcement learning [5] has attracted particular attention. Unlike modularized approaches, where different modules handle perception, localization, decision-making, and motion control, end-to-end learning approaches aim to output a synthesized driving policy from raw sensor data.

However, the limited generalization ability prevents RL from wide application in real SD systems. To obtain a satisfying driving policy, RL methods such as Q-learning and its variants [2], [6] or policy-based ones [7], [8] require an offline training process. Training is performed in advance in a stationary environment, producing a policy that can be used to make decisions at execution time in the same environments as seen during training. However, the assumption that the agent interacts with the same stationary environment as training time is often violated in practical problems. Unexpected perturbations from the nonstationary environments pose a great challenge to existing RL approaches, as the trained policy does not generalize to new environments [9].

To elaborate on the limited generalization issue, we consider the vision-based lane-keeping task under changing weather conditions shown in Figure 1. The agent needs to provide automatic driving controls (e.g., throttling, braking, and steering) to keep a vehicle in its travel lane, using only images from the front-facing camera as input. The driving testbed is built on the CARLA platform [10]. As shown in Figure 1, different weather conditions create significant visual differences, and a vision-based policy learned under a single weather condition may not generalize to other conditions. As later observed in one experiment [see Figure 3a], a vision-based SD policy trained under the cloudy daytime condition does not generalize to the rainy condition. The trained policy relies on the solid yellow lines in the camera images to guide the vehicle. However, such a policy fails on a rainy day when the lines are barely visible.

Fig. 1: An illustration of conjectural online lookahead adaptation. When driving in a changing environment, the agent first uses a residual neural network and bayesian filtering to calibrate its belief at every time step about the hidden mode. Based on its belief, the agent conjectures its performance in the future within a lookahead horizon. The policy is adapted through conjectural lookahead optimization, leading to a suboptimal (empirically) online control.

The challenge of limited generalization capabilities has motivated various research efforts. In particular, as a learning-to-learn approach, meta learning [11] stands out as one of the well-accepted frameworks for designing fast adaptation strategies. Note that the current meta learning approaches primarily operate in an offline manner. For example, in model-agnostic meta learning (MAML) [12], the meta policy needs to be first trained in advance. When adapting

∗The authors contributed equally to this work.
1The authors are with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY, 11201, USA. {tl2636, hl14155, qz494}@nyu.edu. This work is partially supported by grant ECCS-1847056 from National Science Foundation (NSF).
the meta policy to the new environment in the execution
time, the agent needs a batch of sample trajectories to
compute the policy gradients to update the policy. However,
when interacting with a nonstationary environment online,
the agent has no access to prior knowledge or batches of
sample data. Hence, it is unable to learn a meta policy before-
hand. In addition, past experiences only reveal information
about previous environments. The gradient updates based
on past observations may not suffice to prepare the agent
for the future. In summary, the challenge mentioned above
persists when the RL agent interacts with a nonstationary
environment in an online manner.

Our Contributions To address the challenge of limited
online adaptation ability, this work proposes an online meta
reinforcement learning algorithm based on conjectural online
lookahead adaptation (COLA). Unlike previous meta RL
formulations focusing on policy learning, COLA is con-
cerned with learning meta adaptation strategies online in
a nonstationary environment. We refer to this novel learn-
ing paradigm as online meta adaptation learning (OMAL).
Specifically, COLA determines the adaptation mapping at
every step by maximizing the agent’s conjecture of the future
performance in a lookahead horizon. This lookahead
optimization is approximately solved using off-policy data,
achieving real-time adaptation. A schematic illustration of
the proposed COLA algorithm is provided in Figure 1.

In summary, the main contributions of this work include
that 1) we formulate the problem of learning adaptation
strategies online in a nonstationary environment as online
meta-adaptation learning (OMAL); 2) we develop a real-
time OMAL algorithm based on conjectural online lookahead
adaptation (COLA); 3) experiments show that COLA equips
the self-driving agent with online adaptability, leading to self-
adaptive driving under dynamic weather.

II. SELF-DRIVING IN NONSTATIONARY ENVIRONMENTS:
MODELING AND CHALLENGES

We model the lane-keeping task under time-varying
weather conditions shown in Figure 2 as a Hidden-Mode
Markov Decision Process (HM-MDP) [13]. The state input
at time $t$ is a low-resolution image, denoted by $s_t \in S$. Based
on the state input, the agent employs a control action $a_t \in \mathcal{A}$,
including acceleration, braking, and steering. The discrete
control commands used in our experiment can be found in
Appendix Section III. The driving performance at $s_t$ when
taking $a_t$ is measured by a reward function $r_t = r(s_t, a_t)$.

Upon receiving the control commands, the vehicle changes
its pose/motion and moves to the next position. The new
surrounding traffic conditions captured by the camera
serve as the new state input subject to a transition kernel
$P(s_{t+1}|s_t, a_t; z_t)$, where $z_t \in \mathcal{Z}$ is the environment mode or
latent variable hidden from the agent, corresponding to the
weather condition at time $t$. The transition $P(s_{t+1}|s_t, a_t; z_t)$
tells how likely the agent is to observe a certain image $s_{t+1}$
under the current weather conditions $z_t$. CARLA simulator
controls the weather condition through three weather pa-
rameters: cloudiness, rain, and puddles, and $z_t$ is a three-
dimensional vector with entries being the values of these
parameters.

In HM-MDP, the hidden mode shifts stochastically ac-
cording to a Markov chain $p_z(z_{t+1}|z_t)$ with initial distribu-
tion $p_z(z_1)$. As detailed in the Appendix Section III, the
hidden mode (weather condition) shifts in our experiment
are realized by varying three weather parameters subject to
a periodic function. One realization example is provided in
Figure 2. Note that the hidden mode $z_t$, as its name
suggested, is not observable in the decision-making process.

Let $I_t = \{s_t, a_{t-1}, r_{t-1}\}$ be the set of agent’s observations
at time $t$, referred to as the information structure [14]. Then,
the agent’s policy $\pi$ is a mapping from the past observations
$\cup_{k=1}^t I_k$ to a distribution over the action set $\Delta(A)$.

a) Reinforcement Learning and Meta Learning: Stan-
dard RL concerns stationary MDP, where the mode remains
unchanged (i.e., $z_t = z$) for the whole decision horizon $H$.

Due to stationarity, one can search for the optimal policy
within the class of Markov policies [15], where the policy
$\pi: \mathcal{S} \to \Delta(A)$ only depends on the current state input.

This work considers a neural network policy $\pi(s_t, a_t; \theta), \theta \in \Theta \subset \mathbb{R}^d$ as the state inputs are high-dimensional images. The
RL problem for the stationary MDP (fixing $z$) is to find an
optimal policy maximizing the expected cumulative rewards
discounted by $\gamma \in (0, 1]$:

$$\max_{\theta} J_z(\theta) := \mathbb{E}_{\rho_\theta(\cdot|z)} \sum_{t=1}^H \gamma^{t-1} r(s_t, a_t).$$ (1)

We employ the policy gradient method to solve
for the maximization using sample trajectories $\tau =
(s_1, a_1, \ldots, s_H, a_H)$. The idea is to apply gradient descent
with respect to the objective function $J_z(\theta)$. Following
the policy gradient theorem [7], we obtain $\nabla J_z(\theta) =
\mathbb{E}[g(\tau; \theta)]$, where $g(\tau; \theta) = \sum_{t=1}^H \nabla_{\theta} \log \pi(a_t|s_t; \theta) R^\theta(\tau)$, and
$R^\theta(\tau) = \sum_{t=1}^H r(s_t, a_t)$. In RL practice, the policy
gradient $\nabla J_z(\theta)$ is replaced by its MC estimation since
evaluating the exact value is intractable. Given a batch of
trajectories $D = \{\tau\}$ under the policy $\pi(\cdot|\cdot; \theta)$, the MC
estimation is $\hat{\nabla} J(\theta, D(\theta)) := 1/|D| \sum_{\tau \in D(\theta)} g(\tau; \theta)$. Our
implementation applies the actor-critic (AC) method [16], a
variant of the policy gradient, where the cumulative reward $R^h(\tau)$ in $g(\tau; \theta)$ is replaced with a Q-value estimator (critic) [16]. The AC implementation is in Appendix Section IV.

To improve the adaptability of RL policies, meta reinforcement learning (meta RL) aims to find a meta policy $\theta$ and an adaptation mapping $\Phi$ that returns satisfying rewards in a range of environments when the meta policy $\theta$ is updated using $\Phi$ within each environment. Mathematically, meta RL amounts to the following optimization problem [17]:

$$
\max_{\theta, \Phi} \mathbb{E}_{z \sim \rho_z} [J_z(\Phi(\theta, D_z))],
$$

where the initial distribution $\rho_z$ gives the probability that the agent is placed in the environment $z$. $D_z$ denotes a collection of sample trajectories under environment $z$ rolled out under $\theta$. The adaptation mapping $\Phi(\theta, D_z)$ adapts the meta policy $\theta$ to a new policy $\theta_z$ fine-tuned for the specific environment $z$ using $D_z$. For example, the adaptation mapping in MAML [12] is defined by a policy-gradient update, i.e., $\Phi(\theta, D_z) = \theta + \alpha \nabla J(\theta, D_z)$, where $\alpha$ is the step size to be optimized [18]. Since the meta policy maximizes the average performance over a family of tasks, it provides a decent starting point for fine-tuning that requires far less data than training $\theta_z$ from scratch. As a result, the meta policy generalizes to a collection of tasks using sample trajectories.

b) Challenges in Online Adaptation: Meta RL formulation reviewed above does not fully address the limited adaptation issue in execution time since (2) does not include the evolution of the hidden mode $z_{t+1} \sim p_z(\cdot|z_t)$. When interacting with a nonstationary environment online, the agent cannot collect enough trajectories $D_{z_t}$ for adaptation as $z_t$ is time-varying. In addition, past experiences $D_z$ only reveal information about the previous environments. The gradient updates based on past observations may not suffice to prepare the agent for the future.

III. CONJUNCTURAL ONLINE LOOKAHEAD ADAPTATION

Instead of casting meta learning as a static optimization problem in (2), we consider learning the meta-adaptation online, where the agent updates its adaptation strategies at every step based on its observations. Unlike previous works [17], [19], [20] aiming at learning a meta policy offline, our meta learning approach enables the agent to adapt to the changing environment. The following formally defines the online meta-adaptation learning (OMAL) problem.

Unlike (2), the online adaptation mapping relies on online observations $\cup_{k=1}^h D_k$ instead of a batch of samples $D_z$: the meta adaptation at time $t$ is defined as a mapping $\Phi_t(\theta) = \Phi(\theta, \cup_{k=1}^t D_k)$. Let $r_t^z(s_t; \theta) := \mathbb{E}_{\rho_z} [r_t(s_t, a_t)|s_t, \theta]$ be the expected reward. The online meta-adaptation learning in the HM-MDP is given by

$$
\max_{\{\Phi_t\}_{t=1}^H} \mathbb{E}_{z_1, z_2, \ldots, z_H} \sum_{t=1}^H r_t^z(s_t; \Phi_t(\theta))], \quad \text{(OMAL)}
$$

where $z_{t+1} \sim p_z(\cdot|z_t), t = 1, \ldots, H - 1, \theta = \arg \max \mathbb{E}_{z \sim \rho_z} [J_z(\theta)]$. Some remarks on (OMAL) are in order. First, similar to (1) and (2), (OMAL) needs to be solved in a model-free manner as the transition kernel $P(\cdot|z_t)$ regarding image inputs is too complicated to be modeled or estimated. Meanwhile, the hidden mode transition $p_z$ is also unknown to the agent. Second, the meta policy in (OMAL) is obtained in a similar vein as in (2): $\theta$ maximizes the rewards across different environments, providing a decent base policy to be adapted later (see Algorithm 2 in Appendix). Unlike the offline operation in (2) (i.e., finding $\theta$ and $\Phi$ before execution), (OMAL) determines $\theta_t$ on the fly in the execution phase based on $\cup_{k=1}^t D_k$ to accommodate the nonstationary environment.

Theoretically, searching for the optimal meta adaptation mappings $\{\Phi_t\}_{t=1}^H$ is equivalent to finding the optimal nonstationary policy $\{\theta_t\}_{t=1}^H$ in the HM-MDP, i.e.,

$$
\max_{\theta_t} \mathbb{E}_{z_1, z_2, \ldots, z_H} \sum_{t=1}^H r_t^z(s_t; \theta_t). \quad \text{(3)}
$$

However, solving for (3) either requires domain knowledge regarding $P(\cdot|z)$ and $p_z$ [13], [21], [22] or black-box simulators producing successor states, observations, and rewards [23]. When solving complex tasks such as the self-driving task considered in this work, these assumptions are no longer valid. The following presents a model-free real-time adaptation algorithm, where $\Phi_t$ is determined by a lookahead optimization condition on the belief of the hidden mode.

The meta policy in (OMAL) is given by $\theta = \arg \max \mathbb{E}_{z \sim \rho_z} [J_z(\theta)]$. Note that $z \in Z$ in our case is a three-dimensional vector composed of three weather parameters. The hidden model space $Z$ contains an infinite number of $z$, and it is impractical to train the policy over each environment. Similar to MAML [12], we sample a batch of modes $z$ using $p_z$ and train the meta policy over these sampled modes, as shown in Algorithm 2 in the Appendix. Note that the meta training program returns the stabilized policy iterate $\theta_k$, its corresponding mode sample batch $\hat{Z}$, and the associated sample trajectories $D_k \{\hat{D}_z\}$ under mode $z$ and policy $\theta_k$. These outputs can be viewed as the agent’s knowledge gained in the training phase and are later utilized in the online adaptation process.

A. Lookahead Adaptation

In online execution, the agent forms a belief $b_t \in \Delta(\hat{Z})$ at time $t$ on the hidden mode based on its past observations $\cup_{k=1}^t \hat{D}_k$. Note that the support of the agent’s belief is $\hat{Z}$ instead of the whole space $Z$. The intuition is the agent always attributes the current weather pattern to a mixture of known patterns when facing unseen weather conditions. Then, the agent adapts to this new environment using its training experiences. Specifically, the agent conjectures that the environment would evolve according to $P(\cdot|z)$ with probability $b_t(z)$ for a short horizon $K$. Given the agent’s belief $b_t$ and its meta policy $\theta$, the trajectory of future $K$ steps $\pi^K := (s_t, a_t, \ldots, s_{t+K-1}, a_{t+K-1}, s_{t+K})$ follows $q(\tau^K; b, \theta)$ defined as

$$
\prod_{k=0}^{K-1} \pi(a_{t+k}|s_{t+k}; \theta) \prod_{k=0}^{K-1} \sum_{z \in Z} b_t(z) P(s_{t+k+1}|s_{t+k}, a_{t+k}|z).
$$

7207
In order to maximize the forecast of the future performance in K steps, the adapted policy \( \theta_t = \Phi_t(\theta) \) should maximize 
\[
\max_{\theta' \in \Theta} \mathbb{E}_{q_t(b; \theta')} \prod_{k=0}^{K-1} \pi(a_{t+k} | s_{t+k}; \theta') \sum_{k=0}^{K-1} r(s_{t+k}, a_{t+k})
\]
subject to 
\[
\mathbb{E}_{s \sim q} D_{KL}(\pi(\cdot | s; \theta'), \pi(\cdot | s; \theta'')) \leq \delta,
\]
where \( D_{KL} \) is the Kullback-Leibler divergence. In the KL divergence constraint, we slightly abuse the notation \( q(\cdot) \) to denote the discounted state visiting frequency \( s \sim q \).

Note that the objective function in (CLO) is equivalent to the future performance under \( \theta' \):
\[
\mathbb{E}_{q_t(r; b; \theta')} \prod_{k=0}^{K-1} \pi(a_{t+k} | s_{t+k}; \theta') \sum_{k=0}^{K-1} r(s_{t+k}, a_{t+k})
\]
This is because the distribution shift between \( q(\tau^K_t; b; \theta') \) and \( q(\tau^K_t; b; \theta) \) in (CLO) is compensated by the ratio \( \prod_{k=0}^{K-1} \pi(a_{t+k} | s_{t+k}; \theta') / \pi(a_{t+k} | s_{t+k}; \theta) \).
The intuition is that the expectation in (CLO) can be approximated using \( D_z(\theta) \) collected in the meta training. When \( \theta' \) is close to the base policy \( \theta \) in terms of KL divergence, the estimated objective in (CLO) using sample trajectories under \( \theta \) returns a good approximation to
\[
\mathbb{E}_{q_t(r; b; \theta')} \prod_{k=0}^{K-1} r(s_{t+k}, a_{t+k})
\]
This approximation is detailed in the following subsection.

B. Off-policy Approximation

We first simplify (CLO) by linearizing the objective and approximating the constraints with a quadratic surrogate as introduced in [25], [26]. Denote by \( L(\theta'; \theta) \) the objective function in (CLO). A first-order linearization of the objective at \( \theta' = \theta \) is given by
\[
\nabla_{\theta'} L(\theta'; \theta) |_{\theta'} = 0
\]
Note that \( \theta' \) is the decision variable, and \( \theta \) is the known meta policy. The gradient \( \nabla_{\theta'} L(\theta'; \theta) |_{\theta'} = 0 \) is exactly the policy gradient under the trajectory distribution \( q(\cdot; b_1, \theta) \). Using the notions introduced in Section II, we obtain
\[
\nabla_{\theta'} L(\theta'; \theta) |_{\theta'} = 0 = \mathbb{E}_{q_t(b; \theta')} [g(\tau^K_t; \theta)] = \sum_{z \in \mathbb{Z}} b(z) \mathbb{E}_{q_t(z; \theta')} [g(\tau^K_t; \theta)].
\]
Note that \( \tau^K_t \) is the future K-step trajectory, which is not available to the agent at time \( t \), and a substitute is its counterpart in \( D_z \): the sample trajectory within the same time frame \( [t, t + K] \) in meta training. Denote this counterpart by \( \hat{\tau}^K_t \), then
\[
\mathbb{E}_{q_t(b; \theta')} [g(\tau^K_t; \theta)] \approx \sum_{z \in \mathbb{Z}} b(z) \mathbb{E}_{q_t(z; \theta')} [g(\hat{\tau}^K_t; \theta)],
\]
and we denote its sample estimate by \( \hat{g}(b; \theta') \). The distribution \( q(\cdot; z, \theta) \) is defined similarly as \( q(\cdot; b, \theta) \). More details on this distribution and \( \hat{g}(b; \theta') \) are in Appendix Section II.

We consider a second-order approximate for the KL-divergence constraint because the gradient of \( D_{KL} \) at \( \theta \) is zero. In this case, the approximated constraint is
\[
\frac{1}{2} (\theta - \theta')^T A (\theta - \theta') \leq \delta,
\]
where \( A_{ij} = \partial^2 L_{ij} / \partial \theta_i \partial \theta_j \). Let \( \hat{A} \) be the sample estimate using \( \hat{\tau}^K_t \) (see Appendix Section II), then the off-policy approximate to (CLO) is given by
\[
\max_{\theta'} \hat{g}(b; \theta')(\theta' - \theta)
\]
subject to 
\[
\frac{1}{2} (\theta - \theta')^T \hat{A} (\theta - \theta') \leq \delta.
\]

To find such \( \theta' \), we compute the search direction \( d\theta = \hat{A}^{-1} \hat{g} \) efficiently using conjugate gradient method as in original TRPO implementation [25]. Generally, the step size is determined by backtracking line search, yet we find that fixed step size also achieves impressive results in experiments.

C. Belief Calibration

The last piece in our online adaptation learning is belief calibration: how the agent infers the currently hidden mode \( z_2 \) based on past experiences and updates its belief based on new observations. A general-purpose strategy is to train an inference network through a variational inference approach [19]. Considering the significant visual differences caused by weather conditions, we adopt a more straightforward approach based on image classification and Bayesian filtering. To simplify the exposition, we only consider two kinds of weather: cloudy (denoted the mode by \( z_2 \)) and rainy (denoted by \( z_1 \)) in meta training. \( \hat{Z} = \{z_1, z_2\} \).

a) Image Classification: The image classifier is a binary classifier based on ResNet architecture [27], which is trained using cross-entropy loss. The data sets (training, validation, and testing) include RGB camera images generated from CARLA. The input is the current camera image, and the output is the probability of that image being of the rainy type. The training details are included in Appendix Section IV.

Given an arbitrary image, the underlying true label is created by measuring its corresponding parameter distance to the default weather setup in the CARLA simulator (see Appendix Section IV). Finally, we remark that the true labels and the hidden mode \( z_2, z_3 \) are revealed to the agent in the training phase. Yet, in the online execution, only online observations \( I_t = \{s_t, a_{t-1}, r_{t-1}\} \) are available.

b) Bayesian Filtering: Denote the trained classifier by \( f : \mathcal{S} \rightarrow \Delta(\hat{Z}) \), where \( f(z; s) \) denotes the output probability of \( s \) being an image input under the mode \( z \). During the online execution phase, given an state input \( s_t \) and the previous belief \( b_{t-1} \in \Delta(\hat{Z}) \), the updated belief is
\[
b_t(z_r) = \frac{b_{t-1}(z_r) f(z_r; s_t)}{\sum_{z \in \hat{Z}} b_{t-1}(z) f(z; s_t)}.
\]

The intuition behind (5) is that it better captures the temporal correlation among image sequences than the vanilla classifier \( f \). Note that this continuously changing weather is realized through varying three weather parameters. Hence, two adjacent images, as shown in Figure 2, display a temporal correlation: what follows a rain image is highly likely to be another rain image. Considering this correlation, we use
Bayesian filtering, where the prior reveals information about previous images. As shown in an ablation study in Section IV-B, (5) does increase the belief accuracy (to be defined later in Section IV-B) and associated rewards. Finally, we conclude this section with the pseudocode of the proposed conjectural online lookahead adaptation (COLA) algorithm presented in Algorithm 1.

### Algorithm 1 Conjectural Online Lookahead Adaptation

**Input** The meta policy \( \theta \), classifier \( f \), training samples \( \{D_z\}_{z \in \mathcal{Z}} \), sample batch size \( M \), lookahead horizon length \( K \), and step size \( \alpha \). Set \( \theta_1 = \theta \).

for \( t \in \{1, 2, \ldots, H\} \) do

- Obtain the image input \( s_t \) from the camera;
- Implement the action using \( \pi(\cdot|s_t; \theta_t) \);
- Obtain the probability \( f(z|s_t) \) from the classifier;
- Update the belief using (5);
- Sample \( M \) trajectories \( z^K_1 \) under \( z \) from \( \{D_z\}_{z \in \mathcal{Z}} \);
- Obtain \( \theta' \) by solving (4), and let \( \theta_{t+1} = \theta' \);

end

IV. EXPERIMENTS

This section evaluates the proposed COLA algorithm using CARLA simulator [10]. The HM-MDP setup for the lane-keeping task (i.e., hidden mode transition, the reward function) is presented in Appendix Section III. This section reports experimental results under \( K = 10 \), and experiments under other choices are included in Appendix Section V.

We compare COLA with the following RL baselines. 1) The base policy \( \theta_{\text{base}} \): a stationary policy trained under dynamic weather conditions, i.e., \( \theta_{\text{base}} = \arg\max_{\tau} \mathbb{E} [r_{\tau}(s_t; \theta)] \). 2) The MAML policy \( \theta_{\text{MAML}} \): the solution to (2) with \( \Phi \) being the policy gradient computed using the past \( K \)-step trajectory. 3) The optimal policy under the mixture of \( Q \) functions \( \theta_{Q_{\text{mix}}} \): the action at each time step is determined by \( a_t = \arg\max_{a \in \mathcal{A}} \sum_{z \in \mathcal{Z}} b_t(z)Q_z(s_t, a) \), where \( Q_z \) is the function trained under the stationary MDP with fixed mode \( z \). In our experiment, this mixture \( \sum_{z \in \mathcal{Z}} b_t(z)Q_z(s_t, a) \) is composed of two \( Q \) functions \( Q_c, Q_r \) (sharing the same network structure as the critic) under the cloudy and rainy conditions, respectively. 4) The oracle policy \( \theta_{\text{oracle}} \): an approximate solution to (3), obtained by solving (CLO) using authentic future trajectories generated by the simulator.

The experimental results are summarized in Figure 3. As shown in Figure 3b, the COLA policy outperforms \( \theta_{\text{base}} \), \( \theta_{\text{MAML}} \), and \( \theta_{Q_{\text{mix}}} \), showing that the lookahead optimization (CLO) better prepares the agent for the future environment changes. Note that \( \theta_{Q_{\text{mix}}} \) returns the worst outcome in our experiment, suggesting that the naive adaptation by averaging \( Q \) function does not work: the nonstationary environment given by HM-MDP does not equal an average of multiple stationary MDPs. The agent needs to consider the temporal correlation as discussed in Section III-C.

To evaluate the effectiveness of COLA-based online adaptation, we follow the regret notion used in online learning [28] and consider the performance differences between COLA (the other three baselines) and the oracle policy. The performance difference or regret up to time \( T \) is defined as \( \text{Reg}(T) = \mathbb{E} [\sum_{t=1}^{T} r^c(s_t; \theta_{\text{oracle}}) - \sum_{t=1}^{T} r^c(s_t; \Phi_t(\theta))] \). If the regret grows sublinearly, i.e., \( \text{Reg}(T) < O(T) \), then the corresponding policy achieves the same performance as the oracle asymptotically. As shown in Section IV, the proposed COLA is the only algorithm that achieves comparable results as the oracle policy.

![Figure 3: Evaluations of COLA algorithms in the lane-keeping task under nonstationary weather conditions (averaged over 500 episodes under 10 random seeds).](image)

(a) The average rewards of the cloudy-trained the rainy-trained policies in cloudy and rainy conditions. The policy trained under one environment does not generalize to the other. (b) The average rewards of the COLA policy and the four baseline policies. COLA achieves comparable results as the oracle policy and outperforms the other baselines. (c) The regret grows under different policies. Only COLA policy achieves sublinear regret growth [below the linear bound \( O(T) \)]. (d) The evolution of transient accuracies (averaged over 500 episodes) with respect to time step \( t \) under Bayesian filtering and vanilla classification in dynamic weather.

A. Knowledge Transfer through Off-policy Gradients

This subsection presents an empirical explanation of COLA’s adaptability: the knowledge regarding the speed control learned in meta training is carried over to online execution through the off-policy gradient \( \hat{g} \) in (4). As observed in Table II in Appendix, when driving in a rainy environment (i.e., \( z_r \)), the agent tends to drive slowly to avoid crossing the yellow line, as the visibility is limited. In contrast, it increases the cruise speed when the rain stops (i.e., \( z_c \)). Denote the off-policy gradients under \( z_r, z_c \) by \( \hat{g}_r \) and \( \hat{g}_c \), respectively. Then, solving (4) using \( \hat{g}_r \) (\( \hat{g}_c \)) leads to a conservative (aggressive) driving in terms of speeds as shown in Figure 4a (Figure 4b), regardless of the environment changes. This result suggests that the knowledge gained in meta training is encoded into the off-policy gradients and...
Fig. 4: Figure (a) [or (b)] presents average speeds at each time step within the horizon (average of 500 episodes) with $g_c$ (red curves) and $g_r$ (blue curves) being used to adapt the meta policy (grey curves) under the cloudy [or rainy in (b)] mode. Using $g_c$ [or $g_r$] always leads to aggressive [or conservative] driving, regardless of the environment.

is later retrieved according to the belief. Hence, the success of COLA relies on a decent belief calibration, where the belief accurately reveals the underlying mode. Section IV-B justifies our choice of Bayesian filtering.

B. An Ablation Study on Bayesian Filtering

For simplicity, we use the 0-1 loss to characterize the misclassification in the online implementation in this ablation study. Denote by $i_{\nu t} \in \{0, 1\}$ the label returned by the classifier (i.e., the class assigned with a higher probability; 1 denotes the rain class) at time $t \in \{1, \ldots, H\}$ in the $\nu$-th episode (500 episodes in total). Then the loss at time $t$ is $1_{\{i_{\nu t} \neq \hat{i}_{\nu t}\}}$, where $\hat{i}_{\nu t}$ is the true label. We define the transient accuracy at time $t$ as Transient Accuracy $= \frac{\sum_{\nu=1}^{500} 1_{\{i_{\nu t} \neq \hat{i}_{\nu t}\}}}{\sum_{\nu=1}^{500}}$, which reflects the accuracy of the vanilla classifier at time $t$. The average transient accuracy over the whole episode horizon $H$ is defined as the episodic accuracy, i.e., Episodic Accuracy $= \frac{\sum_{\nu=1}^{500} \sum_{t=1}^{H} 1_{\{i_{\nu t} \neq \hat{i}_{\nu t}\}}}{500H}$. These accuracy metrics can also be extended to the Bayesian filter, where the outcome label corresponds to the class with a higher probability.

We compare the transient and episodic accuracy as well as the average rewards under Bayesian filtering and vanilla classification, and the results are summarized in Table I and Figure 3d. Even though the two methods’ episodic accuracy under two methods differ litter, as shown in Table I, the difference regarding the average rewards indicates that Bayesian filtering is better suited to handle nonstationary weather conditions. As shown in Figure 3d, Bayesian filtering outperforms the classifier for most of the horizon in terms of transient accuracy. The reason behind its success is that Bayesian filtering better captures the temporal correlation among the image sequence and returns more accurate predictions than the vanilla classifier at every step, as the prior distribution is incorporated into the current belief. It is anticipated that higher transient accuracy leads to higher mean rewards since the lookahead optimization [see (CLO)] depends on the belief at every time step.

|                        | Bayesian Filtering | Vanilla Classification |
|------------------------|--------------------|------------------------|
| Episodic Accuracy      | 91.18%             | 90.41%                 |
| Average Reward         | 152.37 ± 49.58     | 122.44 ± 54.80         |

Table I: Comparisons of episodic accuracies and average rewards under dynamic weather.

This work proposes an online meta reinforcement learning algorithm based on conjectural online lookahead adaptation (COLA) that adapts the policy to the nonstationary environment modeled as a hidden-mode MDP. COLA enables prompt online adaptation by solving the conjectural lookahead optimization (CLO) on the fly using off-policy data, where the agent’s conjecture (belief) on the hidden mode is updated in a Bayesian manner.
REFERENCES

[1] S. Ross, G. Gordon, and D. Bagnell, “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning,” ser. Proceedings of Machine Learning Research, vol. 15. Fort Lauderdale, FL, USA: JMLR Workshop and Conference Proceedings, 2011, pp. 627–635. [Online]. Available: http://proceedings.mlr.press/v15/ross11a.html

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legre, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[3] T. Li, G. Peng, Q. Zhu, and T. Basar, “The Confluence of Networks, Games and Learning,” arXiv, 2021.

[4] B. R. Kiran, I. Sobb, V. Talpaert, P. Mannion, A. A. Sallab, S. Yagamani, and P. Perez, “Deep Reinforcement Learning for Autonomous Driving: A Survey,” IEEE Transactions on Intelligent Transportation Systems, vol. PP, no. 99, pp. 1–18, 2021.

[5] J. Chen, S. E. Li, and M. Tomizuka, “Interpretable End-to-End Urban Autonomous Driving With Latent Deep Reinforcement Learning,” IEEE Transactions on Intelligent Transportation Systems, vol. PP, no. 99, pp. 1–11, 2021.

[6] T. Li and Q. Zhu, “On Convergence Rate of Adaptive Multiscale Value Function Approximation for Reinforcement Learning,” 2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6, 2019.

[7] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in Advances in Neural Information Processing Systems 12. MIT press, 2000, pp. 1057–1063. [Online]. Available: http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

[8] T. Li, G. Peng, and Q. Zhu, “Blackwell Online Learning for Markov Decision Processes,” 2021 55th Annual Conference on Information Sciences and Systems (CISS), vol. PP, no. 1, pp. 1–6, 2021.

[9] S. Patanakadla, “A Survey of Reinforcement Learning Algorithms for Dynamically Varying Environments,” ACM Computing Surveys, vol. 54, no. 6, pp. 1–25, 2021.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” vol. 78, pp. 1–16, 2017. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[11] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey, “Meta-Learning in Neural Networks: A Survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PP, no. 99, pp. 1–1, 2021.

[12] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in International Conference on Machine Learning (ICML), 2017, pp. 126–135.

[13] S. P. M. Choi, D.-Y. Yeung, and N. L. Zhang, Hidden-Mote Markov Decision Processes for Nonstationary Sequential Decision Making. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 264–287.

[14] T. Li, Y. Zhao, and Q. Zhu, “The role of information structures in game-theoretic multi-agent learning,” Annual Reviews in Control, vol. 53, pp. 296–314, 2022.

[15] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st ed. USA: John Wiley & Sons, Inc., 1994.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning,” 2016.

[17] A. Nagaband, I. Clavera, S. Liu, R. S. Fearning, P. Abbeel, S. Levine, and C. Finn, “Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning,” arXiv, 2018.

[18] Z. Li, F. Zhou, F. Chen, and H. Li, “Meta-sgd: Learning to learn quickly for few-shot learning,” arXiv preprint arXiv:1707.09835, 2017.

[19] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, “Efficient online policy meta-reinforcement learning via probabilistic context variables,” in Proceedings of the 30th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 5331–5340. [Online]. Available: https://proceedings.mlr.press/v97/rakelly19a.html

[20] J. Harrison, A. Sharma, C. Finn, and M. Pavone, “Continuous Meta-Learning without Tasks,” in Advances in Neural Information Processing Systems, vol. 33. Curran Associates, Inc., pp. 17571–17581.

[21] J. Pineau, G. Gordon, and S. Thrun, “Anytime Point-Based Approximations for Large POMDPs,” Journal of Artificial Intelligence Research, vol. 27, pp. 335–380, 2007.

[22] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa, “Online Planning Algorithms for POMDPs,” Journal of Artificial Intelligence Research, vol. 32, no. 2, pp. 663–704, 2014.

[23] D. Silver and J. Veness, “Monte-Carlo Planning in Large POMDPs,” in Advances in Neural Information Processing Systems, vol. 23. Curran Associates, Inc., pp. 11909–11919.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in Proceedings of the 32nd International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1889–1897. [Online]. Available: https://proceedings.mlr.press/v37/schulman15.html

[25] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained Policy Optimization,” vol. 70, pp. 22–31, 2017-06. [Online]. Available: https://proceedings.mlr.press/v70/achiam17a.html

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv, 2015.

[27] S. Shalev-Shwartz, “Online Learning and Online Convex Optimization,” Foundations and Trends in Machine Learning, vol. 4, no. 2, pp. 107–194, 2011.

[28] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and Decision-Making for Autonomous Vehicles,” Annual Review of Control, Robotics, and Autonomous Systems, vol. 1, no. 1, pp. 1–24, 2018.

[29] A. E. Sallab, M. Abdou, E. Perot, and S. K. Yogamani, “End-to-end deep reinforcement learning for lane keeping assist,” ArXiv, vol. abs/1612.04340, 2016.

[30] P. Wang, C. Chan, and A. de La Fortelle, “A reinforcement learning based approach for automated lane change maneuvers,” 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1379–1384, 2018.

[31] D. C. K. Ngai and N. H. C. Yung, “A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers,” IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 2, pp. 509–522, 2011.

[32] Y. Jaafra, A. Deruyver, J. L. Laurent, and M. S. Naceur, “Context-Aware Autonomous Driving Using Meta-Reinforcement Learning,” 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), vol. 00, pp. 450–455, 2019.

[33] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, “Online Meta-Learning,” vol. 97, no. 19, pp. 1920–1930. [Online]. Available: https://proceedings.mlr.press/v97/finn19a.html

[34] X. A. Xie, J. Harrison, and C. Finn, “Deep Reinforcement Learning amidst Lifelong Non-Stationarity,” arXiv, 2020.

[35] M. Caccia, P. Rodriguez, O. Ostapenko, F. Normandin, M. Lin, L. Page-Caccia, I. H. Laradji, I. Rish, A. Lacoste, D. VAquez, and L. Charlin, “Online Fast Adaptation and Knowledge Accumulation (OSAKA): a New Approach to Continual Learning,” in Advances in Neural Information Processing Systems, vol. 33. Curran Associates, Inc., 2020, pp. 16532–16545.

[36] P. Palanisamy, “Multi-agent connected autonomous driving using deep reinforcement learning,” 2019.

[37] J. Schmidhuber and D. Wierstra, “Hands-On Intelligent Agents with OpenAI Gym: Your Guide to Developing AI Agents Using Deep Reinforcement Learning.” Packt Publishing, 2018.

[38] J. Schulman, X. Chen, and P. Abbeel, “Equivalence between policy gradients and soft q-learning,” 2017. [Online]. Available: https://arxiv.org/abs/1706.06440

[39] L. N. Smith and N. Topin, “Super-convergence: Very fast training of neural networks using large learning rates,” 2017. [Online]. Available: https://arxiv.org/abs/1708.07120