Control-Aware Prediction Objectives for Autonomous Driving

Rowan McAllister[1], Blake Wulfe[1], Jean Mercat[1], Logan Ellis[1], Sergey Levine[2], Adrien Gaidon[1]

***Abstract*— Autonomous vehicle software is typically structured as a modular pipeline of individual components (e.g., perception, prediction, and planning) to help separate concerns into interpretable sub-tasks. Even when end-to-end training is possible, each module has its own set of objectives used for safety assurance, sample efficiency, regularization, or interpretability. However, intermediate objectives do not always align with overall system performance. For example, optimizing the likelihood of a trajectory prediction module might focus more on easy-to-predict agents than safety-critical or rare behaviors (e.g., jaywalking). In this paper, we present control-aware prediction objectives (CAPOs), to evaluate the *downstream effect* of predictions on control without requiring the planner be differentiable. We propose two types of importance weights that weight the predictive likelihood: one using an attention model between agents, and another based on control variation when exchanging predicted trajectories for ground truth trajectories. Experimentally, we show our objectives improve overall system performance in suburban driving scenarios using the CARLA simulator.**

I. Introduction

Autonomous vehicles (AVs) must navigate busy roads using predictive models to anticipate what surrounding pedestrians and vehicles might do in order to plan safe trajectories around them. Safe operation requires such components be well calibrated, typically by minimizing some regression error on training data. However, not all errors made by prediction modules are equally important: some errors have minimal effect on downstream decisions, while some perceptual errors [24] and predictive errors [25] can have fatal outcomes. As no model is perfect, it is crucial to identify *which* prediction errors are safety-critical to ensure safety [19].

Whether trained independently or as part of multi-task end-to-end architectures [30], multi-agent trajectory forecasting models typically optimize prediction-specific objectives based on regressing recorded future trajectories by considering all agents equally important *a priori*. However, when considering the target control task of autonomous navigation, some predictions warrant more attention than others when deciding safe controls. Consequently, control-agnostic optimizing of prediction models may not result in improved downstream navigation performance due to limited data, model capacity, rare events, or computational constraints. Even with end-to-end training, multi-task objectives might not be aligned, thus resulting in performance degradation due to task interference [36].

In this work, we propose *Control-Aware Prediction Objectives* (CAPOs) to train prediction models that more accurately reflect the relative effects of predictive errors on downstream control. Computing these downstream effects requires only forward passes without backpropagation between modules. This improves applicability with real-world AV planning and control systems, which might not be fully differentiable due to complex design constraints (e.g., verifiability, interpretability, comfort and safety constraints). Our method introduces importance-weighted prediction likelihood objectives using forward passes of the prediction model and planner. We investigate two weighting methods that can be trained with backpropagation. The first assigns weights based on control variations due to prediction changes. The second uses learned attention weights between agent predictions and AV controls.

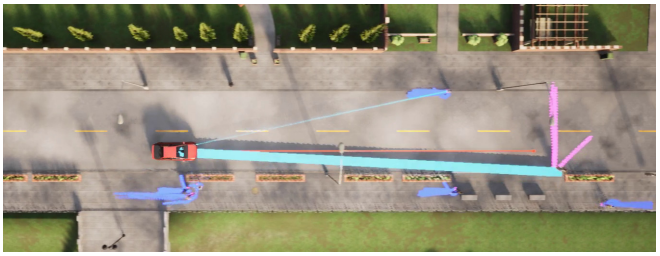

Fig. 1: A vehicle drives to the right while reacting to pedestrians with sample predicted trajectories shown in purple or pink. Our Control-Aware Prediction Objectives (CAPO) can learn to capture which predictions should have more influence on the vehicle's controls (cyan lines proportional to attention). Videos available at `https://sites.google.com/view/control-aware-prediction`

Using the CARLA simulator, we experimentally show that training prediction models with control-aware objectives leads to improved controller performance in complex multi-agent urban driving scenarios. Compared with existing prediction models, including prediction algorithms that treat everything as equal, we show that our new objective helps to avoid precisely those errors that would maximally influence downstream decisions.

II. Related Work

Several related fields of study investigate objective-aware prediction metrics as we discuss here.

A. Objective-Aware Prediction in Reinforcement learning

Model-based reinforcement learning (MBRL) methods learn a dynamics model of an autonomous agent to predict which control decisions lead to states with higher objective rewards [7, 22, 21]. MBRL prediction is related to AV prediction, with the main difference being that AVs predict

[1]Toyota Research Institute {`first.lastname`}`@tri.global`
[2]University of California, Berkeley `svlevine@berkeley.edu`

the trajectories of *other* human agents and not those of the autonomous agent. Nevertheless, several MBRL works have recently challenged the common assumption that the better a dynamics model's predictive accuracy, the better the downstream policy will maximize reward. For example, Lambert et al. [17] show task-agnostic loss functions used to train dynamics models are often uncorrelated with episode rewards, an issue termed "objective mismatch".

Indeed, learned dynamics models need not be accurate everywhere in the state space, only in the areas that help maximize rewards [3]. Some RL works investigate training models insofar as they improve estimating the value function [11, 2], policy gradient [14, 1], or ability to reach a goal state [23]. Others optimize downstream policies directly using Bayesian optimization to search model parameters [3]. Work by Donti et al. [8] points out that in practice we often want a combination of training a model to optimize its likelihood as well as a downstream task term, although in the context of constrained optimisation. Similarly, Lambert et al. [17] correlate both metrics by increasing the weight in the loss of data points closer to data the optimal controller generates.

Unfortunately for AV applications, such objective-aware prediction methods are often inapplicable for several reasons. First, MBRL assumes access to an objective reward function, reward samples, or goal state, but objective measures or goals of desirable driving are often difficult to define. By contrast, human-designed AV control systems are often preferable for verification and interpretability reasons. Second, a common assumption in RL is that the policy is either differentiable or stochastic (in the case of policy gradients), whereas real-world AV control systems often contain complex logic that is neither differentiable nor stochastic. Our work focuses instead on how to learn prediction given access to a safe, potentially non-differentiable controller.

B. Map-Aware Prediction Metrics

Map information can help incorporate prior knowledge into prediction metric design. For example, since the ego vehicle drives on the road, pedestrian forecasting errors could be given more weight on road surfaces than otherwise. Work by Shridhar et al. [32] use maps to help focus on potential collisions with other agents by generating a set of candidate ego trajectories along known lane tracks that any controller might follow. This method does not assume a particular downstream controller, but makes an educated guess as to what a reasonable controller might do. Another map-based prediction metric is Drivable Area Compliance (DAC) [5], which counts the proportion of model samples that exit the drivable area. A conceptual difference with our method is that our prediction metrics assume access to the specific downstream controller that will be used at test-time, improving test-time performance. Since controllers already consider road information, we circumvent the need to explicitly design prediction metrics around mapping information, which can incur additional hyperparameters, such as the relative costs of predicting if an agent will traverse either road / sidewalk / building.

C. Control-Aware Perception

Philion et al. [26] propose a control-aware 3D object perception metric called Planning KL-divergence (PKL) based on how perceptual errors cause distributional divergence in the ego's distribution of planned paths, compared to a planner with perfect observations, measured by the KL divergence. In contrast, our work focuses on prediction objectives, and additionally demonstrates how the new objective empirically affects online-control using a driving simulator. We also avoid KL distance losses in our work since this assumes non-trivial stochasticity in the controller or data, which is not always the case. Other works have also used KL policy distances to investigate how observations can be compressed while preserving human-like actions had they remained uncompressed [28]. Work by Piazzoni et al. [27] also investigates how perceptual metrics affect downstream planning but are specific to perception.

III. Preliminaries

Here we formalize our notation and discuss some existing prediction metrics before presenting our own.

A. Notation and Assumptions

Let $\mathbf{x} \in \mathcal{X}$ denote past trajectory information about all agents, used to make probabilistic predictions $\hat{\mathbf{y}} \in \mathcal{P}_{\mathcal{Y}}$ about the future multi-agent trajectories $\mathbf{y} \in \mathcal{Y}$. Trajectories are predicted up to time horizon T, and $\mathbf{y}_T$ denotes the future state at time T. As the intents of other agents are usually uncertain, we use a *probabilistic* prediction model q_θ with trainable parameters θ to sample the motion of others: $\hat{\mathbf{y}} \sim q_\theta(\mathbf{Y}|\mathbf{x})$, denoting likelihoods as $q_\theta(\mathbf{y}|\mathbf{x}) \doteq q_\theta(\mathbf{Y} = \mathbf{y}|\mathbf{x})$. If multiple samples are taken, $\hat{\mathbf{y}}^k$ refers to the kth sample, and to single out the nth agent we overload notation using $\hat{\mathbf{y}}_n$, and use $\mathbf{y}_{\text{ego}}$ as the AV's future trajectory. Given such predictions, the AV controller π outputs ego controls $\mathbf{u} \in \mathcal{U}$ to anticipate and avoid colliding with other agents' future trajectories: $\mathbf{u} = \pi(\mathbf{y})$.

We assume our AV stack performs behavior prediction before control, a common assumption [31]. While conditioning behavior prediction on ego's intent provides more accurate prediction, for sake of simplicity we assume that other agents do not anticipate the AV's future, only the AV anticipates the other agents' future trajectories in order to avoid collisions.

B. Common Prediction Metrics and Objectives

Common prediction metrics in the literature and in prediction benchmarking challenges–including Argoverse Forecasting [5], Lyft Prediction [12], Waymo Open Motion [10], and AIODrive [35]–are summarized in Table I:

TABLE I: Common prediction metrics in the literature.

Metric Name	Metric Equation
Average Displacement Error (ADE)	$\|\|\hat{\mathbf{y}} - \mathbf{y}\|\|_2$
Final Displacement Error (FDE)	$\|\|\hat{\mathbf{y}}_T - \mathbf{y}_T\|\|_2$
Minimum-ADE (minADE)	$\min_{k\in[K]} \|\|\hat{\mathbf{y}}^k - \mathbf{y}\|\|_2$
Minimum-FDE (minFDE)	$\min_{k\in[K]} \|\|\hat{\mathbf{y}}^k_T - \mathbf{y}_T\|\|_2$
Miss Rate (MR)	$\frac{1}{K}\sum_k \mathbb{1}[\|\|\hat{\mathbf{y}}^k_T - \mathbf{y}_T\|\|_2 > \alpha]$
Negative Log Likelihood (NLL):	$-\log q_\theta(\mathbf{y}|\mathbf{x})$

Most metrics compare the Euclidean distance between either the full predicted state-sequence $\hat{\mathbf{y}}$ (or final state $\hat{\mathbf{y}}_T$) with the true sequence $\mathbf{y}$ (or final state $\mathbf{y}_T$) an agent took, as recorded in data. Probabilistic models are typically trained to minimize the negative log likelihood (NLL) of the data. All such metrics are agnostic to road geometry and downstream planning, which implicitly assumes that all other agents' forecasts are equally relevant. For example, consider two pedestrians: one walking ahead of the ego vehicle and one behind. Assuming independent pedestrian motion, the NLL objective factorizes as: $-\log q_\theta(\mathbf{y}^{\text{ahead}}, \mathbf{y}^{\text{behind}}|\mathbf{x}) = -\log q_\theta(\mathbf{y}^{\text{ahead}}|\mathbf{x}) - \log q_\theta(\mathbf{y}^{\text{behind}}|\mathbf{x})$. Notice that this prediction metric is *equally* concerned with both $\mathbf{y}^{\text{ahead}}$ and $\mathbf{y}^{\text{behind}}$. Intuitively, accurate prediction of the pedestrian ahead of the ego vehicle is more important for safe motion planning since the ego's planned path is more likely to intersect with $\mathbf{y}^{\text{ahead}}$ than $\mathbf{y}^{\text{behind}}$. How can prediction metrics become "aware" that errors in predicting $\mathbf{y}^{\text{ahead}}$ have greater downstream consequences than errors in $\mathbf{y}^{\text{behind}}$?

IV. CAPO: Control-Aware Prediction Objectives

In this section we propose novel prediction loss functions that consider *how* predictions will be used downstream to improve predictive accuracy whenever prediction errors would cause a large change in control outputs. In Bayesian decision theory, a decision is evaluated as the expected utility of a decision $\mathbf{u}$ or controller π, integrating out any uncertainties [4]. In our case, it is the future trajectories of other agents that are unknown but can be probabilistically predicted according to a model with parameters θ. Following the literature on loss-calibrated variational inference [16, 6, 15], we define the *gain* of a decision or controller's value as a function of the model parameters θ that we wish to train.

$$\text{Gain}_{\mathbf{x},\mathbf{y},\pi}(\theta) = \int \text{utility}(\pi, \mathbf{y}, \hat{\mathbf{y}}, \mathbf{x}) q_\theta(\hat{\mathbf{y}}|\mathbf{x}) \text{d}\hat{\mathbf{y}}. \quad (1)$$

The choice of utility function in Eq. (1) is an open one, that is why we considered many possible input parmeter; it defines how desirable a course of actions would be given $\mathbf{x}$ and $\hat{\mathbf{y}}$. Alternatively, an existing metric like the NLL can simply be weighted without integration. In the next subsection we discuss some baseline choices for the utility or weight, and after, we propose two novel methods for computing these weights: a *self-attention* method and a *counterfactual* method.

A. Baseline Objectives

Most predictive metrics in the literature are agnostic to $\mathbf{u}$ and simply use a delta function to only score correct trajectory predictions, recovering the standard log likelihood metric: $\text{Gain}_{\mathbf{x},\mathbf{y}}(\theta) = \int \delta(\mathbf{y} - \hat{\mathbf{y}}) q_\theta(\hat{\mathbf{y}}|\mathbf{x}) \text{d}\hat{\mathbf{y}} = q_\theta(\mathbf{y}|\mathbf{x})$. However, we are interested in utilities that are a function of $\mathbf{u}$ in order to weight predictions by their downstream effect on the ego's control. For instance, we could score trajectory predictions based on the resultant ego controls $\pi(\hat{\mathbf{y}})$ matching the ego's behavior under knowledge of the true future trajectories $\pi(\mathbf{y})$: $\text{Gain}_{\mathbf{x},\mathbf{y},\pi}(\theta) = \int \delta(\pi(\mathbf{y}) - \pi(\hat{\mathbf{y}})) q_\theta(\hat{\mathbf{y}}|\mathbf{x}) \text{d}\hat{\mathbf{y}}$. This integral is unfortunately intractable to derive or estimate, but softer utility functions can be used instead. One example is $||\pi(\hat{\mathbf{y}}) - \pi(\mathbf{y})||_1$, which we include as a baseline in Table II. Optimizing this controller output error guides the learning process towards predicting controller inputs (predicted trajectories) accurately, insofar as they result in the correct control. Any trajectory errors that do not induce a change in the AV's control are thus considered inconsequential and ignored.

B. Attention-based CAPO

We propose a GRU encoder-decoder architecture with an attention mechanism as introduced in [34]. Our method weights the agent predictions using attention factors between agents $\mathbf{x}$ and the AV's future trajectory $\mathbf{y}_{\text{ego}}$. The predictive model is a function parameterized by θ noted $q_\theta : \mathcal{X} \to \mathcal{P}_{\mathcal{Y} \times \mathcal{Y}_{\text{ego}}}$. We note $\theta = \{\theta_{\text{ego}}, \theta_{\text{agent}}\}$ where θ_{ego} is the set of parameters for the ego decoder only; $\mathcal{X}$ is the past observation space and $\mathcal{P}_{\mathcal{Y} \times \mathcal{Y}_{\text{ego}}}$ the probability spaces of future trajectories: $\mathcal{P}_{\mathcal{Y}_{\text{ego}}}$ for the ego and $\mathcal{P}_{\mathcal{Y}}$ for other agents.

We use an architecture similar to [18, 20] where we train a model with multi-head attention; the ego agent attends the other agents. The ego predictions are used as a proxy for the actual planner to compute the importance weights of other agents:

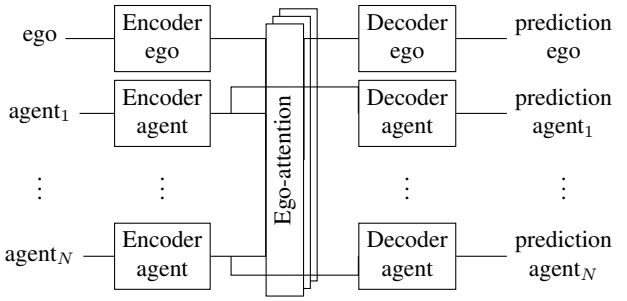

Fig. 2: Diagram of the attention model. All agent encoders and decoders share their weights. Encoders and decoders are GRUs. Attention is not used between agents.

The ego-attention blocks in figure Fig. 2 are heads of a multi-head attention mechanism. The computation performed by each head is given below:

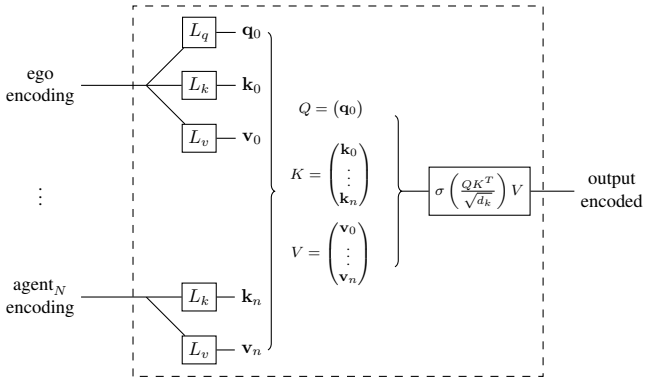

Fig. 3: Diagram of a self-attention head.

The attention vector is given by

$$\alpha = \sigma\left(\frac{QK^\top}{\sqrt{d_k}}\right) = [\alpha_0, ..., \alpha_N], \tag{2}$$

where Q is the query matrix, K the key matrix, and σ the softmax operation that normalizes the attention vector (for details see [34]). The encoded output $= \alpha V$ is a weighted mean of the value vectors over the agents (including ego).

Inspired by [20] the attention model produces outputs in the form of a sequence of Gaussian mixtures for each agent and is trained to minimize the NLL for all agents and the ego trajectory predictions. However, for our application, ego prediction is not the goal, it is only a proxy to compute the importance weights of the pedestrian contribution to the ego behavior.

We propose to use attention coefficients α as importance factors in a weighted sum of per-human state prediction loss (as opposed to uniform weighting). Algorithm 1 summarizes how the model is trained with importance weighting. If multiple heads are used, the attention coefficients are averaged:

$$w_n = \frac{1}{H}\sum_{h=1}^{H} \alpha_n^{(h)} \tag{3}$$

The attention predictor imitates a planner that interacts with the other agents to avoid collision. The only way for the predictor to interact with the other agents is through attention. Therefore, as the model learns the correlations between the planner's trajectories and the agents trajectories, larger attention coefficients are given to the agents that cause larger reactions from the controller. It learns this offline and does not need access to the controller nor its gradient.

Predicting jointly the ego trajectory and the other agents allows us to use the attention coefficients for concern weighting in a single run. The coefficient α_0 quantifies how independent the ego is from other agents.

This method defines a concern about an agent but not about specific trajectories of that agent. It can define the concern without using the controller because it instead uses an offline-learned model that imitates the controller.

Algorithm 1 Attention CAPO

Input: Controller: $\pi : \mathcal{X} \to \mathcal{U}$

Record trajectory data $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}_i$
while training **do**
  Sample batch $\mathbf{x}, \mathbf{y} \sim \mathcal{D}$
  Run model to estimate $\hat{\mathbf{y}}_{\text{ego}}$ and $\hat{\mathbf{y}}_{\text{agent}}$ from $\mathbf{x}$
  Get attention: $\alpha(\mathbf{x})$ ▷ Eq. (2)
  Compute weight: $w(\alpha(\mathbf{x}))$ ▷ Eq. (3)
  Update model:
  $\theta_{\text{ego}} \leftarrow \theta_{\text{ego}} + \nabla_{\theta_{\text{ego}}} \log q_\theta(\mathbf{y}_{\text{ego}}|\mathbf{x})$
  $\theta_{\text{agent}} \leftarrow \theta_{\text{agent}} + w(\mathbf{x})\nabla_{\theta_{\text{agent}}} \log q_\theta(\mathbf{y}_{\text{agent}}|\mathbf{x})$

Output: Predictive model $q_\theta : \mathcal{X} \to \mathcal{P}_{\mathcal{Y}\times\mathcal{Y}_{\text{ego}}}$

C. Counterfactual Action-Discrepancy CAPO

Our second proposal can also be formulated as a re-weighted maximization objective, where we weight the log likelihood of each agent's trajectory in a scene by its *individual* contribution to the ego's control decision. We do this by first enumerating through each agent in a scene, and computing counterfactual outputs from the AV's controller if every agent traversed their individual trajectory as recorded in the replay buffer, except for agent n. If we resample the trajectory that the nth agent *might otherwise have taken*, $\hat{\mathbf{y}}_n^k \sim q_\theta(\mathbf{Y}_n|\mathbf{x})$, we can compute the control output that would result:

$$\hat{\mathbf{u}}_n^k \;=\; \pi(\{\hat{\mathbf{y}}_n^k\} \cup \mathbf{y} \setminus \{\mathbf{y}_n\}), \tag{4}$$

to compare against the control had no agent deviated from their recorded trajectories:

$$\mathbf{u} \;=\; \pi(\mathbf{y}). \tag{5}$$

The difference in these two hypothetical controls corresponds to how much an individual agent affects the ego vehicle, and can represent the concern associated with predicting this particular agent in this particular instance accurately. If the model is probabilistic, then taking multiple samples $(K > 1)$ helps ensure high importance even if the other agent only *might* cause a control deviation:

$$w_n \;=\; \max_{k\in\{1..K\}} ||\mathbf{u} - \hat{\mathbf{u}}_n^k||_1, \tag{6}$$

which we use as weights for predictive model training:

$$\theta^* \;=\; \arg\max_\theta \sum_{n=1}^{N} w_n \log q_\theta(\mathbf{y}_n|\mathbf{x}). \tag{7}$$

We summarize our counterfactual action discrepancy method in Algorithm 2. One benefit of this approach compared to the attention CAPO is that it is

Algorithm 2 Counterfactual CAPO

Input: Controller: $\pi : \mathcal{X} \to \mathcal{U}$

Record trajectory data $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}_i$
while training **do**
  Sample batch $\mathbf{x}, \mathbf{y} \sim \mathcal{D}$
  Compute hypothetical controls: $\mathbf{u}, \hat{\mathbf{u}}_n^k$ ▷ Eq. (4)–(5)
  Compute weight: $w(\mathbf{u}, \hat{\mathbf{u}}_n^k)$ ▷ Eq. (6)
  Update model: $\theta \leftarrow \theta + w(\mathbf{u}, \hat{\mathbf{u}}_n^k)\nabla_\theta \log q_\theta(\mathbf{y}|\mathbf{x})$

Output: Predictive model $q_\theta : \mathcal{X} \to \mathcal{P}_{\mathcal{Y}}$

D. Summary of Objectives

There are various choices for utilities, or weights for traditional module metrics. In Table II we summarize the several baselines methods, including NLL and our novel proposals.

V. Experimental Evaluation

To evaluate our proposed method, we consider a representative scenario that is commonplace in autonomous driving: pedestrian trajectory prediction. The majority of pedestrian behaviors can safely be ignored by the AV's autonomy stack; however, in rare cases of pedestrian-ego interaction (e.g., road crossings), accurate prediction of pedestrian behavior

TABLE II: Comparison of utilities and weighted objectives.

Method	Utility or Weight	Objective $\mathcal{L}(\theta)$
Baselines:		
Attention	$\delta(\mathbf{y}-\hat{\mathbf{y}})$	$q_\theta(\mathbf{y}\|\mathbf{x})+q_\theta(\mathbf{y}_{\text{ego}}\|\mathbf{x})$
R2P2 Gain$_{\mathbf{y}}$	$\delta(\mathbf{y}-\hat{\mathbf{y}})$	$q_\theta(\mathbf{y}\|\mathbf{x})$
R2P2 Gain$_{\pi 1}$	$\|\|\pi(\mathbf{y})-\pi(\hat{\mathbf{y}})\|\|_1$	$\mathbb{E}_{\hat{\mathbf{y}}}\left[\|\|\pi(\mathbf{y})-\pi(\hat{\mathbf{y}})\|\|_1\right]$
R2P2 Weight$_{\nabla\hat{\mathbf{y}}}$	$\|\|\nabla_{\hat{\mathbf{y}}}\pi(\hat{\mathbf{y}})\|\|_1$	$\mathbb{E}_{\hat{\mathbf{y}}}\left[\|\|\nabla_{\hat{\mathbf{y}}}\pi(\hat{\mathbf{y}})\|\|_1\right]q_\theta(\mathbf{y}\|\mathbf{x})$
R2P2 Weight$_{\nabla\mathbf{y}}$	$\|\|\nabla_{\mathbf{y}}\pi(\mathbf{y})\|\|_1$	$\|\|\nabla_{\mathbf{y}}\pi(\mathbf{y})\|\|_1 q_\theta(\mathbf{y}\|\mathbf{x})$
Ours:		
R2P2 Weight$_{\pi}$	$\|\|\pi(\mathbf{y})-\pi(\hat{\mathbf{y}})\|\|_1$	$\mathbb{E}_{\hat{\mathbf{y}}}\left[\|\|\pi(\mathbf{y})-\pi(\hat{\mathbf{y}})\|\|_1\right]q_\theta(\mathbf{y}\|\mathbf{x})$
R2P2 Weight$_{\pi k}$	$\max_k \|\|\pi(\mathbf{y})-\pi(\hat{\mathbf{y}}^k)\|\|_1$	$\max_k \|\|\pi(\mathbf{y})-\pi(\hat{\mathbf{y}}^k)\|\|_1 q_\theta(\mathbf{y}\|\mathbf{x})$
AttentionWeight	$\alpha(\mathbf{x})$	$\alpha(\mathbf{x})q_\theta(\mathbf{y}_{\text{agent}}\|\mathbf{x})+q_\theta(\mathbf{y}_{\text{ego}}\|\mathbf{x})$

becomes crucial in avoiding collisions. This sparsity of interaction showcases how predictive models may perform well with respect to traditional metrics (e.g., ADE) while still leading to suboptimal ego behavior when it matters most. Here, we first detail our experimental evaluation and implementation of the aforementioned scenario within the CARLA autonomous driving simulator [9]. We then compare results between our method and the various baselines discussed in Table II, where our experiments show that predictive models trained using our CAPO methods produce safe behavior with fewer collisions relative to other baselines.

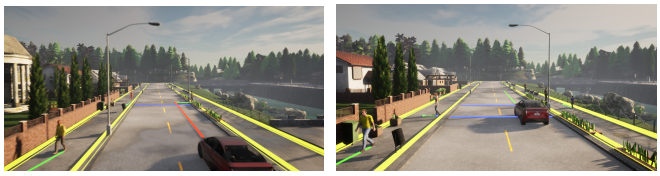

Fig. 4: **Pedestrian Prediction Scenario**. Pedestrians spawn on the sidewalk (yellow region) and the ego (red car) predicts the pedestrian trajectories within the next 3 seconds (green). Some pedestrians will cross the road at right angles (blue). **Left**: the planner predicts a collision with a crossing pedestrian and starts slowing (red ego prediction up to blue line but not further). **Right**: ego is safely passing the road segment where the pedestrian has already crossed.

A. CARLA Scenario Design

We implement our pedestrian prediction scenario in the CARLA simulator's Town01. A single ego vehicle is commanded to drive down a road that is adjacent to sidewalks which are populated with pedestrians. Occasionally, a pedestrian will cross the street and the ego agent must slow to avoid a collision when necessary.

The ego vehicle adjusts its longitudinal control to balance the competing priorities of maintaining the current speed limit (45mph) while avoiding collisions with the pedestrians crossing the road (either their current or future-predicted distance on the road ahead). This balance is performed by an Intelligent Driver Model [33]. It uses predictions to estimate the closest collision distance and controls the vehicle to stop ahead of that point.

Pedestrians spawn at random locations on the sidewalk and are then provided a long-range navigation goal that is also uniformly sampled from the sidewalk. When the long-range goal is reached, another is sampled to replace it. To induce pseudo-random motion, a short-range goal is also generated at each time step. This goal is generated by projecting a point 4m along the path to the long-range goal, starting at the pedestrian's location. The lateral offset β_{t+1} of the short-range goal is generated by sampling from a normal distribution centered about the previous lateral offset β_t after it has been scaled down (to drive it towards the long-range goal):

$$\beta_{t+1} = (1-\varepsilon)\beta_t + \mathcal{N}(0,\sigma^2), \tag{8}$$

where σ is the variance of the noise, and $\epsilon \in [0,1)$ is the commitment to the long-range goal.

When on the sidewalk, pedestrians are programmed to walk at speeds sampled about 2m/s while navigating around other pedestrians to avoid collisions and, occasionally, will pause outside of shops. Each different kind of pedestrian is defined with various noise levels, commitment, and stopping chance.

Pedestrians may also randomly decide to cross the road. The probability increases if their velocity vector points towards the road and increases greatly when the pedestrian is close to the road. While crossing, they travel at 2 m/s in the shortest path possible, i.e., perpendicular to the road direction. To increase task difficulty, the probability that pedestrians will cross the road is increased at test time.

B. Compared models

1) **Oracle distribution** The pedestrian behavior is simulated with a known distribution at each time step, which is sampled to produce a trajectory. The trajectory distribution is approximated by sampling $K = 5$ trajectories for each pedestrian. The planner reacts to the trajectory that would cause the closest intersection with its desired path. This method is a perfect probabilistic predictor which is only accessible with simulated data. Its predictions are not biased towards sampling the most critical trajectories, and planning with relatively few samples can yield suboptimal results.

2) **Gradient weighting** Recent work has also investigated weighing prediction objectives by a measure of local sensitivity of downstream costs to individual predictions [13]. This method first learns a cost function to evaluate ego controls given predicted human trajectories, using inverse optimal control. The method then weights each prediction loss by the gradient magnitude of the cost outputs w.r.t. the prediction inputs. In our work, we consider using the controller directly, forming an analogous baseline: using gradients of the controller w.r.t. the predicted trajectory $||\nabla_{\hat{\mathbf{y}}}\pi(\hat{\mathbf{y}})||_1$, or true trajectory $||\nabla_{\mathbf{y}}\pi(\mathbf{y})||_1$, included in Table II. While we do not assume differentiable controllers in our own methods, we nevertheless experiment using a differentiable controller to compare against this baseline method.

TABLE III: Scenario results, 100 episodes. Arrows indicate higher/lower preferred. Standard errors shown. **Best**, second.

Predictive Model	Success Rate ↑	Collisions ↓	Speed (m/s) ↑	Jerk (m/s^{-3}) ↓	ADE (m) ↓	Control Error ↓
Baselines:						
R2P2 Gain$_{\mathbf{y}}$	89.0%	11	9.97 ±0.222	8.92 ±0.250	2.09 ±0.024	0.59 ±0.012
R2P2 Gain$_{\pi 1}$	85.0%	14	10.45 ±0.268	6.65 ±0.196	3.48 ±0.038	0.63 ±0.016
R2P2 Weight$_{\nabla \hat{\mathbf{y}}}$	94.0%	4	9.53 ±0.216	8.21 ±0.140	**1.98** ±0.024	0.60 ±0.012
R2P2 Weight$_{\nabla \mathbf{y}}$	91.0%	9	9.74 ±0.216	8.74 ±0.184	2.00 ±0.025	0.60 ±0.011
Attention [20]	89.0%	11	13.79 ±0.214	4.48 ±0.147	2.61 ±0.050	0.63 ±0.026
Oracle distribution	98.0%	2	10.54 ±0.231	6.80 ±0.180	1.58 ±0.036	0.51 ±0.013
Our methods:						
R2P2 Weight$_{\pi}$	93.0%	7	8.86 ±0.188	9.26 ±0.194	2.29 ±0.022	0.58 ±0.010
R2P2 Weight$_{\pi k}$	**99.0%**	**1**	9.46 ±0.196	7.89 ±0.159	2.14 ±0.018	**0.55** ±0.011
Attention Weight$_{\alpha}$	91.0%	9	**14.36** ±0.217	**4.22** ±0.154	2.58 ±0.053	0.64 ±0.024

3) **Attention weighting** As presented in section IV-B. We train this model with our CAPO method (algorithm 1) and, as a baseline, we compare it with the unbiased prediction as the predictor [20] would produce.

4) **Reparameterized Pushforward Policy (R2P2)** we use the likelihood-based multi-agent prediction algorithm R2P2 [29] as baseline Gain$_{\mathbf{y}}$, and also use R2P2 as the base model for all other predictive models apart from the attention model. R2P2 is a autoregressive normalizing flow, capable of expressing multimodal agent trajectories, trained with NLL. We parameterize R2P2 to predict 30 steps with data at 10Hz, corresponding to a 3s prediction for all pedestrians. We train it with our CAPO method (algorithm 2) using $K = 10$ samples and we use $K = 1$ samples at test time.

C. Metrics

The table III presents our results for 100 sequences. We track the performance of the system (prediction and planner) with the success rate and the number of collisions. Three conditions may end a sequence:

- Success: vehicles traverses 200m road without incident.
- Collision: a pedestrian was hurt.
- Time out: the car was too slow (> 60s).

We also score efficiency and comfort indicators by average speed and average jerk respectively. Finally, we compute the average pedestrian trajectory prediction errors and their downstream effect on the planner with the average displacement error (ADE) and the *Control Error* equal to $||\pi(\mathbf{y}) - \pi(\hat{\mathbf{y}})||_1$. The control error measures the downstream effect of the prediction error on the ego's plans.

VI. Discussion

The results in Table III show that while all methods do reasonably well, weighting predictive objectives by their downstream effect improves the downstream performance as illustrated by a low collision count and control error. While methods such as R2P2 Weight$_{\nabla \hat{\mathbf{y}}}$ assume a differentiable controller, we find this assumption does not need to be made, and our methods can work with any type of controller. While our methods did not score as well on the ADE metric of agents' trajectories, they did score best on the metrics that matters more: the control error, and success rate, thus mitigating error propagated downstream and improving the end task performance. We compared to objectives weighted by the planner's sensitivity $||\nabla_{\hat{\mathbf{y}}} \pi(\hat{\mathbf{y}})||_1$, which is related to [13]. However, such methods assume the planner (or cost function) is differentiable, which is often not the case in real AV systems. Secondly, *local* sensitivity to a point prediction is not necessarily a good measure of relevance if the gradient is noisy or changes drastically over the full predictive distribution. While our experiments show encouraging results, testing with various setups and environments would be needed to give a clear best method.

VII. Conclusions

Modular autonomous systems (such as those commonly used in AVs) provide a number of advantages, but generally incur the disadvantage that individual components typically do not optimize for system-wide or downstream performance metrics directly. In this paper, we proposed weighted objectives for learning prediction models that account for the downstream objective without imposing stringent requirements on downstream components (such as end-to-end differentiability). These objectives weight the usual likelihood objective, either using attention weights derived from a behavior-cloned policy, or using the impact that substituting predicted trajectories for ground-truth trajectories has on planner output. Accounting for the downstream objective in this manner encourages prediction models to focus on what's important – either at the agent or individual trajectory level – and, as a result, improves system-wide performance, as we showed empirically in a pedestrian jaywalking scenario.

A number of promising avenues exist for future research. First, control-aware objectives may provide out-of-domain generalization benefits by encouraging prediction models to focus on relevant aspects of the scene, and ignore spurious sources of information that are safe to ignore. Second, in this paper we focused on data collected from an expert. However, this requirement limits the applicability of the proposed metrics, and a broader coverage of the state-space resulting from the use of both expert and suboptimal data might improve the learned prediction models. Finally, although we focused in this paper on using control-aware weighting for *optimizing* prediction models. Our method might equally well be used to define a weighted metric for *evaluating* models in a validation setting where training-time access to the downstream planner is either not available or undesirable.

References

[1] Romina Abachi, Mohammad Ghavamzadeh, and Amir-massoud Farahmand. "Policy-aware model learning for policy gradient methods". In: *arXiv preprint arXiv:2003.00030* (2020).

[2] Alex Ayoub et al. "Model-based reinforcement learning with value-targeted regression". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 463–474.

[3] Somil Bansal et al. "Goal-driven dynamics learning via Bayesian optimization". In: *Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 5168–5173.

[4] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[5] Ming-Fang Chang et al. "Argoverse: 3D tracking and forecasting with rich maps". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8748–8757.

[6] Adam D Cobb, Stephen J Roberts, and Yarin Gal. "Loss-calibrated approximate inference in Bayesian neural networks". In: *arXiv preprint arXiv:1805.03901* (2018).

[7] Marc Deisenroth and Carl Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *International Conference on machine learning (ICML)*. Citeseer. 2011, pp. 465–472.

[8] Priya Donti, Brandon Amos, and Zico Kolter. "Task-based End-to-end Model Learning in Stochastic Optimization". In: *Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 5490–5500. URL: https://proceedings.neurips.cc/paper/2017/file/3fc2c60b5782f641f76bcefc39fb2392-Paper.pdf.

[9] Alexey Dosovitskiy et al. "CARLA: An open urban driving simulator". In: *Conference on robot learning*. PMLR. 2017, pp. 1–16.

[10] Scott Ettinger et al. "Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset". In: *arXiv preprint arXiv:2104.10133* (2021).

[11] Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. "Value-aware loss function for model-based reinforcement learning". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 1486–1494.

[12] John Houston et al. "One thousand and one hours: Self-driving motion prediction dataset". In: *arXiv preprint arXiv:2006.14480* (2020).

[13] Boris Ivanovic and Marco Pavone. "Rethinking Trajectory Forecasting Evaluation". In: *arXiv preprint arXiv:2107.10297* (2021).

[14] Joshua Joseph et al. "Reinforcement learning with misspecified model classes". In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2013, pp. 939–946.

[15] Tomasz Kuśmierczyk, Joseph Sakaya, and Arto Klami. "Variational Bayesian decision-making for continuous utilities". In: *arXiv preprint arXiv:1902.00792* (2019).

[16] Simon Lacoste–Julien, Ferenc Huszár, and Zoubin Ghahramani. "Approximate inference for the loss-calibrated Bayesian". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 416–424.

[17] Nathan Lambert et al. "Objective mismatch in model-based reinforcement learning". In: *arXiv preprint arXiv:2002.04523* (2020).

[18] Edouard Leurent and Jean Mercat. "Social attention for autonomous decision-making in dense traffic". In: *arXiv preprint arXiv:1911.12250* (2019).

[19] Rowan McAllister et al. "Concrete problems for autonomous vehicle safety: advantages of Bayesian deep learning". In: International Joint Conferences on Artificial Intelligence. 2017.

[20] Jean Mercat et al. "Multi-head attention for multi-modal joint vehicle motion forecasting". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 9638–9644.

[21] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. "Model-based reinforcement learning: A survey". In: *arXiv preprint arXiv:2006.16712* (2020).

[22] Anusha Nagabandi et al. "Learning image-conditioned dynamics models for control of underactuated legged millirobots". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4606–4613.

[23] Suraj Nair, Silvio Savarese, and Chelsea Finn. "Goal-aware prediction: Learning to model what matters". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7207–7219.

[24] NHTSA. *PE 16-007*. Tech. rep. Tesla Crash Preliminary Evaluation Report. United States Department of Transportation, National Highway Traffic Safety Administration, 2017. URL: https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF.

[25] NTSB. *Preliminary Report Highway: HWY18MH010*. Tech. rep. Highway Accident Report Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian Tempe, Arizona. United States National Transportation Safety Board, 2019. URL: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1903.pdf.

[26] Jonah Philion, Amlan Kar, and Sanja Fidler. "Learning to evaluate perception models using planner-centric metrics". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14055–14064.

[27] Andrea Piazzoni et al. "Modeling Sensing and Perception Errors towards Robust Decision Mak-

ing in Autonomous Vehicles". In: *arXiv preprint arXiv:2001.11695* (2020).

[28] Siddharth Reddy, Anca D Dragan, and Sergey Levine. "Pragmatic Image Compression for Human-in-the-Loop Decision-Making". In: *arXiv preprint arXiv:2108.04219* (2021).

[29] Nicholas Rhinehart, Kris Kitani, and Paul Vernaza. "R2P2: A ReparameteRized Pushforward Policy for Diverse, Precise Generative Path Forecasting". In: *European Conference on Computer Vision*. Springer. 2018.

[30] Abbas Sadat et al. "Perceive, Predict, and Plan: Safe Motion Planning Through Interpretable Semantic Representations". In: *ECCV*. 2020.

[31] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and decision-making for autonomous vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[32] Skanda Shridhar et al. "Beelines: Evaluating Motion Prediction Impact on Self-Driving Safety and Comfort". In: *arXiv preprint arXiv:2011.00393* (2020).

[33] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. "Congested traffic states in empirical observations and microscopic simulations". In: *Physical review E* 62.2 (2000), p. 1805.

[34] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[35] Xinshuo Weng et al. "All-In-One Drive: A Comprehensive Perception Dataset with High-Density Long-Range Point Clouds". In: (2021).

[36] Sen Wu, Hongyang R. Zhang, and Christopher Ré. "Understanding and Improving Information Transfer in Multi-Task Learning". In: *ICLR*. 2020.