A Additional results

A.1 Ablation study

Table 6: Ablation study results.

Method	$\mathcal{L}_{\text{pred}}$ $\times 10^0$	$\mathcal{L}_{\text{plan}}$ $\times 10^2$	$\mathcal{L}_{\text{ctr}}$ $\times 10^3$	ADE (m) ↓ ± Std. Err.	NLL (nats) ↓ ±Std.Err.	Planning Loss ↓ $\times 10^{-2}$, ±Std.Err.	Hindsight Cost ↓ $\times 10^{-2}$, ±Std.Err.
No prediction				–	–	0.00 ± 0.00	0.00 ± 0.00
Standard	1	0	0	1.32 ± 0.06	0.42 ± 0.04	-3.76 ± 0.23	-1.61 ± 0.05
DiffStack (default)	1	1	1	1.27 ± 0.07	0.38 ± 0.05	-5.13 ± 0.18	-1.86 ± 0.04
DiffStack (no $\mathcal{L}_{\text{plan}}$)	1	0	1	1.33 ± 0.08	0.48 ± 0.16	-4.98 ± 0.11	-1.86 ± 0.04
DiffStack (no $\mathcal{L}_{\text{ctr}}$)	1	1	0	1.48 ± 0.21	0.55 ± 0.15	-5.24 ± 0.10	-1.85 ± 0.06
DiffStack	1	5	5	1.64 ± 0.17	0.85 ± 0.08	-5.53 ± 0.07	-1.89 ± 0.03
DiffStack	1	0.2	0.2	1.28 ± 0.04	0.66 ± 0.19	-4.79 ± 0.18	-1.81 ± 0.02
GT prediction				–	–	-7.56 ± 0.00	-2.25 ± 0.00
No pred. (no plan.)				–	–	–	0.0 ± 0.00
Standard (no plan.)	1	–	0	1.27 ± 0.06	0.31 ± 0.19	–	-1.8 ± 0.23
DiffStack (no plan.)	1	–	1	1.40 ± 0.15	0.19 ± 0.40	–	-2.0 ± 0.23
GT pred. (no plan.)				–	–	–	-2.6 ± 0.00
No pred. (no contr.)				–	–	0.00 ± 0.00	0.0 ± 0.00
Standard (no contr.)	1	0	–	1.23 ± 0.02	0.56 ± 0.22	-4.08 ± 0.27	-1.1 ± 0.03
DiffStack (no contr.)	1	1	–	1.46 ± 0.18	0.53 ± 0.13	-5.29 ± 0.10	-1.2 ± 0.01
GT pred. (no contr.)				–	–	-7.56 ± 0.00	-1.5 ± 0.00
No pred. ($\Delta t = 0.1$)				–	–	0.00 ± 0.00	0.0 ± 0.00
Standard ($\Delta t = 0.1$)	1	0	–	1.23 ± 0.02	0.56 ± 0.22	-5.65 ± 1.67	-2.1 ± 0.06
DiffStack ($\Delta t = 0.1$)	1	1	–	1.49 ± 0.12	0.88 ± 0.27	-14.01 ± 0.63	-2.4 ± 0.05
GT pred. ($\Delta t = 0.1$)				–	–	-25.11 ± 0.00	-2.9 ± 0.00

We present results for an ablation study in Table 6. The table is divided into sections in which decision making metrics are comparable. Decision making metrics are not comparable across sections because of the different configuration of the stack and a different criteria for rejecting samples from the validation set. Each section reports planning loss and hindsight cost relative to a **No prediction** baseline, and we report relative results for a **GT prediction** oracle, same as before. In rows of the table we train stacks with different weighted sum of losses, $\mathcal{L} = \alpha_1 \mathcal{L}_{\text{pred}} + \alpha_2 \mathcal{L}_{\text{plan}} + \alpha_3 \mathcal{L}_{\text{ctr}}$. The second to fourth columns indicate the α_i parameters.

In the first section of Table 6 we compare DiffStack trained with different weighted combination of losses. We observe similar improvements when using only the control loss or only the planning loss. The relative weight of the prediction loss vs. the downstream decision making losses behave as expected: the higher/lower the weight for the downstream planning and control losses the more/less planning and control metrics improve at the expense of slightly worse/better raw prediction metrics.

In the second section we remove the planner module, and initialize the control algorithm with a zero control sequence. In the third section we remove the control module and use the output of the planner as the final ego trajectory. In the last section we change the planning and control time resolution from $\Delta t = 0.5$s to $\Delta t = 0.1$s. We observe similar trends in terms of improvement from DiffStack in all cases. In terms of absolute metrics we observe a benefit for using both the planning and the control modules (not shown in Table 6).

A.2 Learned control cost experiment

Table 7: Detailed results for the control cost learning experiment.

Cost weights	w'_1 (C_{coll})	w'_2 (C_{g})	w'_3 ($C_{\ell\perp}$)	w'_4 ($C_{\ell\measuredangle}$)	w'_5 (C_u)	α	Planning Loss ↓	MSE (m^2) ↓
Hand-tuned	5.00	0.50	0.30	0.30	1.00	1.00	2.67	0.38
Learned	5.00	0.68	0.36	0.43	0.63	1.10	**2.41**	**0.32**

We report detailed results for the control cost learning experiment in Table 7. Values are averages over 5 training seeds. All standard errors are ≤ 0.01 (omitted).

We first train a prediction module (as before), then we train for an additional 20 epochs allowing DiffStack to update the weights w_i of the control cost (3) by backpropagating gradients from the final control loss $\mathcal{L}_{\text{ctr}} = \mathcal{L}_{\text{MSE}}$. We do not use a planning loss $\mathcal{L}_{\text{plan}}$. For practical reasons we reparameterize the cost function such that the weights always sum to a fixed constant, and the total cost is multiplied with a scaler α. Specifically, the cost weights are given by $w_i = \alpha w'_i$, and $w'_i = c \exp(\psi_i) / \sum_{j \in 1:5} \exp(\psi_j)$. Here α and ψ_i for $i \in 2:5$ are trainable parameters, c is a constant. To ensure safety we fix ψ_1, the weight parameter for the collision term C_{coll}. We initialize all parameters, ψ_i, α, c such that the default hand-tuned weights (shown in row 1) are recovered. Compared to the default hand-tuned weights (row 1), DiffStack significantly decreases plan MSEs (row 2). The resulting learned weights are lower for the control effort term, and higher for the goal and lane keeping terms. We observed similar results when freezing the prediction module while learning the cost weights.

B Motivation for planning-aware prediction models

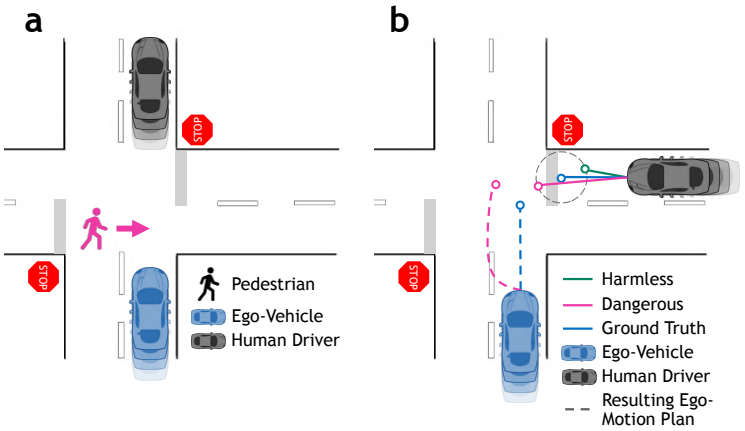

Figure 4: Not all predictions are equally important. (a) The motion of a jaywalker in front of the ego-vehicle is more important than a far-away vehicle. (b) Metrically-equal prediction errors can have severe consequences; incorrectly predicting that a vehicle will not stop at an intersection may trigger a dangerous evasive maneuver in the planner. The visualization was inspired by [27] and the figures include similar graphical elements.

C DiffStack implementation details

C.1 Planning module details

The planner generates trajectory candidates by first sampling a set of lane-centric terminal states. We consider all lanes that are within 4.5m from the ego goal g, and the heading difference between the lane and ego goal state is less than $90°$. For all lanes we generate a set of terminal states based on distance traveled along the lane for a set of fixed acceleration values $a \in \{-3, -2, -1, -0.5, 0, 0.5, 1, 2\}\mathrm{m/s^2}$ and lateral lane offsets $\ell_\perp$ in $\{-0.5, 0, 0.5\}$m. We then fit a cubic spline from the current state to the terminal state that minimizes mean control effort, and drop the dynamically infeasible trajectories given the control limits of the ego vehicle.

C.2 Control module details

The iLQR algorithm iteratively forms and solves a quadratic LQR approximation of the problem around the current solution $s^{(i)}, u^{(i)}$ for iteration i, using a second-order Taylor approximation of the cost function C, and a first-order Taylor approximation of the dynamics function f_d.

In each iLQR iteration we are solving an LQR problem. The solution of the LQR problem is a control sequence which we use as a candidate for a potential trajectory update. Note that the LQR-optimal control sequence is only optimal according to the approximated quadratic cost, and does not necessarily improve the cost for the original non-quadratic problem. Therefore we perform a line-search for the largest change in control sequence in the direction of the LQR-optimal controls that results in a reduction in the original cost. We start with a down-scaling factor of 1 for the change in controls, unroll controls through the dynamics function, and evaluate the original non-quadratic cost for the resulting trajectory. If this cost is larger than the cost of the unchanged trajectory, we decrease the down-scaling factor by a factor of 5 and repeat the search process up to 5 times. Otherwise we update the trajectory and move on to the next iLQR iteration, which makes a new quadratic approximation around the updated trajectory.

We continue iLQR iterations until convergence or up to 5 iterations. Convergence is defined by a threshold (0.05) on the change in control vector magnitude between iterations.

C.3 Cost function details

The cost function is a weighted sum of a set of handcrafted cost terms for penalizing collisions, distance to the goal, lateral lane deviation, lane heading deviation, and control effort:

$$C(\cdot; w) = w_1 C_\mathrm{coll}(s, \hat{s}_{a\in A}) + w_2 C_\mathrm{g}(s, g) + w_3 C_{\ell\perp}(s, m) + w_4 C_{\ell\measuredangle}(s, m) + w_5 C_u(u).$$

The cost function terms are defined as followed:

- $C_\mathrm{coll}(s, \hat{s}_{a\in A}) = \sum_{a\in A} \sum_{t\in 1:T} \varphi \left(\sum_{k\in K} \pi_k ||s^{(t)} - \hat{s}_{a,k}^{(t)}||^2 \right)$, the collision term, where $\hat{s}_{a,k}$ is the k-th mode of the predicted trajectory distribution for agent a, π_k is the probability of the k-th mode, φ is a Gaussian radial basis function, and $||\cdot||$ is the Euclidean norm. The collision term encourages keeping distance from other agents in expectation.
- $C_\mathrm{g}(s, g) = ||s^{(T)} - g||^2$, the goal term, that captures the distance to goal at the end of the planning horizon. We choose the goal g to match the "ground-truth" ego trajectory in the dataset.
- $C_{\ell\perp}(s, m) = \sum_{t\in 1:T} ||s^t - s_{\ell,\perp}||^2$, the lateral lane deviation term, where $s_{\ell,\perp}$ is the position on the lane closest to the ego position s.
- $C_{\ell\measuredangle}(s, m) = \sum_{t\in t:T} (h_\mathrm{e}^t - h_\ell)^2$, the lane heading deviation term, where h_e and h_ℓ denote ego vehicle heading and lane heading, respectively.
- $C_u(u) = \sum_{t\in t:T} \left((u_\mathrm{steer}^{(t)})^2 + (u_\mathrm{acc}^{(t)})^2 \right)$, the control effort term, where u_steer denotes the steering control and u_acc the acceleration control.

The default weight terms are chosen manually as follows: w_1=5.0; w_2=0.5; w_3=0.3; w_4=0.3; w_5=1.0.

D Experimental setup details

D.1 Dataset details

We use the nuScenes dataset which contains a large amounts of driving data, 1000 scenes with annotated states for vehicles and pedestrians, and a map containing lane information. The data is collected in Boston and Singapore. Scenes are 20s long, and trajectories are recorded at 2Hz (Δt=0.5s). We process the dataset into train and test scenarios. Each scenario has $H = 4$s of state history and $T = 3$s of GT future states for all agents. We choose one vehicle in the scene to act as the ego vehicle, and one vehicle to perform prediction for, such that ego and the predicted vehicle are close. For simplicity we use the GT futures for other agents for the purpose of planning. We train models with 75% of the training split, and use the remaining 15% as a validation set. We skip data with insufficient annotations to form a scenario, and where our planner has less then 2 trajectory candidates.

D.2 Training details

We train all models for 20 epochs using the Adam optimizer with gradient clipping, batch size of 256, learning rate of 0.003 by default. We have not performed extensive search over these training parameters. When training for multiple objectives, $\mathcal{L} = \alpha_1 \mathcal{L}_{\text{pred}} + \alpha_2 \mathcal{L}_{\text{plan}} + \alpha_3 \mathcal{L}_{\text{ctr}}$, we use the following default loss scaling parameters: α_1=1, α_2=100, α_3=1000 for reinforcement learning experiments, and α_1=1, α_2=10, α_3=1000 for imitation learning experiments. We choose the default values such that the average magnitude of the resulting gradients are similar, followed by a simple hyperparameter search on the validation set with a different, fixed random seed. We explore the effect of the α_i loss scaling parameters in an ablation study presented in Table 6.

D.3 Closed-loop simulation details

In our log-replay simulation setup ego states are unrolled based on the planned control outputs, while non-ego agents follow their fixed trajectories recorded in the dataset, and replan every $0.5s$. We create $T_{\text{sim}} = 10s$ long evaluation scenarios, one for each scene in our validation set. In each step of the simulation we run the stack to get control commands, update the ego state based on the control command and the known dynamics, and update non-ego agent states according to their logged trajectories in the dataset. The goal and lane inputs for planning are updated in each simulation step based on the logged ego trajectory. For simplicity in the stack we only perform prediction for the most relevant non-ego agent, and ignore all other agents during planning and control. We consider all agents when computing evaluation metrics.