SEAM: SHARED-EXPERT ADAPTATION MASKED BY CLOSED-LOOP SCORES IN REWARD FINE-TUNED DIFFUSION PLANNERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reward fine-tuning is an appealing way to optimize safety and comfort in diffusion-based driving planners, but a higher training reward need not mean better closed-loop driving. We fine-tune the Diffusion Planner by backpropagating through the 10-step DPM-Solver++ that it runs at deployment, on two independent stacks, Autoware and nuPlan. The optimized training reward rises by a factor of 2.5, yet on the nuPlan val14 closed-loop score no fine-tuned configuration improves on the behavior-cloning checkpoint beyond seed variation. The planner is nonetheless learning. Using Shared-Expert LoRA (SE-LoRA), which adds one low-rank adapter shared by all driving domains and one adapter per domain, we show that the adaptation is structured. Domains with little training data are carried by the shared adapter, while domains with more data rely increasingly on their own. Removing the shared adapter collapses the smallest domain even though more parameters are trained, and removing the per-domain adapters costs the largest domain 15 percent. The closed-loop score misses this because the per-scenario changes are small, point both ways, and barely change how many scenarios pass or fail the $0/1$ gates that the score is built from. For reward fine-tuning, this means a rising training reward is not evidence of better driving, and a flat closed-loop score is not evidence that nothing was learned.

1 INTRODUCTION

Diffusion models Ho et al. (2020); Chi et al. (2023) have become a common choice for trajectory planning in autonomous driving Zheng et al. (2025); Liao et al. (2025); Xing et al. (2025). They are pretrained by behavior cloning (BC), and recent work fine-tunes them with rewards that encode safety, comfort, and rule compliance Black et al. (2023); Fan et al. (2023); Ren et al. (2025). However, a higher training reward does not necessarily mean better closed-loop driving. On the Diffusion Planner Zheng et al. (2025), reward fine-tuning raises the optimized reward by a factor of 2.5, while the nuPlan closed-loop score stays within seed variation of the BC checkpoint. We therefore ask what the network learned while the benchmark did not move.

To make sure the flat score is not an artifact of how the planner was trained, we first remove the most likely confound, namely a mismatch between the sampler used in training and the one used at deployment. Reward fine-tuning of diffusion policies usually computes gradients through a stochastic sampler such as DDPM or DDIM, while the deployed planner runs a deterministic ODE solver such as DPM-Solver++ to meet real-time limits. We instead unroll the 10-step DPM-Solver++ that the planner runs at deployment into a differentiable graph, so that the reward gradient flows through the same computation that produces the deployed trajectory (§3.2). This unrolled solver is an instrument for the measurement and not a way to raise the score. Matching the samplers leaves the closed-loop score unchanged (§4.2).

Despite the flat benchmark, the planner is adapting, and the adaptation has structure. We fine-tune with Shared-Expert LoRA (SE-LoRA), which adds to each adapted weight one low-rank adapter shared by all domains and one low-rank adapter per domain, that is, per map or city (§3.1). This lets us remove either half and see what it carried. Removing the shared adapter collapses the domain with the least training data, although more parameters are trained. Removing the per-domain adapters

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

instead costs the domain with the most data. How much a domain relies on its own adapter therefore tracks how much data it has.

The benchmark misses this adaptation for two reasons. First, the per-scenario changes are small and point both ways. Across the 14 scenario types of val14, the configurations that tie the baseline move each type by at most 0.01 in either direction. Second, the nuPlan score multiplies $0/1$ gates such as collision and drivable-area compliance, so a trajectory that improves without crossing a gate adds nothing to the total. To check how closely our smooth training reward follows this score, we built PCDR (Per-Closed-Loop Differentiable Reward), a re-implementation of three nuPlan scoring operators, and we use it only as a measuring instrument (Appendix B.1). The same pipeline also runs on a production ROS 2 vehicle stack (Appendix D).

What we claim, and where each part is shown. We argue one proposition in four parts.

- The optimized training reward rises (§4.2).
- The closed-loop score does not move beyond seed variation, and every fine-tuned configuration ties the baseline or falls below it (Table 2).
- What the network learned is organized into a shared adapter and per-domain adapters, and removing either one costs reward in a different domain (§4.3, Figure 3).
- The closed-loop score stays flat because the per-scenario changes are small, point both ways, and barely change how many scenarios pass or fail the score's $0/1$ gates (§4.4).

The differentiable solver and PCDR are the instruments that make these measurements possible. We describe them in §3 and report their validation in the appendix.

This is a measurement study, and we make no state-of-the-art claim. We release the SE-LoRA code, the differentiable solver, and PCDR so that others can repeat the measurement on their own planners.

2 Related Work

Reward fine-tuning of diffusion planners. Several recent methods fine-tune diffusion policies and planners with rewards, including DPPO Ren et al. (2025), DiffusionDriveV2 Zou et al. (2025), PlannerRFT Li et al. (2026), and Plan-R1 Tang et al. (2025). These methods compute gradients through a stochastic sampler such as DDPM or DDIM during training, while the deployed planner runs a deterministic solver. DRaFT Clark et al. (2024) backpropagates through DDIM for image generation. We apply the same pathwise route to the deterministic DPM-Solver++ that the planner runs at deployment, so that training and deployment use one sampler.

Differentiable proxies for closed-loop metrics. Closed-loop driving metrics are built from non-differentiable thresholds, so every training method has to approximate them. Hydra-MDP and Hydra-MDP++ Li et al. (2024b; 2025) distill the NAVSIM PDM score into per-trajectory classification heads, which keeps a residual error against the true metric. Diffusion-ES Yang et al. (2024) avoids gradients altogether and searches with an evolutionary strategy. PCDR instead re-implements three nuPlan scoring operators, namely collision, progress, and driving-direction compliance. Its collision indicator reproduces the official polygon check on every evaluation cell, and the other two match to machine precision (Appendix B.1). We use it as a measuring instrument, namely to check how far a smooth surrogate of a $0/1$ safety gate carries a gradient (Appendix A.1), and not as a training target.

Shared and per-task adapters. Splitting parameters into a shared part and a per-task part has a long history in multi-task learning James & Stein (1961); Daumé (2007); Baxter (2000); Ando & Zhang (2005). In parameter-efficient fine-tuning, MoLA Gao et al. (2024), MixLoRA Li et al. (2024a), and MoLoRA Shah & Wagle (2026) combine several LoRA experts through a learned per-token router. SE-LoRA differs in two ways. It routes each whole scene by its map identifier, with no learned router, and it keeps one adapter that every domain updates. Our contribution is therefore the measurement rather than the update rule. We show that after reward fine-tuning across several domains, removing the shared adapter and removing the per-domain adapters each cost reward, in different domains. The differentiable solver is an instance of deep unrolling Gregor & LeCun

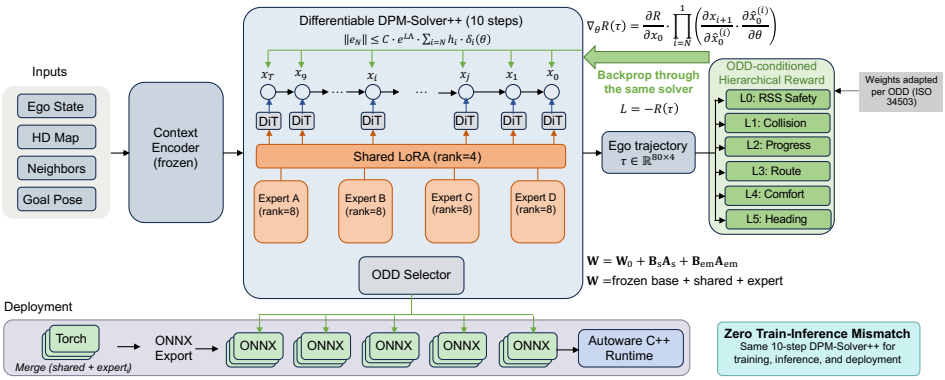


Figure 1: The same 10-step DPM-Solver++ runs in training and at deployment, and the reward gradient flows back through all ten steps into the shared and per-domain adapters.

(2010); Monga et al. (2021), and backward error analysis Hairer et al. (2006) bounds how its error grows.

3 METHOD

The method has one probe and two instruments. The probe, SE-LoRA, separates what the fine-tuned planner learns for all domains from what it learns for one domain. The first instrument, an unrolled DPM-Solver++, makes the training gradient pass through the sampler used at deployment. The second, PCDR, measures how far our smooth training reward departs from the nuPlan closed-loop score.

3.1 SHARED-EXPERT LORA

A domain here is one map or city, which the driving literature calls an operational design domain. To measure how much of the adaptation is shared across domains and how much belongs to one domain, SE-LoRA adds two low-rank updates to each adapted weight of the frozen planner. For a frozen weight matrix $\mathbf{W}_0$ and a scene from domain m, the adapted weight is

$$\mathbf{W} = \mathbf{W}_0 + \underbrace{\mathbf{B}_s\mathbf{A}_s}_{\text{shared}} + \underbrace{\mathbf{B}_{e_m}\mathbf{A}_{e_m}}_{\text{domain } m}. \tag{1}$$

The shared pair $\mathbf{B}_s, \mathbf{A}_s$ is updated by scenes from every domain, while the expert pair $\mathbf{B}_{e_m}, \mathbf{A}_{e_m}$ is updated only by scenes from domain m. The map identifier of the scene selects the expert, in training and at deployment, so no router is learned. We adapt the 16 linear layers of the diffusion transformer's MLP blocks and output projection, with shared rank r_s=4 and expert rank r_e=8. We chose this split by sweeping how a fixed total rank of 12 is divided (§4.3).

3.2 DIFFERENTIABLE DPM-SOLVER++

Score-function estimators such as REINFORCE and GRPO compute gradients through a stochastic sampler, whereas the deployed planner runs DPM-Solver++ for real-time inference. A per-step difference between the two samplers grows over the solver steps, and the discrete Gronwall inequality bounds that growth (Appendix A.2). To remove this mismatch, we unroll the deployment solver itself. From a fixed initial noise, DPM-Solver++ is a deterministic function $f_\theta : \boldsymbol{\epsilon} \mapsto \mathbf{x}_0$, so unrolling its 10 steps into one computational graph lets a differentiable reward backpropagate to the adapters (Figure 1). The resulting pathwise gradient is exact for the deployed solver. It also avoids the high-variance score-function gradient, which diverged in our REINFORCE baseline (Appendix C.4).

Table 1: The training reward rises in every domain and every seed. Autoware training reward by epoch, mean ± standard deviation over four seeds. Training-set size per domain is given in the header.

Epoch	**Overall**	**Map A** (2,663)	**Map B** (1,865)	**Map C** (172)	**Map D** (1,496)
1	147.0 ± 9.0	43.4 ± 5.2	280.8 ± 13.3	-80.4 ± 0.3	190.6 ± 12.5
2	320.2 ± 2.4	131.6 ± 1.1	530.9 ± 2.2	35.9 ± 15.2	426.1 ± 4.7
3	341.0 ± 2.2	148.9 ± 3.0	549.4 ± 1.4	133.8 ± 14.2	447.1 ± 1.3
5	355.7 ± 2.4	168.3 ± 3.2	556.4 ± 1.5	202.4 ± 9.4	456.6 ± 1.6
7	363.9 ± 2.5	180.8 ± 3.6	560.1 ± 1.6	225.2 ± 4.7	461.3 ± 1.8
10	$\mathbf{372.2 \pm 3.1}$	194.5 ± 4.4	563.2 ± 1.9	242.1 ± 5.2	465.2 ± 2.3

3.3 Training Reward and the PCDR Scoring Mirror

We train with a continuous, domain-weighted reward $R_m = r_{\text{rss}} + \mathbf{w}_m^\top [r_{\text{col}}, r_{\text{route}}, \dots]$. The first term rewards states that satisfy Responsibility-Sensitive Safety (RSS) distances, and the weighted terms score collision margin, route following, progress, and comfort, with weights $\mathbf{w}_m$ fixed per domain before training (Appendix B.2). The nuPlan closed-loop score, by contrast, multiplies $0/1$ gates, so it changes only when a trajectory crosses a threshold and it cannot be differentiated. To measure how far the smooth reward departs from that score, we built PCDR (Per-Closed-Loop Differentiable Reward), which re-implements the collision, progress, and driving-direction operators of the nuPlan scorer and reproduces their official values (Appendix B.1). We also use PCDR to measure how much of a smooth surrogate's gradient reaches each safety gate (Appendix A.1).

4 Experiments

4.1 Setup

We fine-tune the released Diffusion Planner checkpoint Zheng et al. (2025) on two stacks with different roles. The Autoware stack gives a controlled setting with four high-definition maps reconstructed from real LiDAR surveys. Map A is a dense urban campus with 2,663 training scenes, Map B a suburban site with 1,865, Map C a test track with 172, and Map D a second campus with 1,496. Because these domains differ in size by more than a factor of ten, this stack also lets us ask how the adaptation depends on the amount of data. The Map C scenes were regenerated with a corrected map projection, and all Map C results use the regenerated scenes (Appendix C.8). The nuPlan stack tests whether the findings hold at scale. There we fine-tune on 20,000 scenes from four cities and report the closed-loop score with non-reactive agents on the official val14 split of 1,118 scenarios. The fine-tuning scenes come from the nuPlan validation split, which also contains the val14 logs (Appendix C.9). Any overlap would favor the fine-tuned models, and none of them beats the baseline. Every run uses learning rate 2×10^{-6} for 10 epochs on a single GPU.

4.2 The Training Reward Rises and the Benchmark Does Not

Fine-tuning raises the training reward by a factor of 2.5, yet no configuration improves on the behavior-cloning checkpoint on val14. On Autoware, the domain-weighted training reward rises from 147.0 ± 9.0 after the first epoch to 372.2 ± 3.1 after the tenth, averaged over four seeds, and it rises in every domain and every seed (Table 1).

This rise does not reach the benchmark. Table 2 compares several fine-tuning configurations of the same planner, each trained with a different reward. The best of them, a smooth copy of the closed-loop score, reaches 0.8975 ± 0.0013 against 0.8973 for the baseline, a difference of 0.0002 with a seed standard deviation of 0.0013. Every other configuration scores below the baseline. The baseline is the released checkpoint, and each fine-tuned row is the mean of three seeds.

Matching the training sampler to the deployed one does not change this either. The same model fine-tuned through DPM-Solver++ and through the mismatched DDIM sampler scores within 0.0013 on val14, a paired t-test gives p=0.55, and the scenarios that each version fails outright overlap by 96 percent (Appendix E). Switching the sampler moves the score between scenario types rather than

Table 2: nuPlan val14 for several fine-tuning configurations of the same base planner. "Catastrophic" counts scenarios scoring exactly zero, "Perfect" counts those scoring one. The four upper rows are single checkpoints and the three lower rows are three seeds each (n=1118), so the comparison that matters is between one point and a three-seed spread. No configuration improves on the baseline. The best ties it, and the rest regress. The last row is the instructive one, making 29 more scenarios perfect and 8 more catastrophic at the same time, which is the cancellation discussed in §4.4. Equivalence testing is in Table 3 and the per-scenario breakdown in Appendix E.

Configuration	**Reward**	**CL Score ↑**	**Cat. ↓**	**Perfect ↑**
Behavior cloning baseline	none	0.8973	**61**	548
Fine-tuned through DPM-Solver++	soft safety margin	0.8849	76	543
Fine-tuned through DDIM	soft safety margin	0.8836	76	535
DPPO, matched compute	policy gradient	0.8960	63	547
Smooth copy of the score	smooth proxy	$\mathbf{0.8975 \pm 0.0013}$	62.0 ± 2.6	547.7 ± 4.5
Smooth copy plus margin	proxy + exact margin	0.8964 ± 0.0013	63.0 ± 1.4	551.0 ± 1.4
Exact score operators	bit-exact	0.8925 ± 0.002	69	**577**

Table 3: Pre-specified paired TOST equivalence Schuirmann (1987); Lakens (2017) on val14 CLS (n=1133, DPM-Solver++ vs. DDIM). Both one-sided t-tests must reject at α=0.05 to conclude equivalence at margin $\pm\Delta_{\text{eq}}$; observed 95% CI on paired difference is $[-0.0029, +0.0055]$.

Δ_{eq} (CLS)	t_{lower} (p)	t_{upper} (p)	TOST p	verdict at α=0.05
±0.003	+2.00 (0.023)	−0.81 (0.211)	0.211	inconclusive
±0.005	+2.94 (0.002)	−1.74 (0.041)	**0.041**	**equivalent**
±0.010	+5.28 (<0.001)	−4.08 (<0.001)	**<0.001**	**equivalent**
±0.015	+7.62 (<0.001)	−6.42 (<0.001)	**<0.001**	**equivalent**

in total. Relative to fine-tuning through DDIM, fine-tuning through DPM-Solver++ gains 0.013 on *stopping with lead* and loses 0.016 on *starting right turn*. Because matching the samplers leaves the score unchanged, sampler mismatch cannot explain the flat score. The question therefore becomes what changed inside the network.

The equivalence also holds under a pre-specified two one-sided test (Table 3), and Figure 2 shows what the sampler does to single scenes. Scenes where DPM-Solver++ and DDIM trajectories diverge from the same initial noise are the scenes where the mismatched sampler scores badly, with r=−0.91 over 35 scenes. Divergence does not, however, predict which sampler scores better, so it marks scenes that are hard for a mismatched sampler rather than a gain from matching it.

4.3 Reliance on Per-Domain Adapters Grows with Training Data

Despite the flat closed-loop score, the planner learns an adaptation with a clear structure. Domains with little training data are carried by the shared adapter, while domains with more data rely increasingly on their own. To show this, we remove one half at a time. We split a fixed total rank of 12 three ways between the shared adapter and the four per-domain adapters and retrain each split from the same checkpoint on the same 6,196 Autoware scenes (Figure 3).

Without the shared adapter, Map C does not learn. Its training reward ends at 2.0 and −10.8 in two seeds, against 240 with SE-LoRA, although this variant trains a third more parameters. The collapse is therefore not a lack of capacity. Without the per-domain adapters, the cost moves to Map A, which loses 15 percent of its reward, while Map C ends slightly higher. Four further SE-LoRA runs with different seeds give Map A 194.5 ± 4.4 and Map C 242.1 ± 5.2, which agrees with the sweep.

The amount of data in a domain explains this pattern. Map C has only 172 training scenes, too few to train an adapter of its own, so the shared adapter carries it almost fully. Map A has 2,663 scenes and loses 15 percent of its reward without its own adapter. Holding a domain out of fine-tuning shows the same ordering. We route the held-out domain through the shared adapter alone and compare its gain with the gain it receives when it is trained on. Held-out domains retain 76 to 96 percent of that gain, and Map C retains the most (Table 4; the protocol is in Appendix C.7).

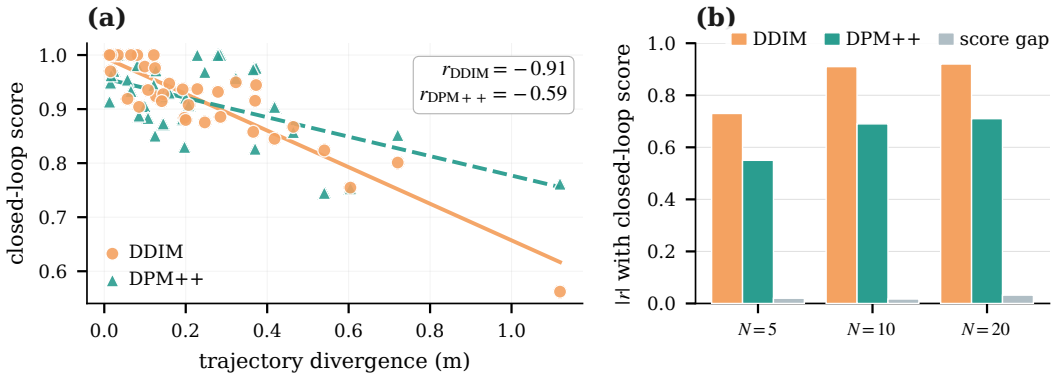


Figure 2: Per-scene solver divergence against closed-loop score on n=35 matched val14 scenes. **(a)** Trajectory L2 between DPM-Solver++ and DDIM against the DDIM score, r=−0.91; against the DPM-Solver++ score, r=−0.69. **(b)** The same divergence against the gap between the two scores, where the relationship disappears.

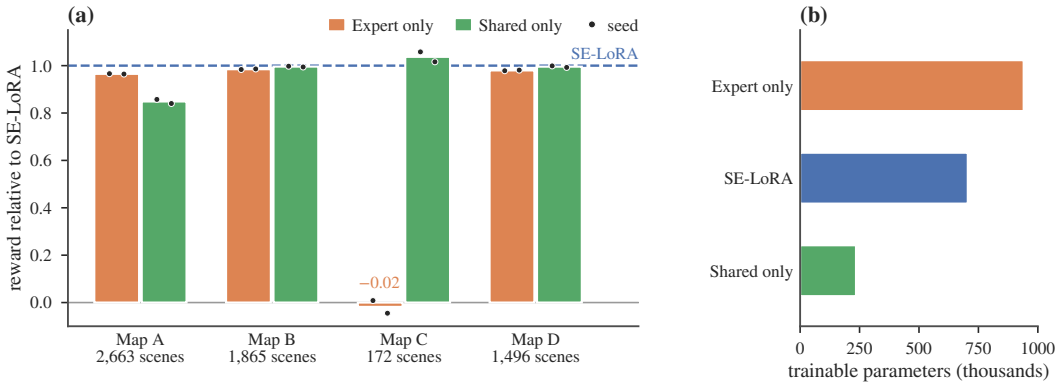


Figure 3: Removing either half of the adapter costs reward in a different domain. **(a)** Training reward of each ablation relative to SE-LoRA in the same domain, one dot per seed. Without the shared adapter, Map C, which has only 172 training scenes, falls to about zero in both seeds. Without the per-domain adapters, Map A, which has the most data, loses 15 percent, and no other domain loses more than 4 percent. **(b)** The variant that collapses trains the most parameters, a third more than SE-LoRA, so the collapse is not a lack of capacity. Absolute values are in Table 10.

Table 4: A domain never seen during fine-tuning keeps most of the gain, and the domain with the least data keeps the most. "Retains" is the held-out gain over the same domain's gain when it was a training domain, with a 2,000-resample scene-level bootstrap interval. Gains are measured against the same base checkpoint with untrained adapters.

Domain	Train scenes	Gain, held out	Gain, trained on	Retains
Map A (dense urban)	2,663	+158.1	+207.0	0.76 [0.68, 0.84]
Map B (suburban)	1,865	+154.6	+199.1	0.78 [0.72, 0.84]
Map C (test track)	172	+148.7	+154.9	**0.96** [0.93, 0.99]
Map D (campus)	1,496	+178.1	+192.4	0.93 [0.80, 1.04]

Two further checks are in the appendix. Patching one domain's adapter into another domain's scenes shifts the planned trajectory in a direction specific to that adapter. For example, on a single nuPlan checkpoint the Pittsburgh adapter raises jerk by 132 m/s^3 while the Singapore adapter lowers speed (Table 5).

Moving the per-domain adapters out of the backbone into separate heads keeps the two halves separable (Appendix C.11). We do not use gradient similarity between adapters as evidence, because

Table 5: Activation patching on nuPlan, single fine-tuned checkpoint, 95% bootstrap intervals over n=90 foreign-scene deltas per adapter. The two adapters move different metrics.

Patched adapter	Δ Speed (m/s)	Δ Jerk (m/s^3)
Singapore	$\mathbf{-0.19}$ $[-0.22, -0.17]$	$+20$ $[-56, +63]$
Pittsburgh	$+0.04$ $[-0.00, +0.08]$	$\mathbf{+132}$ $[+36, +238]$

the gradients of independently initialized adapters live in unrelated coordinate frames. Measured in one common frame, the in-backbone adapters do not separate (Appendix F).

4.4 Safety Improvements Count Only When They Flip a 0/1 Gate

The closed-loop score multiplies 0/1 gates such as collision and drivable-area compliance. A trajectory that becomes safer without crossing a gate therefore adds nothing to the score, while a trajectory pushed across a gate loses the whole scenario. Table 2 shows this. Training on the exact score operators makes 29 more scenarios perfect than the baseline, 577 against 548, and at the same time drives 8 more scenarios to a score of zero, 69 against 61. The 29 gains are worth about $+0.003$ on the aggregate and the 8 failures about -0.006, so the benchmark reports a net loss.

The configurations that tie the baseline barely change these counts. Their number of perfect scenarios stays within 3 of the baseline's 548, and their number of zero-score scenarios within 2 of its 61. Figure 5 shows the same picture by scenario type. These configurations move each of the 14 types by at most 0.01, in both directions, and the smooth copy of the score, for example, gains up to 0.009 on some types and loses up to 0.004 on others. Where fine-tuning moves a type by more, it moves it down. With the soft safety-margin reward, the planner falls below the baseline on 10 of the 14 types, most on *starting left turn*, by 0.058, and on *behind long vehicle*, by 0.049. This is why those two rows of Table 2 regress. For reward fine-tuning, this means a safety improvement shows up in the score only when it flips a gate, while a single wrong flip costs a full scenario.

Training directly on the exact 0/1 operators does not remove this problem, because an exact gate gives no gradient almost everywhere (Figure 4). A smooth surrogate restores a gradient, but mainly for the tightest active constraint, so the other constraints receive little of it (Appendix A.1). This is consistent with the exact-operator row of Table 2, which gains perfect scenarios and loses others at the same time.

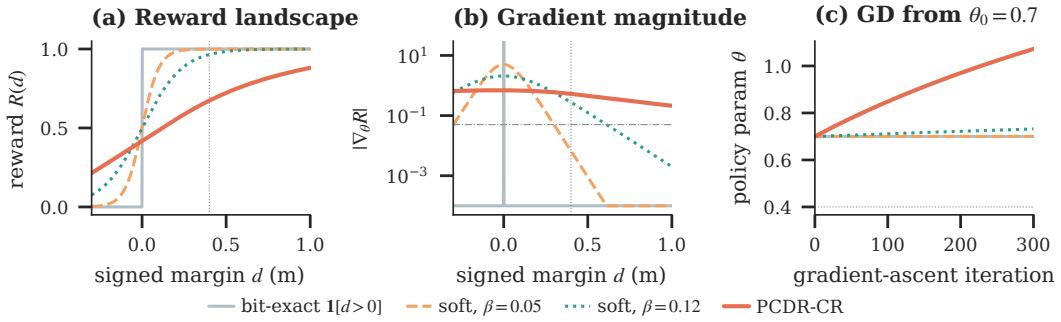


Figure 4: One-dimensional illustration with K=1: four reward parameterizations of the same signed margin $d(\theta) = \theta$ with target buffer ρ_{NC}=0.40 m. **(a)** The bit-exact indicator is piecewise constant. The soft-minimum curves shown at β=0.05 flatten once the margin is clipped at the buffer, which is a property of the clipping we chose rather than a property of smoothing in general. **(b)** Gradient magnitude on a log scale; the bit-exact reward has zero gradient almost everywhere. **(c)** Gradient ascent from θ_0=0.7 under each parameterization.



Figure 5: Per-scenario closed-loop score on val14 for the 14 scenario types, ordered from hardest to easiest for the baseline, with the overall score in the last row. There is one run per configuration. Each cell gives the score, and its color gives the change against the behavior-cloning baseline, blue for better and red for worse. The two configurations that tie the baseline stay within 0.01 of it on every type, while the soft safety-margin reward loses most on turns from standstill and behind long vehicles.

5 Conclusion and Limitations

Reward fine-tuning changes what a diffusion planner does even when the closed-loop benchmark does not move. Matching the training sampler to the deployed one rules out sampler mismatch as the cause. The change is structured, since domains with little data are carried by the shared adapter while domains with more data rely increasingly on their own. The benchmark misses it because its score counts a safety improvement only when the improvement flips a $0/1$ gate. For reward fine-tuning of planners, this means a rising training reward is not evidence of better driving, and a flat aggregate closed-loop score is not evidence that nothing was learned. We therefore recommend reporting per-scenario scores and gate-flip counts next to the aggregate whenever a planner is fine-tuned with rewards.

Limitations. We choose the split between shared and per-domain rank by sweeping, and we have no rule that predicts it. Our measurements show where each half of the adapter matters and relate this to the amount of data per domain. Which driving behaviors each half encodes, such as yielding or how sharply the planner steers, is a question for future work.

Broader impact. A higher training reward, or visible specialization inside a model, is not a substitute for closed-loop evaluation of a driving planner. A fine-tuned planner can also lose far more on the hardest scenario types than its aggregate score shows. With the soft safety-margin reward, the score on *starting left turn* falls by 0.058 while the aggregate falls by 0.012 (Figure 5). Reporting per-scenario scores next to the aggregate would make such changes visible.

Reproducibility Statement

All runs start from the publicly released Diffusion Planner checkpoint. Section 4.1 lists the data splits and hyperparameters, Appendix B.2 gives the reward weights, and Appendix C.1 and Appendix C.6 report per-seed values. The Autoware Map C scenes require a corrected map projection, which Appendix C.8 describes. We will release the SE-LoRA code, the differentiable solver, PCDR, and the scene-conversion fix upon publication.

Author Contributions

Acknowledgments

Use of AI Tools

Generative AI tools (large language model assistants) were used substantially in this work under author direction, namely to draft and revise manuscript text, to write and debug analysis and experiment code, and to assist with literature search and formatting. All experiments were designed by the authors and executed on the authors' own infrastructure; all datasets, model checkpoints, and numerical results originate from the authors' pipelines. Every AI-drafted passage was reviewed and approved by the authors, and all reported numbers were verified against raw experiment outputs. During this process, one analysis in an earlier version of this work (a gradient-cosine comparison) was found to be confounded by a measurement artifact and is retracted and corrected in Appendix F. The authors take full responsibility for all content.

References

R Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, 1 December 2005. ISSN 1532-4435,1533-7928. URL http://www.jmlr.org/papers/volume6/ando05a/ando05a.pdf.

Jonathan Baxter. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training Diffusion Models with Reinforcement Learning. In *The Twelfth International Conference on Learning Representations*, 13 October 2023. URL https://openreview.net/forum?id=YCWjhGrJFD.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, 10 July 2023. doi: 10.15607/rss.2023.xix.026. URL http://dx.doi.org/10.15607/rss.2023.xix.026.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly Fine-Tuning Diffusion Models on Differentiable Rewards. In *The Twelfth International Conference on Learning Representations*, 13 October 2024. URL https://openreview.net/forum?id=1vmSEVL19f.

Hal Daumé, III. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 256–263, 2007. URL https://aclanthology.org/P07-1033/.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2 November 2023. URL https://openreview.net/forum?id=8OTPepXzeh.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and V S Subrahmanian. Higher layers need more LoRA experts. *arXiv [cs.CL]*, 13 February 2024. doi: 10.48550/arXiv.2402.08562. URL http://arxiv.org/abs/2402.08562.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 399–406. Omnipress, 21 June 2010. doi: 10.5555/3104322.3104374. URL https://dl.acm.org/doi/10.5555/3104322.3104374.

Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: Structure-preserving algorithms for ordinary differential equations*. Springer Series in Computational Mathematics. Springer, Berlin, Germany, 2 edition, 22 February 2006. ISBN 9783540306634. doi: 10.1007/3-540-30666-8. URL http://dx.doi.org/10.1007/3-540-30666-8.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

W James and Charles Stein. Estimation with Quadratic Loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4.1, pp. 361–379. University of California Press, 1 January 1961.

Daniël Lakens. Equivalence tests: A practical primer for t tests, correlations, and meta-analyses: A practical primer for *t* tests, correlations, and meta-analyses. *Soc. Psychol. Personal. Sci.*, 8 (4):355–362, May 2017. ISSN 1948-5506,1948-5514. doi: 10.1177/1948550617697177. URL http://dx.doi.org/10.1177/1948550617697177.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing Large Language Models fine-tuning with LoRA-based Mixture of experts. *arXiv [cs.CL]*, 21 April 2024a. doi: 10.48550/arXiv.2404.15159. URL http://arxiv.org/abs/2404.15159.

Hongchen Li, Tianyu Li, Jiazhi Yang, Haochen Tian, Caojun Wang, Lei Shi, Mingyang Shang, Zengrong Lin, Gaoqiang Wu, Zhihui Hao, Xianpeng Lang, Jia Hu, and Hongyang Li. PlannerRFT: Reinforcing diffusion planners through closed-loop and sample-efficient fine-tuning. *arXiv [cs.RO]*, 19 January 2026. doi: 10.48550/arXiv.2601.12901. URL http://arxiv.org/abs/2601.12901.

Kailin Li, Zhenxin Li, Shiyi Lan, Yuan Xie, Zhizhong Zhang, Jiayi Liu, Zuxuan Wu, Zhiding Yu, and Jose M Alvarez. Hydra-MDP++: Advancing end-to-end driving via expert-guided hydra-distillation. *arXiv [cs.CV]*, 17 March 2025. doi: 10.48550/arXiv.2503.12820. URL http://arxiv.org/abs/2503.12820.

Zhenxin Li, Kailin Li, Shihao Wang, Shiyi Lan, Zhiding Yu, Yishen Ji, Zhiqi Li, Ziyue Zhu, Jan Kautz, Zuxuan Wu, Yu-Gang Jiang, and Jose M Alvarez. Hydra-MDP: End-to-end multimodal planning with multi-target Hydra-distillation. *arXiv [cs.CV]*, 11 June 2024b. doi: 10.48550/arXiv.2406.06978. URL http://arxiv.org/abs/2406.06978.

Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. DiffusionDrive: Truncated diffusion model for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12037–12047. IEEE, June 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Liao_DiffusionDrive_Truncated_Diffusion_Model_for_End-to-End_Autonomous_Driving_CVPR_2025_paper.pdf.

Vishal Monga, Yuelong Li, and Yonina C Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Process. Mag.*, 38(2):18–44, March 2021. ISSN 1053-5888,1558-0792. doi: 10.1109/msp.2020.3016905. URL http://dx.doi.org/10.1109/MSP.2020.3016905.

Allen Z Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion Policy Policy Optimization. In *The Thirteenth International Conference on Learning Representations*, 4 October 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

D J Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *J. Pharmacokinet. Biopharm.*, 15(6):657–680, December 1987. ISSN 0090-466X,2168-5789. doi: 10.1007/bf01068419. URL http://dx.doi.org/10.1007/bf01068419.

Shrey Shah and Justin Wagle. MoLoRA: Composable Specialization via Per-Token Adapter Routing. *arXiv [cs.CL]*, March 2026. doi: 10.48550/arXiv.2603.15965. URL http://arxiv.org/abs/2603.15965.

Xiaolong Tang, Meina Kan, Shiguang Shan, and Xilin Chen. Plan-R1: Safe and feasible trajectory planning as language modeling. *arXiv [cs.RO]*, 23 May 2025. URL http://arxiv.org/abs/2505.17659.

Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. GoalFlow: Goal-Driven Flow Matching for Multimodal Trajectories Generation in End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1602–1611, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Xing_GoalFlow_Goal-Driven_Flow_Matching_for_Multimodal_Trajectories_Generation_in_End-to-End_CVPR_2025_paper.html.

Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-ES: Gradient-free planning with diffusion for autonomous driving and zero-shot instruction following. *arXiv [cs.LG]*, 9 February 2024. doi: 10.48550/arXiv.2402.06559. URL http://arxiv.org/abs/2402.06559.

Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 26 January 2025.

Jialv Zou, Shaoyu Chen, Bencheng Liao, Zhiyu Zheng, Yuehao Song, Lefei Zhang, Qian Zhang, Wenyu Liu, and Xinggang Wang. DiffusionDriveV2: Reinforcement learning-constrained truncated diffusion modeling in end-to-end autonomous driving. *arXiv [cs.CV]*, 8 December 2025. doi: 10.48550/arXiv.2512.07745. URL http://arxiv.org/abs/2512.07745.

Appendix Overview

This appendix provides theoretical derivations, diagnostic implementation details, and extended empirical results supporting the main text. To facilitate navigation, the contents are organized into four primary sections:

- **Appendix A: Analysis of the Instruments**. How far a smooth surrogate of a $0/1$ safety gate carries a gradient, and the discrete Gronwall bound on error growth under solver mismatch.
- **Appendix B: Diagnostic Tools and Implementation**. The operator-level mirror of the closed-loop scorer, the per-domain reward weights, and the step allocation method.
- **Appendix C: Extended Empirical Results**. Training curves, empty-road and solver sweeps, the per-layer and expert-swap probes, the corrected transfer study, the placement control, and three sections on things that went wrong: a map-projection fault in the Map C scenes, overlap between the fine-tuning pool and the evaluation split, and a multi-seed run that exploited the reward and cannot be used.
- **Appendix D: Real Vehicle Deployment**. Vehicle configuration, pipeline wiring, and a paired offline trajectory comparison. This documents that the pipeline runs in production; it is not evidence for the claim.
- **Appendix E and F: Per-Scenario Breakdown and Gradient Geometry**. The full per-scenario table and equivalence test, and a note on why gradient similarity between independently initialized adapters is not a valid measure.

A Analysis of the Instruments

This section analyses the two instruments the main text relies on: how far a smooth surrogate of a $0/1$ safety gate carries a gradient, and how fast trajectory error grows when training and deployment use different solvers.

A.1 How Far a Smooth Surrogate Carries a Gradient

Closed-loop driving scores are products of $0/1$ gates: a collision, a departure from the drivable area, or a wrong-way violation sets the whole scenario to zero, and a trajectory that does not cross a gate contributes nothing no matter how much it improved. Training against such a score directly gives no gradient almost everywhere, so a differentiable objective has to replace each gate with something smooth. This appendix states what that replacement does and does not cost, because the main text appeals to it in §4.4.

Let $\tilde{R}_\beta(\theta) = -\beta \log \sum_{i=1}^{K} e^{-d_i(\theta)/\beta}$ be the log-sum-exp soft-minimum of an active set $\{d_i\}_{i=1}^{K}$ of signed margins with smoothing parameter $\beta > 0$, and let $d_{\min} = \min_i d_i$. Two elementary bounds hold.

1. **Forward error**: $\left|\tilde{R}_\beta - d_{\min}\right| \le \beta \log K$. Smoothing costs accuracy in proportion to β and to the number of simultaneously active constraints.
2. **Attention decay across constraints**: $\left\|\partial \tilde{R}_\beta / \partial d_i\right\| \le e^{-(d_i - d_{\min})/\beta}$ for any d_i strictly greater than $d_{\min}$. A constraint that is slack relative to the tightest one receives exponentially little of the gradient.

Holding forward error below ε requires $\beta \le \varepsilon / \log K$, and a small β is exactly what makes the second bound bite. The consequence is about *which* constraint the gradient reaches, not about whether any gradient survives: the tightest active margin always keeps a gradient of order one, and for $K{=}1$ the surrogate is the margin itself. What a small β buys in fidelity it pays for in breadth, so with many active constraints the objective effectively optimizes one of them at a time. Figure 4 illustrates the one-constraint case. The bounds therefore concern which constraint receives the gradient. They do not imply that a smooth surrogate loses its gradient far from the boundary.

A.2 Error Propagation under Solver Mismatch

When fine tuning utilizes a solver different from the deployment solver, the resulting parameters optimize a different dynamical system. We bound this divergence using backward error analysis.

Let the per step implementation discrepancy between two solvers be ϵ_0. Define the error at step i as $\mathbf{e}_i$. Applying the discrete Gronwall inequality over the sequence of N solver updates yields:

$$\|\mathbf{e}_N\| \le \epsilon_0 \cdot \sum_{i=0}^{N-1} c_i \cdot \prod_{j=i+1}^{N-1} (1 + c_j L) \tag{2}$$

where L is the Lipschitz constant of the network and c_i represents the solver coefficients. Assuming uniform step spacing and a constant Lipschitz bound, this constraint simplifies to exponential error growth bounded by $\epsilon_0 \cdot e^{cLN}$.

For typical network architectures where $L \approx 3$ and $N = 10$, a minute per step numerical discrepancy strictly amplifies into a macroscopic trajectory error exceeding standard anytime convergence thresholds. Our differentiable unrolling method eliminates this exponential divergence entirely by enforcing architectural equivalence between the training and deployment forward passes.

B Diagnostic Tools and Implementation

This section documents the reward implementations, the per-domain reward weights, and the step allocation method used throughout.

B.1 Closed Loop Metric Conformance Audit

We distinguish two objects that must not be conflated. First, the *soft training reward* (§3.3) is a differentiable surrogate: it replaces the scorer's hard thresholds with smooth sigmoids parameterized by physical distances and replaces polygon-intersection collision checks with a penetration-depth formulation. A per-metric conformance audit of this training reward against the standard closed-loop scoring formula shows the aggregate skeleton matches exactly (the same weighted average

multiplied by the no-collision, drivable-area, driving-direction, and progress gates, with identical weights), while individual components are mixed: some match the scoring formula in functional form, others are smooth approximations, and some omit route-aware floors or backward-progress penalties. This residual mismatch between training reward and deployment scoring is one mechanism contributing to the aggregate-score saturation of §4.2.

Second, PCDR is the separate, bit-exact construction: an operator-level mirror of the closed-loop scoring functions themselves (the collision indicator reproduced exactly against the scorer's polygon checks on all evaluation cells; the progress and direction-compliance operators at machine precision). PCDR is used as an audit reference and, in PCDR-CR, as a margin penalty added to the smooth proxy; it is not itself the smooth training objective. Because it is exact, its pathwise gradient vanishes almost everywhere, which is why it cannot serve as a training objective on its own and why a smooth proxy term is needed alongside it (Appendix A.1).

B.2 Domain Conditioned Reward Weight Profiles

Each domain utilizes a specific reward profile following standard operational design domain characteristics. The weight values in Table 6 were configured prior to training to reflect the structural priorities of each environment. Dense urban domains (Map A) emphasize route compliance and curve following due to complex intersections. Suburban domains (Map B) prioritize progress and speed on straight roads. The test track (Map C) emphasizes comfort due to tight track layouts.

Table 6: Domain conditioned reward weight profiles. Weights are configured to match specific structural priorities per map environment.

Weight Component	**Map A (Urban)**	**Map B (Suburban)**	**Map C (Track)**	**Map D (Campus)**
Progress	1.5	3.0	2.0	2.5
Route Deviation	3.0	1.5	2.5	2.0
Curvature	4.0	2.0	3.5	2.5
Comfort	1.0	1.0	2.0	1.5
Heading Alignment	3.0	1.5	2.0	2.0
Safety (RSS margin)	2.0	1.5	1.0	1.5

B.3 Adaptive Step Allocation

The global error bound is scene dependent. Complex scenes exhibit higher effective Lipschitz constants and require more denoising steps. We train a lightweight step budget predictor that maps the context encoder output to the minimum steps necessary to retain high reward fidelity. This predictor achieves significant compute reduction while maintaining trajectory quality. On the validation set, it achieves a Pareto improvement by dynamically allocating fewer steps to simpler suburban scenes compared to complex urban intersections.

C Extended Empirical Results

This section gives the full tables behind the main text: the training curves that show the optimized reward rising, the empty-road and solver sweeps, the per-layer and expert-swap probes, the corrected transfer study, and an account of the data fault and the failed multi-seed run that shaped what we can and cannot claim.

C.1 Training Curves on Autoware

Table 1 reports the domain-conditioned training reward per epoch, averaged over four seeds, on the corrected data of §C.8. The reward rises monotonically on every domain in every seed and has not saturated at epoch 10. This is the measurement behind the first part of the claim in §4.2; the absolute scale is set by the per-domain weights of §B.2 and is not comparable to the unweighted base reward used elsewhere.

C.2 Empty Road Evaluation

Table 7 reports the multi seed empty road evaluation across domains. Without dynamic obstacles, the baseline model already drives competently. The shared parameterization achieves statistically significant violation reduction, while the expert parameterization shows consistent lateral acceleration trends. These modest effect sizes are expected on empty roads, as the collision and safety reward components are not exercised.

Table 7: Empty road evaluation across three maps and four seeds. This evaluation predates the Map C projection fix of Appendix C.8. Bold indicates best performance per row. Statistical significance is computed versus the baseline via paired testing.

Metric	Baseline	Expert Component	Shared Component
Safety Violations (%) ↓	29.2 ± 0.5	28.8 ± 0.4	$\mathbf{28.5 \pm 0.5}$
Speed (m/s) ↑	$\mathbf{4.43 \pm 0.02}$	4.42 ± 0.03	$\mathbf{4.43 \pm 0.01}$
Route Deviation (m) ↓	5.78 ± 0.02	5.73 ± 0.15	$\mathbf{5.68 \pm 0.07}$
Jerk (m/s^3) ↓	37.5 ± 1.7	35.1 ± 1.3	$\mathbf{35.0 \pm 2.3}$
Lateral Acc. (m/s^2) ↓	0.77 ± 0.02	$\mathbf{0.74 \pm 0.02}$	0.76 ± 0.01

C.3 Complete Solver Sweep

Table 8 documents the complete per epoch training reward for all evaluated solvers under identical capacity limits and data splits. Deployment is evaluated exclusively under the second order deployment solver.

Euler achieves the highest training reward because the simple solver provides short gradient paths. However, it deploys poorly because the optimization direction is misaligned with the deployment trajectory. The stochastic solver diverges to numerical instability (NaN) due to gradient explosion through random noise samples. The matched solver converges steadily to the highest deployment quality because gradients are correctly aligned.

Table 8: Complete solver sweep training reward per epoch (single run per solver). Deployment verdict indicates qualitative performance under the matched deployment solver.

Training Solver	Ep.1	Ep.2	Ep.3	Ep.4	Ep.5	Deploy Verdict
Matched Deployment Solver	143.0	341.8	360.7	364.5	366.8	Best
Deterministic Alternate	160.8	244.1	272.7	288.8	302.7	Poor
Stochastic Sampler	NaN	NaN	NaN	NaN	NaN	Diverged
Euler Approximation	317.9	377.2	383.4	385.0	385.5	Worst

C.4 REINFORCE Baseline: Gradient Instability

We attempted a standard REINFORCE baseline using stochastic DDIM (η=0.3) for sampling and per-step log-probability computation:

$$\nabla_\theta \mathbb{E}[R] \approx R \cdot \sum_{t=1}^{N} \nabla_\theta \log p(x_{t-1}|x_t). \tag{3}$$

The training immediately diverged to NaN rewards on the first batch. This occurs because the log-probability $\log \mathcal{N}(x_{t-1}; \mu_\theta, \sigma^2 I)$ involves division by σ^2_{ddim}, which becomes very small at late timesteps, causing gradient magnitudes to explode.

This negative result provides additional evidence for our approach: not only does the deployment solver need to match (Appendix C.3), but the REINFORCE gradient estimator itself is unstable for diffusion models due to the high variance of per-step log-probabilities across the noise schedule. Our direct backpropagation approach avoids both issues by eliminating log-probabilities entirely.

C.5 Causal Probing: Per-Layer Attribution

Patching one domain's adapter activations into another domain's scenes changes the planned trajectory in a way that depends on which adapter is patched in. Table 5 reports this on nuPlan. We report it as a single-checkpoint result: the two adapters move different metrics, but we do not claim they form a mirrored pair, because the one multi-seed replication we were able to run produced the degenerate policy described in §C.10 and can neither confirm nor refute the pattern, and the Autoware replication below finds no mirrored pair.

On the Autoware stack the same protocol over four seeds finds the adapters behaviorally distinct but not mirrored: the dense-urban adapter is the outlier, the suburban and campus adapters are the same adapter to within noise (correlation 0.96 to 1.00 on every metric in every seed), and the Map C adapter barely trains at all. Measured on the four corrected-data checkpoints, the Frobenius norm of its adapter weights is 0.16 ± 0.01 against 0.42 to 0.67 for the other three domains, and in the swap study it behaves almost identically to using no adapter. That observation is what the corrected transfer result of §C.7 rests on.

Table 9 shows the per-layer causal effect when patching expert LoRA activations from high-curvature scenes into low-curvature scenes. The effect is concentrated in specific DiT layers:

Table 9: Per-layer causal attribution: speed change (m/s) when patching expert activation from high-curvature to low-curvature scene (single-case illustration; the systematic per-expert analysis is Table 5). Only layers with $|\Delta\text{speed}| > 0.1$ shown.

DiT Layer	ΔSpeed	ΔJerk	ΔLatAcc
`blocks.2.mlp1.fc1`	+0.355	+3.7	+0.051
`preproj.fc1`	+0.247	+9.2	−0.014
`final_layer.proj.4`	+0.472	+72.9	+14.802

The final projection layer (`proj.4`) carries the largest jerk and lateral acceleration effect (+72.9 and +14.8), suggesting it encodes the "output gain" of the trajectory. Block 2's MLP (`blocks.2.mlp1.fc1`) and the input projection (`preproj.fc1`) primarily affect speed, indicating they modulate the driving dynamics at a higher level. This mechanistic analysis is, to our knowledge, the first per-layer interpretability result for diffusion-based trajectory planners.

C.6 Rank-Allocation Sweep: Absolute Values

Table 10 gives the absolute training rewards behind Figure 3.

Table 10: Rank-allocation sweep on Autoware at a fixed total rank of 12. Entries are training reward at epoch 10, averaged over two seeds for the two extreme arms and one seed for SE-LoRA. Parameter counts are not matched and cannot be: with four per-domain adapters the trainable count grows as $r_s + 4r_e$, so the expert-only arm trains a third more parameters than SE-LoRA and the shared-only arm a third as many. Training scenes per domain are given in the header.

Allocation	Trainable params	Map A (2,663)	Map B (1,865)	Map C (172)	Map D (1,496)
Expert only (r_s=0, r_e=12)	940,416	190.3	554.7	**−4.4**	456.2
SE-LoRA (r_s=4, r_e=8)	705,312	**197.1**	563.2	240.0	465.6
Shared only (r_s=12, r_e=0)	235,104	**167.3**	560.9	249.0	463.6

C.7 Leave One Domain Out Transfer

We hold one domain out of fine-tuning entirely and route its scenes through the shared adapter alone, which asks how much of the benefit survives on a domain the adapters never saw. All four runs use the same recipe as the main experiments (five epochs, seed 0) and are scored with the unweighted base reward on 50 held-out scenes per domain. All Map C scenes are the regenerated ones of §C.8.

The natural-looking summary, held-out gain divided by the average gain of the three domains that were trained on, is not comparable across rows: the denominator changes composition each time, and when Map C is held out the denominator loses Map C's own low gain and is inflated. We therefore compare each domain against itself. The numerator is that domain's gain in its own hold-out run; the denominator is the same domain's gain in the three runs where it was trained on and had its own adapter.

Map C, the test track, is the best transfer case rather than the worst, and the ordering follows training-set size. The rank sweep explains why. Map C's own adapter contributes very little in the first place, so removing it costs very little. Removing every per-domain adapter leaves Map C unchanged while costing Map A 15 percent (Figure 3), and the Map C adapter's weight norm is 0.16 ± 0.01 against 0.42 to 0.67 for the other domains (§C.5). The transfer gap therefore follows the amount of data per domain rather than the road geometry of the held-out site.

One caveat on the measure itself: the held-out run never saw that domain's data, so "retains" combines not having an adapter with not having seen the data. That is the right quantity for a transfer statement but it does not isolate the adapter. Single seed.

C.8 A Data Fault in the Map C Scenes and What It Invalidated

Every Map C scene we originally processed carried empty lane, route, boundary and polygon tensors. The converter in the public Autoware planner repository instantiates its projector with a fixed origin at latitude and longitude zero, while the Map C site is published in a local Cartesian frame with its own origin. Every lanelet therefore projected far outside the ego-centred crop and was discarded, so the planner received no map at all on that domain, in training and in deployment alike. We verified this on all 172 scenes of the affected split and on 20 random scenes from each of the other three domains, which were unaffected.

We regenerated the affected scenes with the projector taken from the map's own metadata, which required two further fixes to handle lanelets that carry no speed limit. The regenerated scenes are bit-identical to the originals in ego state and goal, and now carry 140 lanes and 11 to 13 route polylines per scene. The main-text results use the regenerated data, namely the rank-allocation sweep, the transfer study of §C.7, and the training curves of §C.1. Anyone converting Autoware maps published in a local Cartesian frame with the public converter will meet the same fault, so we will release the fix with our code.

C.9 Overlap Between the Fine-Tuning Pool and val14

The nuPlan scenes we fine-tuned on were processed from the nuPlan v1.1 validation split, which is the split the val14 evaluation set is drawn from. We report what we could and could not establish about the resulting overlap.

The processed pool we still hold contains 66,929 scenes, each identified by a scenario token. Of these, 24 tokens appear in the val14 scenario list, 21 from Las Vegas and 3 from Pittsburgh. The exact 20,000-scene subset used for the fine-tuning runs was not recorded and cannot be reconstructed, so we cannot state the overlap of that subset directly, and we cannot measure log-level overlap at all, since the processed scenes retain only a token and a map name rather than a source log. The behavior-cloning baseline is unaffected: it is the publicly released checkpoint, trained by its authors on the training split.

Two things follow. A future version of this comparison should fine-tune on the training split and evaluate on val14, with token and log filtering applied in advance. And the direction of the contamination matters for how to read our result: overlap would inflate the fine-tuned configurations relative to the baseline, and no fine-tuned configuration beat the baseline, so the main finding is if anything understated by this flaw rather than produced by it.

C.10 A Multi-Seed nuPlan Rerun Under the Unweighted Reward

We re-ran SE-LoRA fine-tuning on nuPlan with five seeds under the unweighted base reward. The resulting policies exploit that reward. Its progress term is a clamped raw path length, so an oscillating trajectory accumulates path length without going anywhere. After fine-tuning, the planned path is

about 218 m over eight seconds while the net displacement is 16 m, below the 29 m of the base model, and the raw lateral acceleration exceeds 800 m/s^2. Across all 1,800 foreign-scene deltas the raw speed and raw jerk deltas correlate at r=0.997, meaning both metrics measure the same thing in this regime, namely the amplitude of the oscillation. A double dissociation between two metrics cannot be tested when the two metrics have collapsed onto one axis.

This run has three consequences for the paper. First, it can neither confirm nor refute the nuPlan patching result, which is why §C.5 reports that result for a single checkpoint. Second, we report no shared-adapter ablation on these policies. Removing the shared adapter there takes raw speed from about 27 to about 8 m/s, and that shift reflects the exploited reward rather than what the shared adapter represents. Third, the run is a concrete case of the paper's point, since the reward was optimized well while the driving it should induce got worse. We keep the progress term unchanged in all other experiments, because changing it would change the objective every other result was trained against.

C.11 Docked-Expert Placement Control

To isolate the role of parameter placement, we retrain the shared adapter unchanged (in-backbone, rank 4, same 16 layers) while replacing each in-backbone expert adapter with an external head: a learned softmax mixture over all DiT block outputs, followed by layer normalization and a single-hidden-layer MLP whose output layer is zero-initialized, adding a correction to the final-layer output (approximately 150K parameters per expert). Routing, data, reward, and hyperparameters are identical to the in-backbone configuration; this control is a single-seed run.

The docked variant trains to a mean reward of 363.2 versus 383.4 for the in-backbone configuration, with an identical per-domain reward ordering. Table 11 reports the behavioral head-swap analysis on held-out scenes: each scene is rolled out under every head, and we measure the mean trajectory divergence against the native head. The dense-urban head is strongly distinct on its own scenes, while the suburban and campus heads are nearly interchangeable, mirroring the urban versus non-urban split observed in the gradient clusters. Under the coordinate-consistent protocol of Appendix F, docked-head gradients are substantially more cross-domain aligned (0.87 mean) than in-backbone expert gradients (0.43), indicating that capacity at the output interface receives largely common supervision, while domain-conflicting signals are absorbed inside the backbone.

Table 11: Docked-expert head-swap trajectory divergence (m) versus the native head (rows = active head, columns = scene domain; up to 30 held-out scenes per domain, single seed). Maps A/B/C/D = dense urban / suburban / OOD test track / campus.

Head	Scenes A	Scenes B	Scenes C	Scenes D
Head A	0	5.71	10.54	6.73
Head B	9.87	0	5.06	1.16
Head C	11.18	5.90	0	5.16
Head D	9.69	1.33	5.32	0

D Real Vehicle Deployment

This section documents the test vehicle, the sensor and planner wiring, and a paired offline comparison of planned trajectories on recorded drives. It shows that the fine-tuned planner runs in a production stack; it is not evidence for any part of the claim in §1, and we report it separately for that reason.

D.1 Vehicle Configuration and Pipeline Wiring

The physical deployment utilizes a drive-by-wire research vehicle equipped with a mechanical rotating lidar, a dual-antenna inertial navigation system, and a front-facing radar. Compute is handled by a consumer-grade graphical processing unit running the planning stack via hardware acceleration.

Deploying the diffusion planner required specific modifications to the base autonomy stack. First, we inserted the exported network and a downstream trajectory optimizer into the planning tree,

routing the output directly to the chassis controller. Second, we remapped the standard validation topics to prevent race conditions with the trajectory optimizer. Third, we replaced the standard deterministic safety validators with stochastic-friendly alternatives, as the original rule-based blocks rejected the natural physical variance of diffusion outputs. Finally, we disabled the default steer convergence locks on the longitudinal controller to allow continuous motion initialization. These modifications ensured that the physical chassis correctly executed the planned trajectories while maintaining system-level safety gates.

D.2 Hardware Export and Post-Processing Alignment

Physical deployment exposed a critical scaling misalignment invisible to pure simulation environments. The deployed models shared a common configuration file containing denormalization parameters. Because the specialized models were trained under a different absolute output scale than the baseline, the shared post-processor scaled the trajectories incorrectly upon initial physical deployment.

This hardware-specific failure demonstrates that bit-exact simulation equivalence is necessary but entirely insufficient for safe deployment. A separate forward equivalence check between the exported model and the deployment-side post-processor is strictly required. This empirical failure reinforces the broader theme of our work: researchers must audit the entire deployment pipeline mechanistically rather than relying exclusively on aggregate simulation scores.

D.3 Paired Offline Trajectory Evaluation

To validate physical plausibility prior to closed loop execution, we performed a paired offline evaluation using real sensor logs. Both the baseline and our specialized planner ran on the identical bagged sensor stream. Table 12 reports the trajectory intrinsic differences. The pairwise differences in peak velocity, jerk, and path length remain strictly within the simulation noise envelope. This confirms that the specialized models produce physically viable trajectories on real lidar input, successfully transferring the geometric and dynamic properties learned in simulation to physical hardware.

Table 12: Paired offline trajectory evaluation on real sensor input. Metrics compare the baseline model against the specialized architecture evaluated on the identical hardware sensor stream.

Trajectory Metric	**Baseline Model**	**Specialized Model**	**Relative Change**
Peak Velocity Mean (m/s)	4.49	4.42	-1.5%
Jerk Mean (m/s^3)	2.69	2.75	$+2.4\%$
Path Length Mean (m)	17.15	16.45	-4.1%
First Point Velocity Median (m/s)	2.96	3.05	$+3.0\%$

E Per-Scenario-Type Breakdown and Solver Equivalence Test

Figure 2 reports a related per-scene measurement. Scenes where DPM-Solver++ and DDIM trajectories diverge from the same initial noise are the scenes where the mismatched sampler scores badly ($r{=}-0.91$, $n{=}35$). The correlation with the matched sampler's own score is weaker ($r{=}-0.69$) and the correlation with the gap between the two is near zero, so divergence indicates which scenes are difficult rather than which sampler is better. We report it as a per-scene difficulty diagnostic and not as evidence that matching the sampler raises the benchmark, which §4.2 shows it does not.

Table 3 gives the pre-specified TOST equivalence test on val14 closed-loop score for DPM-Solver++ against the mismatched DDIM sampler. Figure 5 in the main text splits the score into the 14 scenario types. Relative to DDIM, DPM-Solver++ gains on some types and loses on others, for example $+0.013$ on *stopping with lead* and -0.016 on *starting right turn*, so the aggregate equivalence of the two samplers comes from these per-type differences cancelling, not from the two samplers behaving the same on every type.

F Gradient Subspace Geometry and the Coordinate-Alignment Artifact

Evaluating cross-ODD gradient cosine similarity through independently initialized expert parameters compares vectors expressed in different parameter coordinates. Because low-rank factorizations are identified only up to rotation (consistent with the 83.5° mean principal angle we measure between equally performing cross-seed solutions), such cross-parameter cosines are deflated toward zero regardless of the supervision geometry. An apparent $10\times$ alignment gap between shared and expert gradients measured under this protocol is therefore a measurement artifact, and we do not use cross-adapter gradient similarity as evidence anywhere in this paper.

The corrected protocol forces the same expert's parameters across all ODDs' data, so all compared gradients live in one coordinate system (10 scenes per ODD, gradients accumulated per ODD). Under this protocol, the mean cross-ODD alignment is 0.42–0.43 for expert parameters and 0.40–0.42 for shared parameters, with no separation at any layer position (final projection, output block, or interior layers). The corrected gradient geometry therefore does not separate the two components, and we position gradient-space analysis strictly as secondary; the causal behavioral interventions and sign-uniformity ablations in §4 carry the decomposition claim.