CW-ERM: Improving Autonomous Driving Planning with Closed-loop Weighted Empirical Risk Minimization

Eesha Kumar[1]*, **Yiming Zhang**[2], **Stefano Pini**[1], **Simon Stent**[1], **Ana Sofia Rufino Ferreira**[2], **Sergey Zagoruyko**[1], **Christian S. Perone**[1]*

[1]Woven Planet United Kingdom Limited
[2]Woven Planet North America, Inc.
`{firstname}.{lastname}@woven-planet.global`

Abstract

The imitation learning of self-driving vehicle policies through behavioral cloning is often carried out in an open-loop fashion, ignoring the effect of actions to future states. Training such policies purely with Empirical Risk Minimization (ERM) can be detrimental to real-world performance, as it biases policy networks towards matching only open-loop behavior, showing poor results when evaluated in closed-loop. In this work, we develop an efficient and simple-to-implement principle called Closed-loop Weighted Empirical Risk Minimization (CW-ERM), in which a closed-loop evaluation procedure is first used to identify training data samples that are important for practical driving performance and then these are upsampled to help debias the policy network. We evaluate CW-ERM in a challenging urban driving dataset and show that this procedure yields a significant reduction in collisions as well as other non-differentiable closed-loop metrics.

1 Introduction

Learning effective planning policies for self-driving vehicles (SDVs) from data such as human demonstrations remains one of the major challenges in robotics and machine learning. Since early works such as ALVINN Pomerleau (1989), Imitation Learning has seen major recent developments using modern Deep Neural Networks (DNNs) Bansal et al. (2019); Xu et al. (2017); Bojarski et al. (2016); Codevilla et al. (2018); Kuefler et al. (2017); Vitelli et al. (2022). Imitation Learning (IL), and especially Behavioral Cloning (BC), however, still face fundamental challenges Codevilla et al. (2019), including causal confusion de Haan et al. (2019) (later identified as a feedback-driven covariate shiftSpencer et al. (2021)) and dataset biases Codevilla et al. (2019), to name a few.

There is one particular limitation of IL policies trained with BC that is, however, often overlooked: the mismatch between training and inference-time execution of the policy actions. Most of the time, BC policies are trained in an open-loop fashion, predicting the next action given the immediate previous action and optionally conditioned on recent past actions Bansal et al. (2019); Xu et al. (2017); Bojarski et al. (2016); Codevilla et al. (2018); Vitelli et al. (2022). These policies, however, when executed in real-world, impact the future states. Small prediction errors can drive covariate shift and make the network predict in an out-of-distribution regime.

In this work, we address the mismatch between training and inference through the development of a simple training principle. Using a closed-loop simulator, we first identify and then reweight samples

[*]Equal contribution

Machine Learning for Autonomous Driving Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, USA.

that are important for the closed-loop performance of the planner. We call this approach **CW-ERM** (Closed-loop Weighted Empirical Risk Minimization), since we use Weighted ERM Shimodaira (2000) to correct the training distribution in favour of closed-loop performance. We extensively evaluate this principle on real-world urban driving data and show that it can achieve significant improvements on planner metrics that matter for real-world performance (e.g. collisions).

Our contributions are therefore the following:

- We motivate and propose Closed-loop Weighted Empirical Risk Minimization (CW-ERM), a technique that leverages closed-loop evaluation metrics acquired from policy rollouts in a simulator to debias the policy network and reduce the distributional differences between training (open-loop) and inference time (closed-loop);
- we evaluate CW-ERM experimentally on a challenging urban driving dataset in a closed-loop fashion to show that our method, although simple to implement, yields significant improvements in closed-loop performance without requiring complex and computationally expensive closed-loop training methods;
- we also show an important connection of our method to a family of methods that addresses covariate shift through density ratio estimation.

In Section 2, we detail the proposed CW-ERM and in Section 4 we show the CW-ERM experiments and compare them against ERM.

2 Methodology

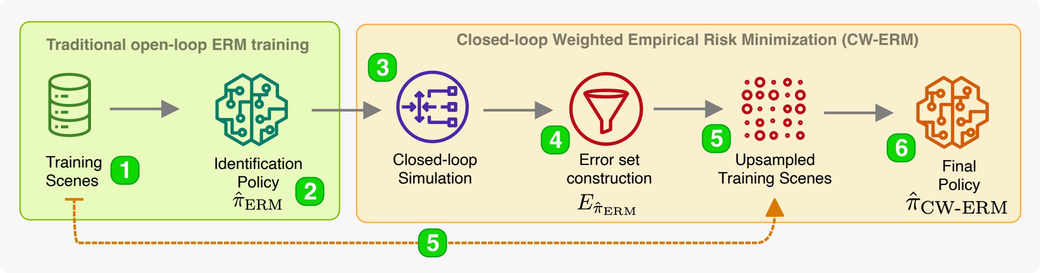
Traditional open-loop ERM training
Closed-loop Weighted Empirical Risk Minimization (CW-ERM)
Training Scenes
1
Identification Policy
$\hat{\pi}_{\text{ERM}}$
2
3
Closed-loop Simulation
4
Error set construction
$E_{\hat{\pi}_{\text{ERM}}}$
5
Upsampled Training Scenes
6
Final Policy
$\hat{\pi}_{\text{CW-ERM}}$
5

Figure 1: High-level overview of our proposed Closed-loop Weighted Empirical Risk Minimization (CW-ERM) method. In steps **(1-2)** we train an identification policy $\hat{\pi}_{\text{ERM}}$ using traditional ERM Vapnik (1991) on a set of training data samples or driving "scenes". In step **(3)**, we perform closed-loop simulation of the policy $\hat{\pi}_{\text{ERM}}$ and collect metrics to construct the error set in step **(4)**. With the error set in hand, we upsample scenes in the training set as shown in step **(5)**. We train the final policy $\hat{\pi}_{\text{CW-ERM}}$ using CW-ERM as shown in step **(6)** with the upsampled $\mathcal{D}_{\text{up}}$ set.

2.1 Problem Setup

The traditional formulation of supervised learning for imitation learning, also called behavioral cloning (BC), can be formulated as finding the policy $\hat{\pi}_{BC}$:

$$\hat{\pi}_{BC} = \operatorname*{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}, a \sim \pi^*(s)}[\ell(s, a, \pi)] \tag{1}$$

where the state s is sampled from the expert state distribution d_{π^*} induced when following the expert policy π^*. Actions a are sampled from the expert policy $\pi^*(s)$. The loss ℓ is also known as the surrogate loss that will find the policy $\hat{\pi}_{BC}$ that best mimics the unknown expert policy $\pi^*(s)$. In practice, we only observe a finite set of state-action pairs $(s_i, a_i^*)_{i=1}^m$, so the optimization is only approximate and we then follow the Empirical Risk Minimization (ERM) principle to find the policy π from the policy class Π.

If we let $\mathbb{E}_{s\sim d_{\pi^*},a\sim\pi^*(s)}[\ell(s,a,\pi)]=\epsilon$, then it follows that $J(\pi)\leq J(\pi^*)+T^2\epsilon$ as shown by the proof in Ross and Bagnell (2010), where J is the total cost and T is the task horizon. As we can see, the total cost can grow quadratically in T.

When the policy $\hat{\pi}_{BC}$ is deployed in the real-world, it will eventually make mistakes and then induce a state distribution $d_{\hat{\pi}_{BC}}$ different than the one it was trained on (d_{π^*}). During closed-loop evaluation of driving policies, non-imitative metrics such as collisions and comfort are also evaluated. However, they are often ignored in the surrogate loss or only implicitly learned by imitating the expert due to the difficulty of overcoming differentiability requirements, as smooth approximations of these metrics are still different than the non-differentiable counterparts often used. These policies can often show good results in open-loop training, but perform poorly in closed-loop evaluation or when deployed in a real SDV due to the differences between $d_{\hat{\pi}_{BC}}$ and d_{π^*}, where the estimator is no longer consistent.

2.2 Closed-loop Weighted Empirical Risk Minimization

In our method, called “Closed-loop Weighted Empirical Risk Minimization” (CW-ERM), we seek to debias a policy network from the open-loop performance towards closed-loop performance, making the model rely on features that are robust to closed-loop evaluation. Our method consists of three stages: the training of an identification policy, the use of that policy in closed-loop simulation to identify samples, and the training of a final policy network on a reweighted data distribution. More explicitly:

Stage 1 (identification policy): train a traditional BC policy network in open-loop using ERM, to yield $\hat{\pi}_{\text{ERM}}$.

Stage 2 (closed-loop simulation): perform rollouts of the $\hat{\pi}_{\text{ERM}}$ policy in a closed-loop simulator, collect closed-loop metrics and then identify the error set below:

$$E_{\hat{\pi}_{\text{ERM}}}=\{(s_i,a_i)\text{ s.t. }C(s_i,a_i)>0\},\tag{2}$$

where s$_i$ is a training data sample, or “scene” with a fixed number of timesteps from the training set, ai is the action performed during the rollout and $C(\cdot)$ is a cost such as the number of collisions found during closed-loop rollouts.

Stage 3 (final policy): train a new policy using weighted ERM where the scenes belonging to the error set $E_{\hat{\pi}_{\text{ERM}}}$ are upweighted by a factor $w(\cdot)$, yielding the policy $\hat{\pi}_{\text{CW-ERM}}$:

$$\underset{\pi\in\Pi}{\operatorname{argmin}}\,\mathbb{E}_{s\sim d_{\pi^*},a\sim\pi^*(s)}[w(E_{\hat{\pi}_{\text{ERM}}},s)\ell(s,a,\pi)]\tag{3}$$

As we can see, the CW-ERM policy in Equation 3 is very similar to the original BC policy trained with ERM in Equation 1, with the key difference of a weighting term based on the error set from closed-loop simulation in Stage 2. In practice, although statistically equivalent, we upsample scenes by a fixed factor rather than reweighting, as it is known to be more stable and robust An et al. (2021).

By training a policy using CW-ERM, we expect it to upsample scenes that perform poorly in closed-loop evaluation, making the policy network robust to the covariate shift seen during inference time while unrolling the policy.

We describe the complete CW-ERM training procedure in Algorithm 1 and in Figure 1 we show a high-level overview of our method.

2.3 Relationship to covariate shift adaptation with density ratio estimation

One important connection of our method is with covariate shift correction using density ratio estimation Shimodaira (2000). To correct for the covariate shift, the negative log-likelihood is often weighted by the density ratio $r(s)$:

$$\underset{\pi\in\Pi}{\operatorname{argmin}}\,\mathbb{E}_{s\sim d_{\pi^*},a\sim\pi^*(s)}[r(s)\ell(s,a,\pi)]\tag{4}$$

where $r(s)$ is defined as the density ratio between test and training distributions:

Algorithm 1 CW-ERM training procedure

```
Input: Training set $\mathcal{D}$ and hyperparameters $K$ (number of epochs) and $w$ (upsampling factor).
  Stage 1: Identification policy
  1. Train $\hat{\pi}_{\text{ERM}}$ on $\mathcal{D}$ using ERM for $K$ epochs (Equation 1).
  Stage 2: Closed-loop simulation
  2. Perform closed-loop simulation of the policy $\hat{\pi}_{\text{ERM}}$ in training scenes;
  3. Compute closed-loop evaluation metrics;
  4. Build the error set $E_{\hat{\pi}_{\text{ERM}}}$ of training scenes from closed-loop metrics (Equation 2).
  Stage 3: Final policy
  5. Construct upsampled dataset $\mathcal{D}_{\text{up}}$ by upsampling the error set $E_{\hat{\pi}_{\text{ERM}}}$ by $w$ times;
  6. Train final model $\hat{\pi}_{\text{CW-ERM}}$ on $\mathcal{D}_{\text{up}}$ via CW-ERM (Equation 3).
```

$$r(s) = \frac{p_{\text{test}}(s)}{p_{\text{train}}(s)} \tag{5}$$

In practice, $r(s)$ is difficult to compute and is thus estimated. The density ratio will be higher when the sample is more important for the test distribution. In our method (CW-ERM), instead of using the density ratio to weight training samples, we resample the training set based on an estimate of each data point's importance towards good closed-loop behaviours. Like the density ratio, the weighting in our case will also be higher for when the sample is important for the test distribution.

One key characteristic of the importance weighted estimator is that it can be consistent even under covariate shift. We leave, however, the analysis of theoretical properties of our approximation for future work.

3 Related Work

Closely related to our work is the "Learning from Failure" method Nam et al. (2020) (also known as LfF), where the authors train two models at the same time with a similar purpose of mitigating bias. The difference is that in CW-ERM we train models sequentially and we use closed-loop evaluation metrics instead of the loss to upsample, which permits the use of non-differentiable metrics. We also do not use GCE (generalized cross-entropy) to bias the identification model. Our method is simple from a training perspective, but unlike LfF, it does assume the availability of a simulator.

A similar approach that has been successfully applied to computer vision and natural language processing is the *Just Train Twice* (JTT) algorithm Liu et al. (2021). JTT similarly first trains an identification model, then trains another model which upweights samples misclassified by the identification model. Although similar in the sense that two models are trained sequentially, in JTT the goal is to deal with worst-group accuracy and not to improve robustness to closed-loop behaviors of a planning model, as in our case.

Several works have attempted to address the covariate shift problem in imitation learning. ChauffeurNet Bansal et al. (2019) and SafetyNet Vitelli et al. (2022) add state perturbation to the training data for improved generalization. Similarly, DAVE-2 Bojarski et al. (2016) used video captured from three different cameras as well as perturbation on the captured images. Another common approach is to supplement training with on-policy data Ross et al. (2011); Pan et al. (2020); Prakash et al. (2020), however, in practice, collecting on-policy data for use during training can be extremely expensive and time-consuming.

Similar to our work, Urban Driver Scheel et al. (2022) also utilizes a closed-loop simulator, but the simulator is used directly during training to generate unrolls while using BPTT (backpropagation through time). Urban Driver needs a differentiable simulator and does not scale well due to the need for rollouts during training and the memory requirements of BPTT. In contrast, our work takes a much simpler approach where the closed-loop simulator is only used to identify which samples to up-weight and does not require a differentiable simulator, while being able to directly identify scenes that are important for closed-loop evaluation without having to change the training loss to add differentiable collision losses as in Urban Driver Scheel et al. (2022). In our work, any closed-loop metric can be used, with no requirements for differentiability.

4 Experimental Evaluation

4.1 Policy network architecture

Our method is agnostic to model architecture choices. To evaluate our CW-ERM approach, we adopt the recent network architecture of Vitelli et al. (2022) to represent a strong baseline performance for SDV planning. This model uses a transformer-based Vaswani et al. (2017) architecture with a vectorial input representation Gao et al. (2020) to create features for each element into vector sets. It consists of a PointNet-based Qi et al. (2017) module for local processing of vectorized inputs and a global graph using a Transformer encoder for reasoning about interactions with agents and map features. Differently from Vitelli et al. (2022), we don't use a safety layer, as we want to evaluate the planner performance without external trajectory fallbacks. For further details, please refer to Vitelli et al. (2022).

4.2 Training

During training, we found that stopping training of the identification policy before convergence, similar to what was done in JTT Liu et al. (2021) and LfF Nam et al. (2020), also yielded better results. We limited the capacity of the identification policy by training it until K epochs. The insight is that important biases are learned in early training phases Nam et al. (2020) and limiting model capacity can avoid overfitting and avoid depletion Liu et al. (2021) of the error set used for the training of the final policy. We train the final policy for 40 epochs.

For the ERM baseline, we compare our method against two different experiments where we have the traditional BC trained with ERM with and without perturbations (details can be found in Appendix A).

4.3 Datasets

We train and test CW-ERM on a proprietary dataset. Our SDV data is collected in challenging urban missions on San Francisco and Palo Alto roads. This dataset is a collection of driving trajectories from our SDV and surrounding agents, along with recorded HD Maps. Various types of behavioral scenarios in urban driving such as stopping behind a lead vehicle, stopping at intersections, and driving among dense cars, pedestrians, cyclists etc. are captured. The majority of scenes in our dataset are between 11-13 seconds long, with the longest lasting up to 30 seconds. The total data used during training is 180 hours and we validate and test on 60 hours of driving data each.

4.4 Evaluation framework

We compute the closed-loop evaluation metrics by doing rollouts of the policy in the log-replayed scenes on a simulator [2] During the unroll, trajectories are recorded. An evaluation plan composed of a set of metrics and constraints is executed over the recorded trajectories. We count every scene that violated a constraint (e.g., a collision) and then compute the confidence intervals (CIs) for each metric using a Binomial exact posterior estimation with a flat prior, which gives similar results (up to rounding errors) to bootstrapping as recommended in Agarwal et al. (2021).

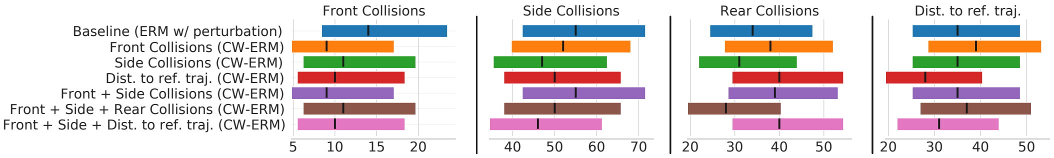

Figure 2: Visual representation of the experimental results from closed-loop evaluation in simulation shown in Table 1. Confidence intervals (CIs) were calculated using .95 interval from an exact Binomial posterior with a flat prior. In this plot we only compare against the best baseline (ERM with perturbation).

[2] We will open-source the simulator and metrics used in this work after the review period.

Table 1: Experimental results from closed-loop evaluation in simulation. In this table we show a baseline method of behavioral cloning (ERM) with and without perturbations (details can be found in Appendix A) together with the results from single and multi-metric experiments. p95 confidence interval bounds are provided in brackets. Lower is better for all metrics.

Method	**Upsampled metric**	**Perturbation**	**Front Collisions**	**Side Collisions**	**Rear Collisions**	**Dist. to ref. traj.**
ERM (baseline)	(not applicable)		67 (52.8, 85.0)	97 (79.6, 118.2)	114 (94.9, 136.8)	75 (59.9, 93.9)
ERM (baseline)	(not applicable)	✓	14 (8.4, 23.5)	55 (42.3, 71.6)	34 (24.4, 47.5)	35 (25.2, 48.6)
CW-ERM (ours)	Front Collisions	✓	**9** (4.8, 17.1)	52 (39.7, 68.1)	38 (27.7, 52.1)	39 (28.6, 53.2)
CW-ERM (ours)	Side Collisions	✓	11 (6.2, 19.7)	47 (35.4, 62.5)	31 (21.9, 44.0)	35 (25.2, 48.6)
CW-ERM (ours)	Dist. to Reference Trajectory	✓	10 (5.5, 18.4)	50 (37.9, 65.8)	40 (29.4, 54.4)	**28** (19.4, 40.4)
CW-ERM (ours)	Front + Side Collisions	✓	9 (4.8, 17.1)	55 (42.3, 71.6)	39 (28.5, 53.2)	35 (25.2, 48.6)
CW-ERM (ours)	Front + Side + Rear Collisions	✓	11 (6.2, 19.7)	50 (37.9, 65.8)	**28** (19.4, 40.4)	37 (26.9, 51.0)
CW-ERM (ours)	Front + Side + Dist. to ref. traj.	✓	10 (5.5, 18.4)	**46** (34.5, 61.3)	40 (29.4, 54.4)	31 (21.9, 44.0)

4.5 Metrics

Metrics computed in the closed-loop simulator are used to construct the error set (Equation 2). In our evaluation we consider certain important metrics: the number of front collisions, side collisions, rear collisions, and distance from reference trajectory. The distance from reference trajectory considers the entire target trajectory for the current simulated point. A failed scene with respect to this metric is one where the distance of the simulated center of the SDV to the closest point in the target trajectory is farther than four meters.

In our evaluation, we perform two sets of experiments: *single metric* and *multi metric*. In single metric experiments we construct the error set using only a single metric, while for multi metric we use scenes from multiple metrics together.

4.6 Results

4.6.1 Single Metric

We show the results from single metric experiments in Table 1. We can see that the number of collisions significantly reduced for both side and front collision experiments. We found improvements in the range of $\sim$35% on the test set for some metrics when compared to the baseline.

We also found that the largest margin of improvements targeting single metrics in isolation were seen when using single metric based error set, while a balance was achieved when targeting multiple metrics, which suggests a Pareto front of solutions when targeting multiple objectives.

Variance is also lower in some cases when compared to the baseline. We note that while upsampling a certain metric, it shows noticeable improvements in other related metrics. For example, in our single metric experiments, we see that improving side collisions also improve rear collisions. This is evidence that the model is not only getting better at side collisions but also becoming less passive (as indicated by reduction in rear collisions, due to log-replayed agents in simulation that are non-reactive).

Qualitative results during closed-loop unroll are shown in Figures 4 and 5. Here, we show improved behavior when the scene is upsampled in CW-ERM - once for front collisions Figure 4 and another for side collisions Figure 5. Here, we see a better response in the CW-ERM model which avoids collisions by waiting at intersection and slowing down next to a lead vehicle.

4.6.2 Multi Metric

In our multi-metric experiments, we combine two or more metrics - namely $m_1, m_2..m_N$ - into a single upsampling experiment. The metrics are equally weighted and hence scenes that fail due to any m_i will be added to the error set. While improvements are noticeable upon combining Front and Side collisions or Front, Side and Distance to the reference trajectory in Table 1, considerable regression is observed when adding rear collisions. As we can see from the experiments, this is clearly related to the amount of false-positives (FPs) in rear collisions due to the lack of agent reactivity during log playback in the simulator.

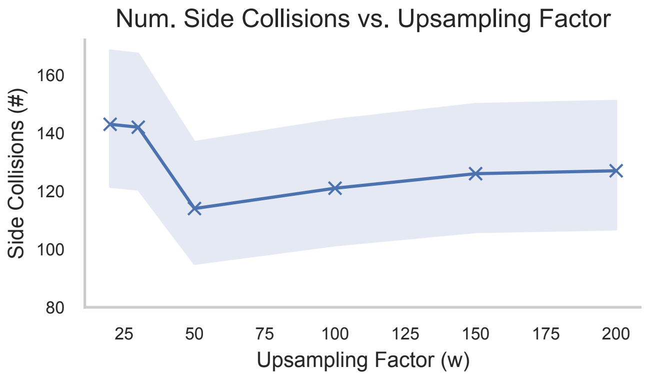

Figure 3: A sample plot showing the effect of upsampling factor on the performance based on single cost upsampling. Here, the scenes are upsampled based on side collisions by factor w along X-axis. The resulting side collisions obtained while evaluating on the validation dataset is obtained on the Y-axis. It is evident from the plot that the performance improves until an upsampling factor of $w = 50$ after which number of side collisions begin to increase.

4.7 Hyper-parameter Tuning / Upsampling Experiments

We evaluate the performance of the upsampled training set using various identification models $K \in \{10, 20\}$ with various upsampling factors $w \in \{10, 20, 30, 50\}$ on the validation dataset. Single metric upsampling experiments responded better to error set extracted from the training where $K = 10$, while using $K = 20$ performed better for multi-metric experiments. We find that the size of the resulting upsampled error set influences performance. As seen from Table 1 and Figure 3, there exists a limit beyond which performance does not improve during the upsampling experiments. A similar observation was noted in JTT Liu et al. (2021), where they also found an upsampling factor for which beyond it, worst-group accuracy could not be improved.

5 Limitations

Although our method is efficient, easy to implement and showed significant improvements, it also comes with limitations that are important to highlight. In this work, we use a proprietary dataset for evaluation, primarily due to the current lack of available closed-loop evaluation benchmarks. Most public datasets available today are focused on agent prediction tasks and on open-loop metrics (e.g. Chang et al. (2019); Ettinger et al. (2021)). Recently, the closed-loop planning benchmark nuPlan H. Caesar (2021) was released, but is still under active development and requires a special license for industrial labs to use.

Our method also introduces two new hyper-parameters: K (number of epochs for the identification model) and w (upsampling factor), however, we found the improvement to be robust to different parameterizations of these parameters, similar to past observations Liu et al. (2021).

We performed log-replay of agents in simulation, which can produce false positives for the rear collision metric. We leave further analysis and the usage of a reactive simulator as future direction.

We have not yet deployed our policy in a real-world SDV, however, we evaluated it on a closed-loop evaluation framework that is known to be closely representative of real-world performance. Deployment and testing of a policy in a real-world SDV on public roads requires further safety evaluations that we leave as future work.

6 Discussion

Most recent improvements in imitation learning are based on improving the asymptotic performance of algorithms. In this work we showed a different direction that tackles the problem by directly addressing the mismatch between training and inference without requiring an extra human oracle or adding extra complexity during training. Our method is as simple as upsampling scenes by leveraging any existing simulator and training two models, yet it showed that there is still room

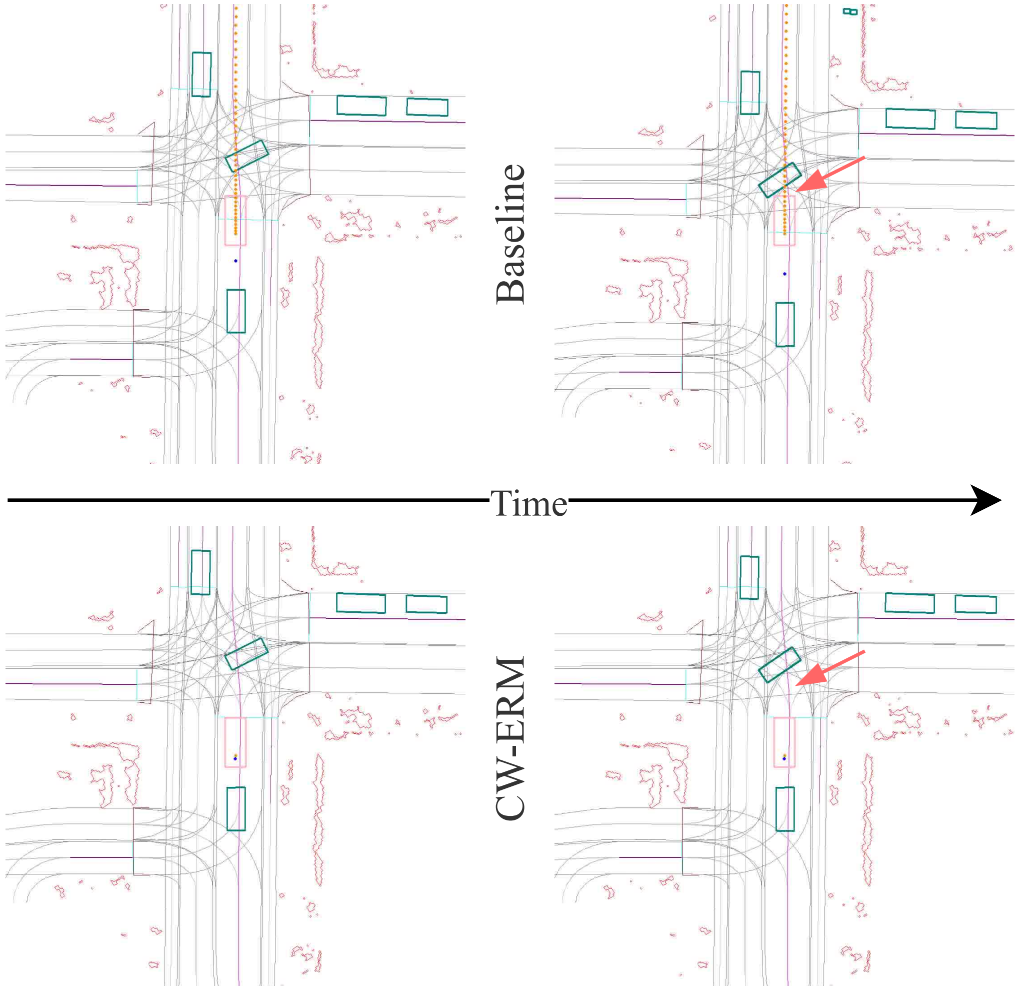

Figure 4: A scene from the test dataset showing the behavior of ego in Baseline (ERM) (with perturbation) and CW-ERM front collisions upsampled. The blue dots are the target trajectory and the yellow dots are the predicted trajectory. The ego is the box in pink and blue-green boxes are other agents. Here, we see that the baseline moves ahead at intersection ignoring the car from the right resulting in a front collision. In contrast, for the CW-ERM policy, it waits at the intersection.

for significant improvements without having to deal with human-in-the-loop, training rollouts or impacting the policy inference latency. We also described an important potential connection of our method with density ratio estimation for covariate shift correction Shimodaira (2000), which we believe is an exciting future research direction that could provide better theoretical understanding of the improvements seen in our experiments.

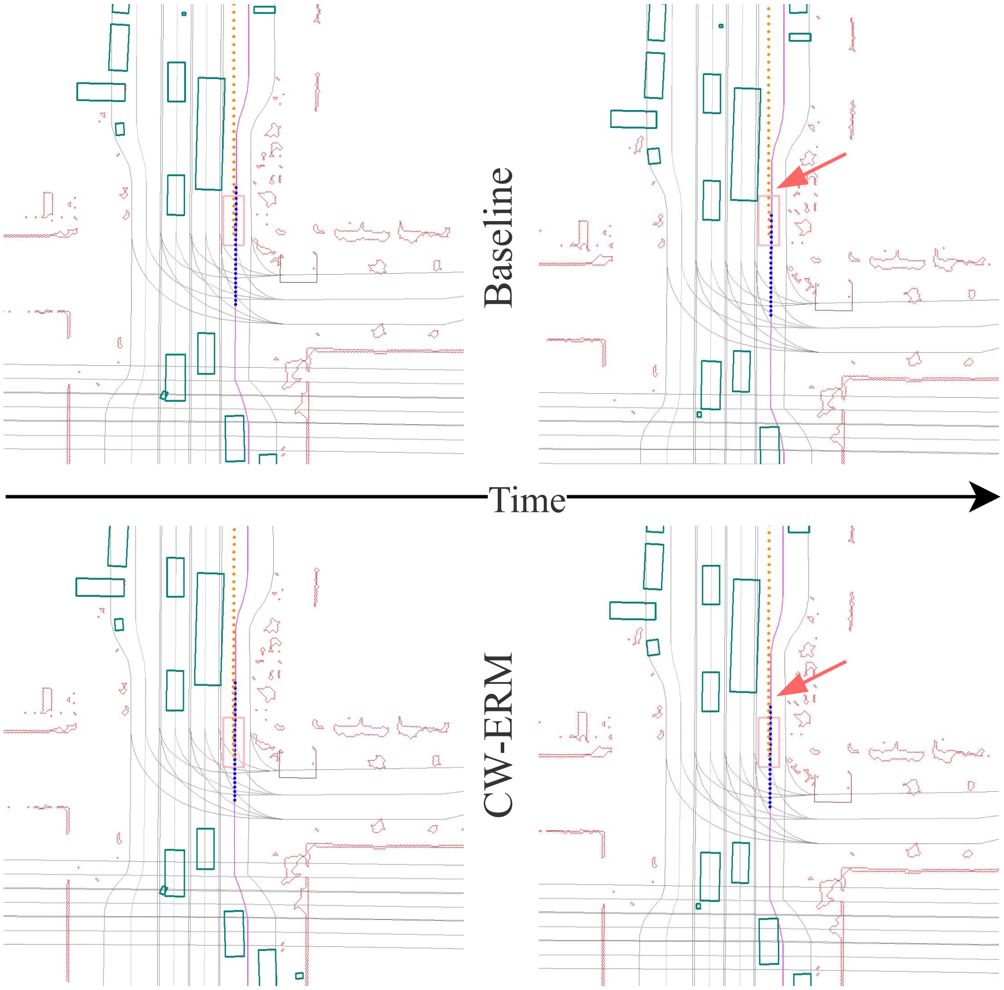

Figure 5: A scene from the test dataset showing the behavior of ego in Baseline (ERM) (with perturbation) and CW-ERM side collisions upsampled. The blue dots are the target trajectory and the yellow dots are the predicted trajectory. The ego is the box in pink and blue-green boxes are other agents. Here, we see that the baseline deviates from target trajectory and collides with the bus ahead. The CW-ERM policy slows down behind the bus and prevents a collision.

References

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

J. An, L. Ying, and Y. Zhu. Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proceedings of Robotics: Science and Systems*, June 2019. doi: 10.15607/RSS.2019.XV.031.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars, 2016.

M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, 2018. doi: 10.1109/ICRA.2018.8460487.

F. Codevilla, E. Santana, A. M. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019.

P. de Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021.

J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

K. T. e. a. H. Caesar, J. Kabzan. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *CVPR ADP3 workshop*, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211, 2017. doi: 10.1109/IVS.2017.7995721.

E. Z. Liu, B. Haghgoo, A. S. Chen, A. Raghunathan, P. W. Koh, S. Sagawa, P. Liang, and C. Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pages 6781–6792. PMLR, 2021.

J. Nam, H. Cha, S. Ahn, J. Lee, and J. Shin. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33:20673–20684, 2020.

Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots. Imitation learning for agile autonomous driving. *The International Journal of Robotics Research*, 39(2-3):286–302, 2020. doi: 10.1177/0278364919880273.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, 1989.

A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11763–11773, 2020.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

S. Ross and D. Bagnell. Efficient reductions for imitation learning. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, pages 718–728. PMLR, 2022.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. ISSN 0378-3758. doi: https://doi.org/10.1016/S0378-3758(00)00115-4.

J. C. Spencer, S. Choudhury, A. Venkatraman, B. D. Ziebart, and J. A. Bagnell. Feedback in imitation learning: The three regimes of covariate shift. *CoRR*, abs/2102.02872, 2021.

V. Vapnik. Principles of risk minimization for learning theory. In *Advances in Neural Information Processing Systems*, 1991.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

M. Vitelli, Y. Chang, Y. Ye, A. Ferreira, M. Wołczyk, B. Osiński, M. Niendorf, H. Grimmett, Q. Huang, A. Jain, et al. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 897–904. IEEE, 2022.

H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3530–3538, 2017. doi: 10.1109/CVPR.2017.376.

Appendix for CW-ERM: Improving Autonomous Driving Planning with Closed-loop Weighted Empirical Risk Minimization

A Perturbation

In our experiments, we employed similar perturbation techniques as in Vitelli et al. (2022). We randomly perturb the ego's current state to shift from the current trajectory in some of the training examples. To put it more concretely, for the perturbed states, we add a zero-mean Gaussian noise to the ego's current position and heading. We perturb the ego's speed by $av + |b|$ where v is the current speed, a and b are speed multiplier and bias terms generated by a zero-mean Gaussian distribution. Taking the absolute value of the bias term is to ensure that the perturbed speed is always non-negative. Additionally, we perform collision checks for every perturbed state and we do not include any perturbed states which includes collisions.

A.1 Policy network hyper-parameters

To train the baselines and our policy we employ a distributed training with a local batch size of 64 for each replica (with an effective batch size of 4096 when using 64 GPU replicas). We use a learning rate of 0.001 that is annealed with cosine schedule during training for 40 epochs (except for the identification models as described in the Section 4). We also used MAE (mean absolute error) loss and the Adam Kingma and Ba (2015) optimizer with default PyTorch Paszke et al. (2019) parameters for β_1 and β_2.

B Scene distribution

We select training, validation and test data subsets to be balanced. The scenarios selected for the datasets are diverse, and a variety of complex urban scenarios have been curated such that it is possible to evaluate closed-loop performance, even on minority scenario groups. In Figure 6, a detailed split of the various scenarios is provided.

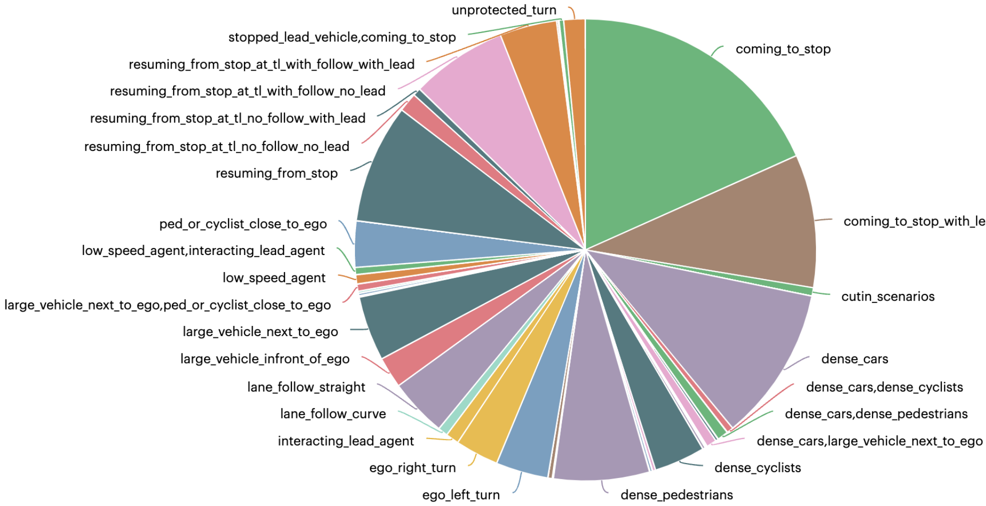

Figure 6: Scenario distribution of the training dataset. Scenes can be of single or multiple scenarios - in the case of multiple scenario tags, the scenario names are comma separated.