Self-Supervised Data Quality Scoring for Offline RL in Driving

Roman Gasiorowski
CS224R

One-Page Extended Abstract

Problem. Offline reinforcement learning is attractive for driving-related tasks because it can train policies from stored data rather than requiring risky online exploration. The limitation is that offline datasets are not always clean. A model may train on examples that are noisy, unreliable, or inconsistent with the behavior the final policy should learn. In this project, a low-quality or corrupted data point was defined as a driving transition whose numeric state and/or action values were deliberately altered so that the sample no longer faithfully represented the original offline driving data.

Approach. I used a CARLA-based offline driving dataset, but all experiments were performed directly on the stored data; I did not render or run new CARLA simulator rollouts. To create controlled corruption levels, I altered 10%, 30%, and 50% of the dataset using context-aware perturbations rather than a single simple random-noise pattern. The corruption procedure selected target transitions, identified related state/action features, and applied feature perturbations, plausible action swaps, and transition mismatches. This made corrupted samples look like altered driving data rather than obvious artifacts produced by one memorized noise pattern.

I then trained an ensemble of independently initialized neural networks to score whether each data point appeared consistent with the rest of the dataset. Each network learned its own scoring function from the same offline data. A data point was flagged as corrupted when a majority of the networks marked it as suspicious. Majority voting made the filtering step more stable than relying on a single model. The resulting filtered dataset was then used as a cleaner input for downstream offline RL training.

Main findings. The ensemble-based data quality score detected corrupted examples most reliably when most of the dataset was still clean. With a 10-network ensemble, detection AUC was 0.86 at 10% corruption, 0.76 at 30% corruption, and 0.61 at 50% corruption. Detection accuracy followed the same trend, dropping from 91% to 82% to 68% as corruption increased. I also found that larger ensembles improved detection quality. At 30% corruption, a 3-network ensemble achieved 0.68 AUC and 75% accuracy, a 5-network ensemble achieved 0.76 AUC and 82% accuracy, and a 10-network ensemble achieved 0.82 AUC and 87% accuracy.

Accomplishments and contribution. The project showed that disagreement among multiple neural-network scoring models could serve as a self-supervised quality check for offline RL data. The method did not require human labels during training, a separate expert reference dataset, or simulator rollouts. The main contribution was to use ensemble disagreement and majority voting as a data-quality filter before downstream offline RL training. The results supported the hypothesis that corrupted examples can be identified from the structure of the dataset itself, but also showed a clear limitation: the signal weakened when corrupted data became too common.

Limitations and future directions. The current corruption process was controlled and synthetic, even though it was designed to be harder than simple random noise. Future work should test more realistic forms of low-quality driving data, such as inconsistent actions, unstable transitions, or distribution shifts in the offline dataset. A second direction is to replace hard removal with a confidence score, where highly suspicious examples are removed while moderately suspicious examples are downweighted rather than fully excluded.

1 Introduction

Offline reinforcement learning (offline RL) trains policies from a fixed dataset rather than collecting new experience through interaction. This setting is appealing for autonomous driving because it avoids unsafe trial-and-error exploration during training. A policy can, in principle, learn from pre-collected driving logs without needing to test risky behavior on the road.

The problem is that offline data is not always clean. Real or simulated driving datasets can contain examples that are noisy, unreliable, or inconsistent with the behavior the final policy should learn. If a policy treats every transition as equally trustworthy, corrupted examples can influence the learned value function and policy. This project studied whether a model could identify corrupted offline driving data using only the dataset itself.

The central question was:

> *Can disagreement among independently trained neural networks identify corrupted data points in an offline driving dataset, and how does that signal degrade as the amount of corruption increases?*

I approached this as a self-supervised data quality problem. Instead of using human labels during training, I trained multiple neural-network scoring models on the offline dataset. Each model judged whether a data point appeared consistent with the rest of the data. If a majority of the models flagged a point as suspicious, I treated that point as likely corrupted and removed it before downstream offline RL training.

This report describes the method, the controlled corruption process, the ensemble-based scoring approach, and the main experimental findings.

2 Problem Setup

I used a CARLA-based offline driving dataset. The dataset contained stored transitions of the form

$$(s, a, r, s', d),$$

where s was the current state, a was the action, r was the reward, s' was the next state, and d indicated whether the episode was done. Importantly, the experiments were performed directly on the stored dataset. I did not render new CARLA scenes or run simulator rollouts.

The task was to identify corrupted transitions in the offline dataset. A corrupted data point was defined as a transition whose numeric state and/or action values had been deliberately altered. These samples were low-quality because they no longer accurately reflected the clean driving behavior represented by the original dataset.

The corruption labels were known only because I created the corrupted datasets synthetically. These labels were not used to train the scoring networks. They were used only after training to measure whether the scoring method correctly identified altered examples.

3 Related Work

This project built most directly on three areas: offline RL, imperfect demonstration learning, and uncertainty-based offline RL.

Implicit Q-Learning. Implicit Q-Learning (IQL) is an offline RL method that avoids querying out-of-distribution actions by fitting value functions in a conservative way [Kostrikov et al., 2021]. It is well suited to fixed datasets, but it does not directly solve the problem of deciding which data points in the dataset are reliable. Like many offline RL methods, it largely assumes the dataset is usable as given.

D4RL and offline benchmarks. D4RL provides standardized datasets for deep data-driven reinforcement learning, including offline datasets that make it easier to compare algorithms [Fu et al., 2020]. Benchmarks are useful, but they often treat dataset quality as fixed. This project instead focused on the dataset-quality problem itself: identifying examples that should not be trusted as much as the rest of the data.

Imperfect demonstrations. Prior work on imperfect demonstrations has studied how to learn from mixed-quality data. DemoDICE, for example, uses distribution matching ideas to learn from supplementary imperfect demonstrations [Kim et al., 2022]. Other approaches learn behavior priors or use external signals to identify lower-quality trajectories. These methods are useful, but they often rely on expert data, human labels, or another reference signal. This project tested a more self-contained setting in which the method had only the offline dataset.

Uncertainty and ensemble disagreement. Uncertainty-based offline RL methods use disagreement among value functions to avoid overconfident learning in uncertain regions. Ensemble-Diversified Actor-Critic uses diversified Q-ensembles as part of conservative offline RL [An et al., 2021]. This project borrowed the idea that disagreement can be informative, but used it differently. Rather than using disagreement only as a training-time penalty, I used it as a pre-training data-quality signal.

4 Method

4.1 Controlled Context-Aware Corruption

A key design choice was how to corrupt the data. A simple option would have been to add the same kind of random noise to selected values. However, that would risk creating an easy shortcut: the scoring models might learn the noise pattern itself instead of learning that the sample was inconsistent with the dataset.

To avoid this, I used a more context-aware corruption procedure. The goal was to produce altered examples that looked plausible at the feature level but violated the relationships that clean driving data should satisfy. The corruption process had four steps:

1. **Select corruption targets.** I randomly selected a controlled percentage of transitions to alter: 10%, 30%, or 50%.
2. **Identify related features.** I grouped features that should have meaningful relationships, such as state features and action dimensions that jointly describe a driving decision.
3. **Apply heterogeneous perturbations.** I used several corruption modes rather than one fixed noise rule. These included feature perturbations, plausible-but-mismatched action swaps, and transition mismatches.
4. **Preserve valid ranges.** Corrupted values were clipped or constrained so that the data still looked numerically valid, even though the transition was no longer consistent with the original clean example.

This design made the detection task harder and more meaningful. The model could not simply memorize a single Gaussian noise pattern. It instead had to identify samples that were inconsistent with the

relationships learned from the broader dataset.

4.2 Neural-Network Scoring Ensemble

I trained an ensemble of K independently initialized neural networks. Each network received offline transitions and learned to score whether a data point appeared consistent or suspicious. The networks were trained independently so that they did not all learn the exact same scoring function.

The core intuition was that clean transitions should be easier for the networks to score consistently, while corrupted transitions should produce more disagreement or more suspicious scores. For each transition $x_i = (s_i, a_i, r_i, s'_i, d_i)$, each scoring model produced a suspiciousness score:

$$q_k(x_i), \quad k = 1, \ldots, K.$$

The ensemble produced two useful signals. The first was the average suspiciousness score:

$$\bar{q}(x_i) = \frac{1}{K} \sum_{k=1}^{K} q_k(x_i).$$

The second was disagreement:

$$u(x_i) = \mathrm{Var}_{k=1}^{K}[q_k(x_i)].$$

The current filtering method used majority voting. Each network voted whether a point was suspicious. A data point was removed if more than half of the ensemble voted that it was likely corrupted.

4.3 Countering Contaminated Data

After the ensemble identified suspicious data points, I removed the majority-voted corrupted examples from the dataset. The downstream offline RL model was then trained on the filtered dataset.

The goal was not to prove that hard filtering is always optimal. Rather, hard filtering provided a clear first test of whether the scoring networks were identifying useful data-quality information. A natural next step is to replace hard removal with confidence-weighted training:

$$w(x_i) = 1 - c(x_i),$$

where $c(x_i)$ is a corruption-confidence score based on the fraction of networks that flagged the point and the strength of their disagreement. Highly suspicious points could be removed, while moderately suspicious points could be downweighted instead of fully excluded.

5 Experiments

5.1 Experimental Design

I created three corrupted versions of the offline dataset: 10%, 30%, and 50% corrupted data. For each corruption level, the scoring ensemble was trained without using corruption labels. Labels were held out and used only for evaluation.

I evaluated detection using two metrics:

- **Detection AUC:** how well the scoring method ranked corrupted examples as more suspicious than clean examples. AUC near 0.5 means random guessing, while higher values indicate better separation.
- **Detection accuracy:** the percentage of examples correctly classified as clean or corrupted after applying the majority-vote decision rule.

I also tested the effect of ensemble size using 3, 5, and 10 networks at the 30% corruption level.

5.2 Detection Performance Across Corruption Levels

Corrupted Data Level	Detection AUC	Detection Accuracy
10%	0.86	91%
30%	0.76	82%
50%	0.61	68%

Table 1: Detection performance by corruption level using a 10-network ensemble.

Table 1 shows that the method performed best when the dataset was mostly clean. At 10% corruption, the ensemble identified altered data points with strong performance: 0.86 AUC and 91% accuracy. At 30% corruption, performance declined but remained useful, with 0.76 AUC and 82% accuracy. At 50% corruption, detection became much harder, dropping to 0.61 AUC and 68% accuracy.

This supported the main hypothesis. Ensemble-based quality scoring worked when corrupted data was a minority, but the signal weakened as corruption became more common.

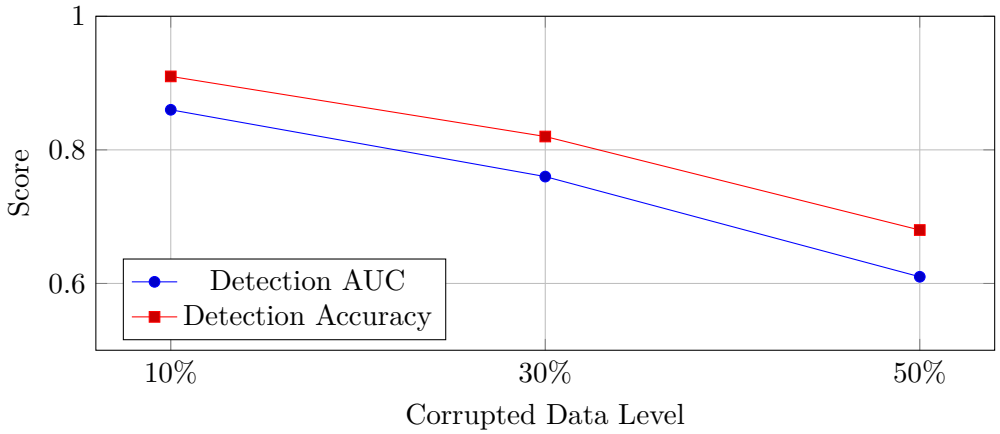

Figure 1: Detection AUC and accuracy decreased as the corruption level increased.

5.3 Effect of Ensemble Size

Table 2 shows that larger ensembles improved detection. With only 3 networks, the ensemble achieved 0.68 AUC and 75% accuracy. With 5 networks, it improved to 0.76 AUC and 82% accuracy. With 10 networks, it reached 0.82 AUC and 87% accuracy.

This suggested that independently trained networks provided complementary information. A larger ensemble gave a more reliable estimate of whether a point was truly suspicious.

Ensemble Size	Detection AUC	Detection Accuracy
3 networks	0.68	75%
5 networks	0.76	82%
10 networks	0.82	87%

Table 2: Effect of ensemble size at 30% corrupted data.

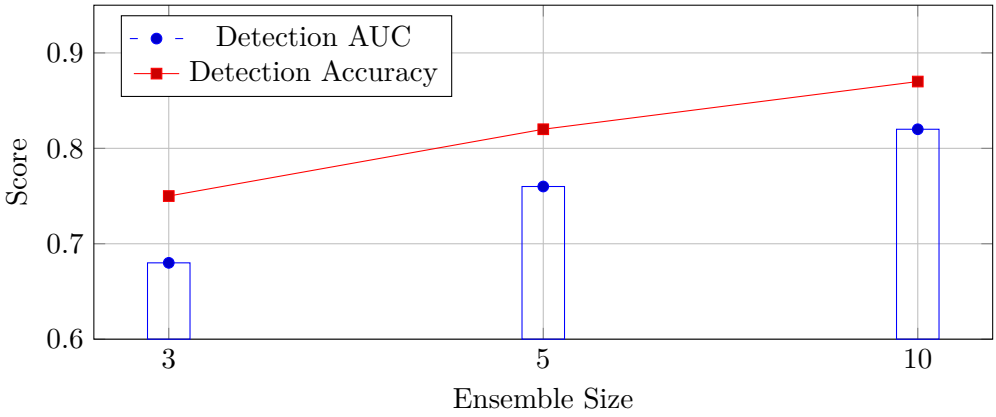

Figure 2: Larger ensembles improved both AUC and detection accuracy at 30% corruption.

6 Discussion

The results showed that self-supervised data quality scoring can identify corrupted examples in an offline driving dataset. The method worked especially well when the dataset was mostly clean. This was important because many practical offline RL settings may have mostly usable data with a smaller fraction of unreliable examples.

The degradation at 50% corruption was also informative. When corrupted examples became too common, the scoring networks had a harder time learning what clean consistency should look like. In other words, the ensemble could no longer rely on the clean data distribution as the dominant pattern. This supported the hypothesis that ensemble-based scoring depends on the majority of the dataset remaining reasonably reliable.

The ensemble-size result was another useful finding. Larger ensembles improved detection because they reduced the chance that any single network's mistake would dominate the decision. However, larger ensembles also increased compute cost. This created a clear tradeoff: better detection required more independently trained models.

7 Limitations

The main limitation was that the corruption process was synthetic. Although the corruption was designed to be context-aware and harder than a simple memorized noise pattern, it still did not fully capture all forms of real-world low-quality driving data. Real datasets may include distribution shifts, sensor artifacts, inconsistent human driving, delayed reactions, or rare but valid maneuvers.

A second limitation was that the current method used hard filtering. Removing examples by majority vote was simple and interpretable, but it may discard some useful data. A confidence-weighted method

could preserve more information by downweighting uncertain examples rather than removing them completely.

A third limitation was compute cost. Training multiple neural networks was more expensive than training a single model. The ensemble-size results showed that more networks improved detection, but this improvement came with additional training cost.

8 Conclusion

This project investigated self-supervised data quality scoring for offline RL in a driving setting. I used a CARLA-based offline dataset, created controlled context-aware corruption levels, trained ensembles of neural-network scoring models, and used majority voting to identify suspicious examples.

The main result was that corrupted data points could be detected using only the dataset itself. Detection was strong at 10% corruption, useful at 30% corruption, and much weaker at 50% corruption. Larger ensembles improved detection, suggesting that disagreement among independently trained networks was a useful signal for identifying low-quality data.

Overall, the project showed that offline RL should account for dataset quality, not just algorithm design. A self-supervised scoring ensemble can help identify and filter unreliable examples before downstream training, making offline RL more robust when datasets contain moderate amounts of corruption.

References

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *Advances in Neural Information Processing Systems*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Geon-Hyeong Kim, Sangwoo Seo, Jongmin Lee, Woojin Jeon, Honglak Hwang, Hongseok Yang, and Kee-Eung Kim. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.