CAUSAL SCENE BERT: IMPROVING OBJECT DETECTION BY SEARCHING FOR CHALLENGING GROUPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous vehicles (AV) rely on learning-based perception modules parametrized with neural networks for tasks like object detection. These modules frequently have low expected error overall but high error on atypical groups of data due to biases inherent in the training process. Multiple heuristics are employed to identify 'failures' in AVs, a typical example being driver interventions. After identification, a human team combs through the associated data to group perception failures that share common causes. More data from these groups is then collected and annotated before retraining the model to fix the issue. In other words, error groups are found and addressed in *hindsight* as they appear. Our main contribution is a pseudo-automatic method to discover such groups in *foresight* by performing causal interventions on simulated driving scenes. To keep our interventions on the data manifold, we use masked language models. We verify that the prioritized groups found via intervention are challenging for the object detector and show that retraining with data collected from these groups helps inordinately compared to adding more IID data. We also release software to run interventions in simulated scenes, which we hope will benefit the causality community.

1 INTRODUCTION

To deploy autonomous road vehicles, it is vital that they are robust and safe. An important aspect of safety is handling unusual scenarios. Current data-driven approaches trained to minimize expected error are sensitive to imbalanced data distributions. Even models with low expected error can still exhibit large errors on atypical groups of data that are nonetheless important for safe driving. The status quo approach to finding these groups in the AV stack operates in *hindsight* by analyzing real-world scenes requiring driver intervention or by feeding replayed or simulated scenes to a model and finding those that result in poor performance. Advanced techniques may use adversarial attacks to actively find failures (Xie et al., 2017; Athalye et al., 2017; Wang et al., 2021a). In all cases, the found data is fed back into the retraining process. While this improves the model, a notable problem remains — without knowing the underlying cause of a failure, it is impossible to ensure that the problem is adequately resolved. To identify the causal factors in the failures, human experts typically comb through the data and group commonalities, an expensive and time-consuming procedure.

We propose an alternative method to discover potential failures in *foresight*. Instead of finding failures from previously collected data, we perform interventions on existing data to find those interventions that are consistently detrimental to the performance of an AV stack. We focus on perception, and object detection specifically, in this work. We identify interventions that consistently cause performance drops as challenging groups. Concretely, consider a scene where a truck was not detected. Many explanations exist, ranging from the scene composition to the weather conditions to the way the light reflects off of a puddle and into the camera. The actual cause is unclear. If we however arrived at this scene counterfactually, by performing a single intervention on another scene, e.g. changing a car to the truck, we now have some clue that the underlying causal error is related to the truck itself. We can duplicate this intervention across many scenes and see if it consistently remains a problem. While the *exact* cause is still opaque, the proposed method provides automatic insight into what interventions cause consistent errors without collecting new data to analyze or manual scrubbing of failures.

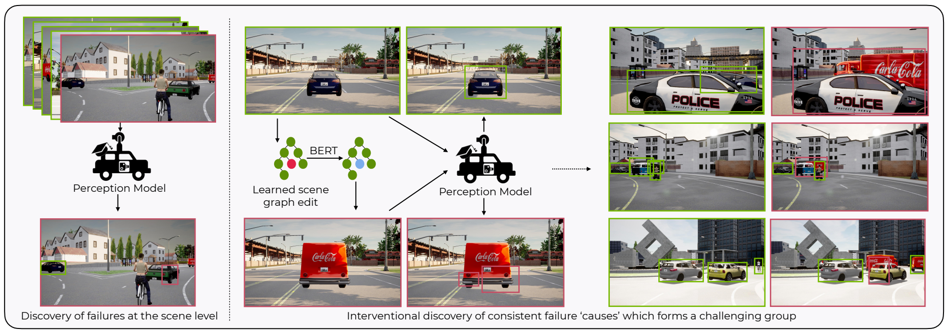
Perception Model
Discovery of failures at the scene level
BERT
Learned scene graph edit
Perception Model
Interventional discovery of consistent failure 'causes' which forms a challenging group

Figure 1: Instead of retrospectively discovering individual failure cases for AV perception, we actively search for causal interventions (edits) to existing scenes that consistently result in perception failures. Shown in the middle is an example of a single intervention causing perception failure, which we attribute to the intervention, as opposed to the left side where a combinatorial set of factors could explain the error. Consistent failures through this type of intervention constitute a challenging group for the perception model as seen on the right.

Performing such interventions requires the ability to manipulate scenes and re-render images. We demonstrate this in simulation, although recent advances (Ost et al., 2020) show promise in migrating our approach to real-world scenes. We assume access to a scene graph representation of the underlying scene on which we perform interventions. These interventions include changing agent properties like position, rotation, or asset type, as well as global weather conditions. While many interventions can potentially fail the detector, not all are useful. A scene with a flying truck could drop perception performance, but it is unlikely to occur in the real world. Ideally, interventions should be from the data distribution. We achieve this by training a density model of scenes (represented as flattened scene graphs) using a masked language model (MLM), a keystone in modern natural language processing pipelines. Taking interventions using the MLM amounts to masking a part of the scene graph and re-sampling from the predicted distribution.

Our work focuses on 2D object detection from input images of driving scenes. We verify that the prioritized groups we find via intervention are indeed challenging for the base object detector and show that retraining with data collected from these groups helps inordinately compared to adding more IID data. We additionally confirm our hypothesis that interventions on the data distribution are preferred vis a vis data efficiency by comparing against random interventions. The latter are confounded by their propensity to stray from the data distribution.

Our contribution is a novel method using Masked Language Models (MLMs) to intervene on scene graphs of simulated scenes to causally uncover semantic groups of data upon which a detection model is likely to fail. Unlike sporadic failures, our found failure groups provide insight into the model's weaknesses and help us systematically improve the model.

2 Background

Notation Our objective is to ascertain the capabilities of a given object detection model ϕ. We represent a scene x as a triplet $x = (G, I, L)$ of a scene graph, scene image, and a set of bounding box labels, respectively. We flatten and discretize G to get the corresponding sequence $S \in \mathbb{N}^{O(d)}$ where d is the variable number of objects in the scene. The scene image $I \in \mathbb{R}^3$ is the RGB image of the scene as observed by the ego car and is deterministically defined by G. The label L is a set of ground truth bounding boxes $l_k \in \mathbb{R}^4$, where $k < d$ is the number of objects to identify in the scene. Scenes are drawn from a distribution $p_R(x)$ dictated by the scene generation process R. Examples of R include driving scenes from a particular city or simulations from AV simulators. We also need a per-example scoring function $f : (\phi, I, L) \rightarrow y$ as well as a threshold τ with which to gauge whether an intervention was detrimental.

Scene Graphs are 3D world representations, with nodes corresponding to objects and edges to hierarchical relationships between nodes, where the hierarchy is determined by physical presence (e.g. road is a parent of the vehicles on the road). Objects include the vehicles and pedestrians, the weather, the camera parameters, and the orientation of the ego agent. Each node has associated attributes, exemplified by continuous rotation, continuous position, discrete asset type, etc.

Object Detection reached a milestone with Faster RCNN (Ren et al., 2016). We use their approach as the representative state of the art detection module via the Detectron2 library (Wu et al., 2019).

Simulation is crucial to our method. We need a simulator that can initialize from G, have satisfactory traffic policies for autonomous vehicles, and return the current G on command. The chosen CARLA (Dosovitskiy, 2019) simulator satisfies these constraints and is ubiquitous in the field.

Masked Language Models (MLM) are density models for generating from sequences. Devlin et al. (2019) showed their tremendous efficacy in language generation. They are trained by receiving sequences of discrete tokens, a few of which are masked, and predicting what tokens are in the masked positions. Through this process, they learn the data distribution (Ng et al., 2020; Wang and Cho, 2019) of those sequences. At inference, they are fed a sequence with a chosen token masked and replace the mask with their prediction. We perform causal intervention on scenes by asking a pre-trained MLM to re-sample a single position from a scene graph - see Section 5.1 for details.

3 Method

We aim to improve object detection models by utilizing the advantages of AV simulators over collecting real world data, namely that they quickly synthesize scenes in parallel and that we have fine control over the synthesis. A naive approach is to generate lots of scenes, test detection on those scenes, and set aside the hard ones for retraining. A more advanced one is to use adversarial techniques to find hard scenes. Both approaches share two downsides: a) we are much more interested in plausible scenes than implausible ones and b) we still have the combinatorial challenge of understanding what in the scenes was the problem; only if we know why the error is happening can we understand if it is fixed after we retrain.

We propose an efficient procedure that tackles both of these concerns. We find hard groups of data for a trained model ϕ by taking interventions on scene sequences with a pre-trained MLM, the results of which are assessed with our surrogate scoring function f. Identifying challenging *scenes* does not provide insight into why nor how to fix ϕ, but asserting that a type of *intervention* is consistently hard narrows greatly where the model's difficulties lay and how to fix those difficulties. After finding challenging interventions, we utilize hard negative mining (Wang et al., 2017; Kumar et al., 2017; Wang et al., 2014), a common technique for improving models by first mining the data for the hardest examples and then emphasizing those examples by either retraining or fine-tuning. Our approach notably achieves this without relying on humans in the loop. We now explain in detail each of the components of our method.

The scoring function f should delineate between interventions that were minimal and those that caused a significant change in perception performance, with the assumption being that large negative (positive) changes imply that the intervention (reverse intervention) was detrimental to ϕ.

Our goal in designing f is to replicate the average precision (AP) score's intent, which values having few predictions with high intersection over union (IOU) to ground truth targets. Another goal was to evaluate entire scenes and not just target assets. This is important because even though our interventions are local to a node, they may still impact detecting any scene constituent. We choose not to use the popular mAP because it is defined over a dataset and thus is not suitable for identifying individual challenging scenes. To compute f, we get the model's predictions and order them by descending confidence. We sequentially align each prediction with the highest IOU ground truth. If IOU $> .05$, an empirically chosen threshold, then we mark this ground truth as claimed. The per prediction score is the product of the prediction's confidence and its IOU. We then take the mean over all predictions to get the model's final score on this example. This penalizes the model for having low confidence or a poor IOU and bolsters it for having high confidence on quality boxes.

Causal interventions on simulated scenes We draw from causal inference where interventions allow us to assess the causal links between the scene and the model's score. We change an aspect of a scene sequence S_i, such as a rotation or location of a specific vehicle, render this new scene S_i' as I', and then compute the $\delta = f(\phi, I', L') - f(\phi, I, L)$. We decide sufficiency by whether $|\delta| \geq \tau$, the aforementioned threshold parameter. After performing this procedure N times, filtering by sufficiency, and then grouping by the intervention type, we arrive at a prioritized list of challenging groups defined by either rotation, vehicle type, or weather pattern.

Generating interventions Uniformly random interventions produce unlikely scenes under the true data distribution[1]. Even if such an intervention would identity a weakness in the detector,

[1]Empirically, so do interventions sampled uniformly over the categories of interest.

its utility in improving our model is unclear because such a weakness may be very far from a realistic setting. We should favor finding groups that have higher probability under the data distribution. This is especially true for a limited model capacity because learning to detect flying cars and other unrealistic low-priority scenarios will take capacity away from pressing needs.

Formally, with $p_{\mathrm{R}}(x)$ as the generation process, y our surrogate score, and z a confounder that affects both x and y, we need to draw a counterfactual x' that is independent of z with which we can causally probe the model's weaknesses. Sampling from $p_{\mathrm{R}}(x)$ is challenging because retrieving the same scene again with just one change is difficult. We could act directly on the scene graph and model the conditional distributions of a single node change, then select changes via Gibbs sampling, and define interventions as sampling from these conditional distributions. Instead, we choose to discretize the scene (van den Oord et al., 2016; Engel et al., 2017; Razavi et al., 2019) and use masked language models (Dosovitskiy et al., 2021; Khan et al., 2021) because of their resounding recent success modeling distributions of combinatorial sequences relative to other approaches, as demonstrated clearly in language. Specifically, we train an MLM as a denoising autoencoder (DAE) to sample from $p_{\mathrm{R}}(x)$ (Bengio et al., 2013; Mansimov et al., 2019; Vincent et al., 2008), where the MLM operates on discretized scene graphs, flattened to be sequential. This provides a mechanism to sample counterfactuals from the data distribution DAE (Ng et al., 2020; Wang and Cho, 2019).

For each scene drawn from the original training distribution, the MLM infers a new scene close to the original distribution by making a singular semantic change over weather, vehicle asset type, rotation, or location. Because the MLM was trained to a low perplexity on data drawn from the distribution, it samples scenes that are likely under the original distribution $p_{\mathrm{R}}(x)$. Because it is not the exact distribution and errors will accumulate when applying many interventions sequentially, we intervene for just one step, equivalent to a single node change in the scene graph.

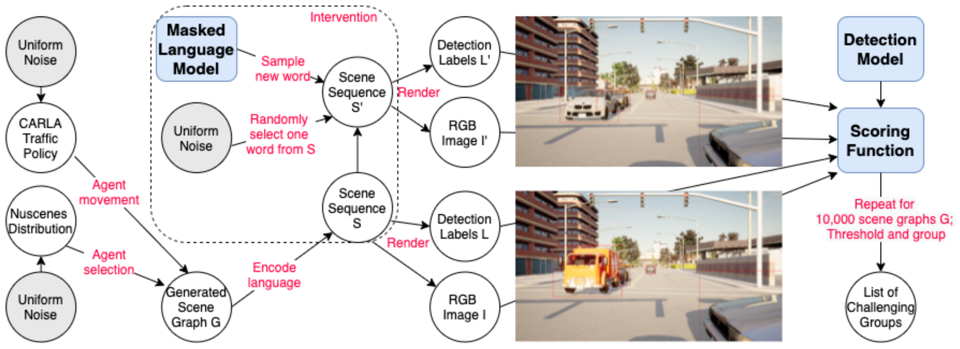

Figure 2: **A complete diagram of our approach**: We intervene on the scene by performing transformations like the pictured yellow truck becoming a white car and then evaluating the delta change in the object detection model's efficacy. The interventions are guided by a trained MLM. Repeat 10000 times and group the scores to attain an ordered list of challenging groups drawn from vehicle type, weather, and rotation.

4 Related Work

MLM as a generator While we believe we are the first to propose using an MLM as a generator in order to take causal interventions, Ng et al. (2020) generates from an MLM in order to augment training with generated examples. Mansimov et al. (2019) and Wang and Cho (2019) do so in order to generate high quality examples for use in downstream examples, with the former producing molecules closer to the reference conformations than traditional methods and the latter producing quality and diverse sentences.

AV Testing and Debugging See Corso et al. (2020) for a detailed survey on black-box safety validation techniques. While we believe that we are the first to take causal interventions in static scenes to test AV detection systems, multiple approaches (Ghodsi et al., 2021; Abeysirigoonawardena et al., 2019; Koren et al., 2018; Corso et al., 2019; O'Kelly et al., 2018) test AV systems through adversarial manipulation of actor trajectories and operate on the planning subsystem rather than detection. Wang et al. (2021a) generates adversarial scenarios for AV systems by black-box optimization of actor trajectory perturbations, simulating LiDAR sensors in perturbed real scenes. Prior research has focused on various optimization techniques for adversarial scenario generation through the manipulation of trajectories of vehicles and pedestrians. They either test only the planning subsystem in an open-loop manner or the AV system as a whole in a closed-loop fashion. Unlike our work, they do not allow a practitioner to glean causal factors for the errors. We focus on open-loop evaluation of AV perception and attempt to find causal factors for performance degradation through the generation of in-distribution counterfactuals with a masked language model trained on scene graphs. Concurrently, Leclerc et al. (2021) proposed a configurable system to diagnose vulnerabilities in perception systems through synthetic data generation. Compared to this work, we show how to generate complex scene manipulations using the MLM and study scenes of significantly higher complexity. It is possible in theory to implement our method within their framework.

Scene manipulation Ost et al. (2020) learn neural scene graphs from real world videos via a factorized neural radiance field (Mildenhall et al., 2020), while Kar et al. (2019); Devaranjan et al. (2020) generate scene graphs of AV scenes that match the image-level distribution of a real AV dataset as a means to produce realistic synthetic training data. All three can be seen as a precursor to our method for handling real world data. Dwibedi et al. (2017) generate synthetic training data for object detectors by learning to cut and paste real object instances on background images, which elicits a confounder because of how artificial the pasted scenes appear.

Adversarial detection is another way of viewing our work. Xie et al. (2017) showed that we should consider the detection task differently from the perspective of adversarial attacks, but did not explore finding root causes. Liu et al. (2018) use a differentiable renderer to find adverse lighting and geometry. Consequently, images appear stitched, a confounder to the natural distribution. Athalye et al. (2017) synthesizes real 3D objects that are adversarial to 2D detectors. They are limited to single objects, moving in the location, rotation, or pixel space, and do not identify causal factors. Zeng et al. (2019); Tu et al. (2020) synthesize 3D objects for fooling AV systems, both camera and LIDAR, with a goal to demonstrate the existence of one-off examples.

Challenging groups Improving the model to recognize found groups, potentially sourced from the distribution's long tail, is an important goal. Numerous methods (Ren et al., 2019; An et al., 2021) do this by re-weighting or re-sampling the training set, with Chang et al. (2021) focusing on detection. Sagawa et al. (2020) uses regularization and Wang et al. (2021b) uses dynamic routing and experts. All of these approaches require us to know the problematic groups in advance, which would only happen *after* applying our method. Further, they do not assess why the model is weak, but only seek to fix the problem. This makes it challenging to understand if the core issue has been addressed. Gulrajani and Lopez-Paz (2020) suggests that these approaches are not better than ERM, which is how we incorporate our found groups in Section 5.

5 Experiments

We run a suite of experiments analyzing our method and compare it against random interventions.

5.1 Setup

Model From Detectron2, we selected six battle-tested models: 18C4, 18FPN, 34C4, 34FPN, 50C4, and 50FPN. These are all common ResNet (He et al., 2015) architectures that include a litany of other attributes such as Feature Pyramid Networks (Lin et al., 2017). We created additional configurations that are 2x, 3x, 4x, and 5x wider versions of 50FPN, exemplified by 50FPN2x, for a total of ten tested model architectures. The C4 and FPN mix provided variation in model configuration, while the 18, 34, and 50 layer counts and their widths vary in capacity. We made minimal changes to the models to account for training on our dataset and with 4 gpus instead of 8. All models were trained for 90000 steps (8-9 hours) without pre-training; none reached zero training loss.

Datasets We first selected the CARLA preset map – Town03 or Town05. Then we randomly chose from among the pre-defined weather patterns. We sampled the camera calibration and the number V of vehicle assets according to the Nuscenes (Caesar et al., 2019) distributions, then placed those V vehicles, the ego agent, and $P = 20$ pedestrian assets, at random town waypoints suitable for the asset type. Finally, we attached the calibrated camera to the ego agent and enabled autopilot for all agents. We stabilized the scene for 50 timesteps after spawning, then recorded for 150 steps and saved every 15th frame. We needed the 2D ground truth boxes for each asset, but found the suggested approach[2] lacking because it frequently had trouble with occlusions and other challenging scenarios.

[2]See client_bounding_boxes.py in the CARLA library, commit 4c8f4d5f191246802644a62453327f32972bd536.

See the Appendix for heuristics we developed to help filter the ground truth boxes. For detection results on all charts, we report average precision (AP) over vehicle datasets.

MLM To train the MLM, we used the MaskedLMModel transformer architecture[3] from the FairSeq (Ott et al., 2019) library. We train and validate on held out IID datasets of sequences converted from scene graphs. Encoding the scene graph language required us to translate G with continuous node attributes into discrete sequence S. The first 10 tokens corresponded to weather attributes (cloudiness, precipitation, sun altitude angle, etc), the next 5 to camera intrinsics, and the following 15 to the ego agent. After these 30, we had a variable number of agents, each sequentially represented by 17 tokens. The two extra tokens for the non-ego agents were related to vehicle type, which was fixed for the ego agent. Although the 10 weather attributes were each continuous, we selected these vectors from 15 weather choices during training and so, with regards to the encoding, they each corresponded to discrete choices. Similarly, because the camera intrinsics were drawn from the (realistic) discrete Nuscenes distribution, their encoding was discrete.

The agent tokens had a set order. First is discrete type ('blueprint'), then continuous (x, y, z) locations and $(\text{roll}, \text{yaw})$ rotations. To discretize the locations, we first subtracted their minimum possible value. The resulting value (in $[0, 600)$) was encoded with tokens $w_0 \in [0, 5]$ for the hundreds place, $w_1 \in [0, 99]$ the ones, and $w_2 \in [0, 9]$ the decimal. This small precision sacrifice marginally impacted scene reproduction. Rotation used the same encoding, albeit is bounded in $[0, 360)$.

5.2 INTERVENTIONS

In this section, we investigate the relative ordering of groups by the MLM, where the order is determined by the degree to which that group is involved in a detrimental intervention.

Table 1 shows selected ordered results from the intervention procedure described in Section 3. We performed the procedure on $N = 10000$ held out scenes G_k where our ϕ was an 18C model trained on the base 10000 subset from Town03 and $\tau = 0.2$. We additionally filtered the categories to have at least 20 entrants.

Intervention	Percent > 0.2	Total
Tier 1: Likely Challenging Groups		
DiamondbackBike	24.4	123
Cloudy Dark	19.4	36
GazelleBike	18.9	122
Cloudy Dark Puddles	17.2	29
CrossBike	16.5	121
Rotation - 178.6	15	20
Rotation - 121.3	13.0	23
Tier 2: Borderline Groups		
KawasakiBike	6.5	92
Cybertruck	6.4	94
Carla Cola	6.0	198
Sunny Puddles	5.4	56
Tier 3: Easy Groups		
Citroen C3	1.6	188
Mercedes CCC	1.0	206

Table 1: Illustrative table of interventions, ordered by the percent of time that they were involved in a high magnitude δ intervention. Section 5.3 suggests that between 6.0 and 6.4 is where our cutoff resides.

On the left side we see the intervention taken, for example changing a single agent type to a Cybertruck or changing the weather such that it is now sunny with reflective puddles. The second column shows the probability that the intervention produced a $\delta \geq 0.2$. We include both when the change was *to* that target and the delta was negative as well as when it was *from* that target and the delta was positive. The last column in the table reports how many times in total this intervention occurred in the 10000 scenes.

Summarizing the table, we find that a handful of asset switches appear to be detrimental for the model according to this metric. Small bikes had an outsized effect, as did cloudy weather and the rotations where a car faced the ego agent or turned to the left. Just after the last bike are two large vehicles, the Cybertruck and the Cola Car. The specificity of the weathers and rotations are because they are translations of our discretization. Practically, there is a range of rotation and weather values around the group that would all suffice. Finally, we do not include location results in the table because the MLM frequently re-positioned the asset outside the camera's view. This said more about the asset than it did about the location and was rife with confounders based on what was behind that asset. We could have localized the location interventions more by masking MLM options, but leave that for future work.

[3]See masked_lm.py#L30 in the FairSeq library, commit 1bba712622b8ae4efb3eb793a8a40da386fe11d0

5.3 ANALYSIS

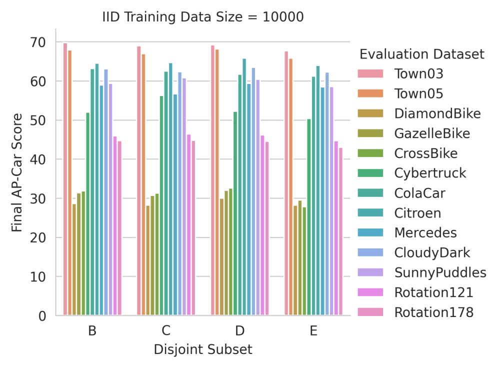

Figure 3: Test results with config 18C4 when training on disjoint IID subsets. Results are consistent, suggesting that the harder groups - bikes, rotations, and cybertruck - are ubiquitously hard.

After obtaining candidate groups from the designed interventions, we investigated the effect of modifying the data sampling procedure to increase the prevalence of these groups by building and evaluating datasets sampled from the MLM training set. For asset groups, for each datum, we uniformly sampled $n_{\text{v}} \in [3, 6]$ vehicles selected from the scene. We then randomly chose vehicles $v_0, v_1, \ldots, v_{n_{\text{v}}}$ in that scene, including vehicles that may not be in the camera's purview, and changed them to be the target group. So as to not accidentally introduce a bias through the random process, we selected the same vehicles v_k for all group datasets. For rotation groups, we chose those same vehicles but rotated them to be the target rotation instead of switching their asset. For weather groups, we changed those scenes to have the target weather instead.

Does our method correlate with AP score? Figure 3 shows evaluation results on these groups when training 18C4 on four disjoint 10000 sized subsets of the data. The models performed best on the IID data from Town03 and just a little bit worse on the same from Town05. Further, they did exceptionally well on those two datasets, validating that they were trained sufficiently. The group results are mostly in line with our expectations from the interventions - the models did well on Citroen and Mercedes, poorly on the rotations, and terribly on the bikes. There is a large jump from the reasonable results on ColaCar and SunnyPuddles to the mediocre results on Cybertruck, which is directionally correct per Table 1. However, the strong results on CloudyDark are surprising.

Summarizing, if the threshold for choosing a group is between 5.5% and 6.5% and we focus on interventions affecting vehicles directly (rotation and type), then our method correlates well with empirical results. This does not mean that we have found the exact causes plaguing the model, but that we have narrowed them greatly. The model's regression when changing a car to a bike may be because it performed poorly on bikes. It may also be because the car was occluding another vehicle or that it itself was not occluded. This is especially true in light of the weather results suggesting that weather is not a conclusive factor. Finding the exact cause is a difficult problem, even in simple settings Arjovsky et al. (2020). We restrict our scope to this level of understanding and leave improvements for future work.

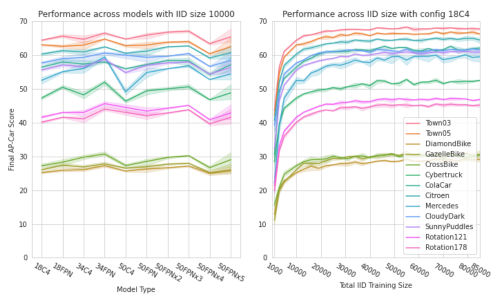

Figure 4: Independently increasing the model capacity (left) and increasing the data size (right). No model distinguished themselves and we quickly taper in how effectively the model utilizes the data. We consider the dip in the capacity chart to be an artifact of the training procedure and using the same settings for all models.

Can we address these issues by increasing capacity? Recent papers Zhai et al. (2021); Bahri et al. (2021) suggest that scaling our models will improve results. An affirmative answer would mean we would not need to collect more data. While it is possible that we did not scale enough or use the right architectures, the left side of Figure 4 suggests a negative answer. We see that no model was distinguished with respect to their final values when training on 10000 IID examples.

What if we increased IID data? This is preferable because IID data is easier to collect than group specific data. The right side of Figure 4 suggests this will not be sufficient. Test efficacy on town and group data jumped from 1000 to 10000 IID examples, but then slowed precipitously. Figure 8

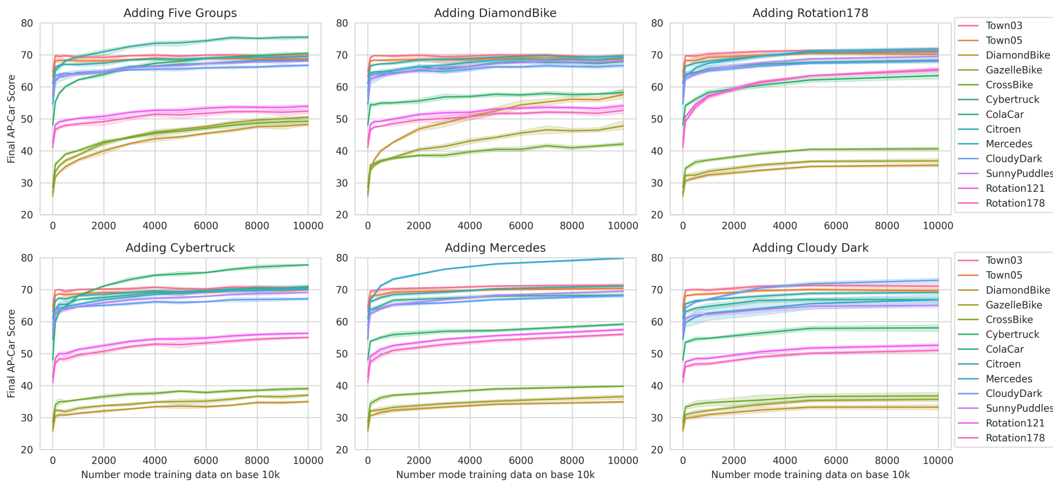

Figure 5: Results of training 18C4 on the base IID 10000 training set plus additional group data. The five groups in the top left (Cybertruck, Cola Car, Diamondback, Gazelle, and Crossbike) were added equally. For all charts, adding any one group improved all of the other evaluation scores, and at no point did we lose efficacy on the IID data as a whole. Figure 10 (Appendix) zooms in on the initial jump.

(Appendix) affirms that this is unlikely to change by suggesting that the percentage of representation of the group is what matters, rather than absolute count.

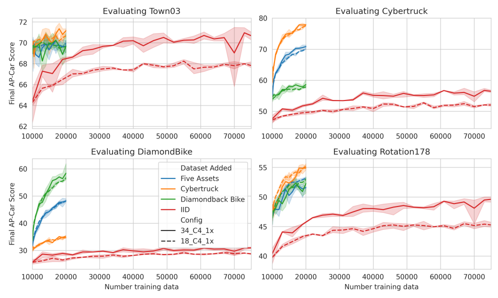

Figure 6: How much IID data is required to match a small amount of extra hard group data. Top left shows 20000 more IID data was required to reach par on IID with 250 group data. Bottom left shows that we never reached the same level on Diamondbacks with IID data as with adding Cybertrucks, let alone actual bikes.

What if we increased data and capacity simultaneously? Results remained negative, as seen in Figures 7 and 9 (Appendix). The left graphic in Figure 7 evaluates all of the models on 85000 examples and the right one hones in on the 34C4 model, showing its results across a range of IID data counts. Three aspects stand out. The first was that all of the models have similar evaluation scores. The second is that they all struggled on the harder groups. And the third, seen more clearly in Figure 9, is that more data yielded a small accretive effect. All else equal, adding data may be better than adding model capacity.

Using group data We expect that when we add data from the groups to the training set that we will address the issues. The top left plot in Figure 5 confirms that to be true. We added an even amount of each group to the base 10000 IID subset and see that every group improved without impacting the Town03 and Town05 results. The other plots in Figure 5 show what happens when we add in training data from any one group M. This predictably improved the model's results on M's validation set. It surprisingly also improved results on *all* of the other M' and the Town data. The improvement to M' is smaller than that to M, but it is notable. The gains for a specific group were more pronounced for like groups - adding data from a biker group (Diamondback, Omafiets, Crossbike) improved the other biker groups more than adding data from the heavy car groups (Cybertruck, Colacar), and similarly for the heavies. Adding rotation groups helped ubiquitously albeit not as much as adding a bike group did for the other bikes. The least effective fix was adding the CloudyDark weather mode. Figure 8 shows that these trends persisted for a base of 85000 IID data as well.

Comparison with random interventions As we alluded to in Section 3, taking random interventions is problematic because whether the group is reasonable for the distribution will be a confounder. We wish to prioritize the found groups to be those that are more likely seen in the wild. We show here that this is true by taking the 10000 source scenes used for the MLM interventions and applying random manipulations of the same type. For example, if we changed agent a_j's vehicle type in $G_k \rightarrow G_k^{\text{MLM}}$, then we changed a_j to a random vehicle type in $G_k \rightarrow G_k^{\text{Random}}$.

Table 2 shows results for random and MLM interventions over the same assets specified in Table 1. Observe that the assets were ordered incorrectly with CarlaCola higher than both Cybertruck and Kawasaki Bike. Second, note that Random had a higher percent of high threshold events. This held in general, with 13.2% of random interventions impacting the model versus 10.2% of MLM interventions. We hypothesize this is because random re-sampling of elements of the scene graphs corresponded to sampling from a data distribution that does not faithfully represent the original training distribution. A 3% difference is large with respect to how much extra work would be required by humans combing through the data for plausibility and whether to include in retraining.

Intervention	MLM	Rand	Rand Total
CrossBike	16.5	19.0	126
GazelleBike	18.9	18.7	171
DiamondbackBike	24.4	15.8	152
Carla Cola	6.0	8.1	210
Cybertruck	6.4	6.8	176
KawasakiBike	6.5	6.6	121
Citroen C3	1.6	3.5	197
Mercedes CCC	1.0	3.8	183

Table 2: Results for MLM and Random asset intervention strategies, ordered by the percent of times that they were involved in a high magnitude δ *random* event. While the top three are the same, Random flubbed the dividing line by placing a) Cybertruck above Kawasaki and b) Carla Cola well ahead of both. Its failure rate for the easy cars was much higher and, in general, posited 3% more failures than MLM. All told, its results created more need for human verification and testing and reduced the degree of automation that we could employ to find hard groups.

Figure 11 (Appendix) shows density plots for rotation and cloudiness interventions, conditioned on the intervention having been detrimental. We use density plots to demonstrate the differences between Random and MLM because these interventions are continuous for Random. For rotation, there was a mostly steady plateau for Random while MLM showed a clear single group aligned with the bi-modal humps in Original. For weather, Original and MLM were almost overlapping and, while Random was similarly bi-modal, its shape was less pronounced and more even as expected. These both reinforce our claim that the advantage of MLM is that it gears us towards higher priority groups to fix that are in line with the actual data distribution.

Why do these groups exist? With causal groups in hand, we can now ascertain why our models were failing. For the bikes, it was because they are underrepresented in Nuscenes. For Rotation121, the model infrequently trained on turning cars due to the town layout. For Rotation178, the model infrequently trained on cars facing it due to the traffic policy and car quantity. The Cybertruck was challenging because it is large and caused occlusion labeling issues. The ColaCar also had this issue, just not as severely as the Cybertruck. Without the groups, these issues can only be hypothesized.

6 Conclusion

Combining causal interventions, MLMs, and simulation, we presented a novel method that finds challenging groups for a detection model in foresight by having the MLM resample scene constituents. These interventions help identify and prioritize groups with poor performance without humans in the loop. Our approach is a significant step towards addressing safety-critical concerns in AV. Beyond AV, we think the associated software[4] will benefit the causality community because the current state of the art (Koh et al., 2020) involves static datasets with low complexity tasks.

Our method has limitations. We cannot yet apply it to real world data because we need full control over the scenes for the MLM to properly operate Ost et al. (2020) is a step towards overcoming this concern. Until then, the sim2real gap (Sadeghi and Levine, 2017; Jakobi, 1998) is ever-present. Another limitation is that our method only works on first-order interventions because the samples drift from the data distribution, but intervening multiple times is necessary for understanding complicated causal interactions. Each of these limitations are also potential future directions. A final one is understanding better why many groups improved when adding a single group, which remains a compelling question.

[4]We will release a github repository after the review period.

References

Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277, 2019. doi: 10.1109/ICRA.2019.8793740. 4

Jing An, Lexing Ying, and Yuhua Zhu. Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients, 2021. 5

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020. 7

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. URL http://arxiv.org/abs/1707.07397. 1, 5

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *CoRR*, abs/2102.06701, 2021. URL https://arxiv.org/abs/2102.06701. 7

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. *CoRR*, abs/1305.6663, 2013. URL http://arxiv.org/abs/1305.6663. 4

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 5

Nadine Chang, Zhiding Yu, Yu-Xiong Wang, Anima Anandkumar, Sanja Fidler, and Jose M. Alvarez. Image-level or object-level? a tale of two resampling strategies for long-tailed detection, 2021. 5

Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019. 4

Anthony Corso, Robert J Moss, Mark Koren, Ritchie Lee, and Mykel J Kochenderfer. A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*, 2020. 4

Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Learning to generate synthetic datasets. In *ECCV*, 2020. 5

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 3

Alexey Dosovitskiy. carla-simulator/carla, Aug 2019. URL https://github.com/carla-simulator/carla/blob/master/PythonAPI/examples/client_bounding_boxes.py. 3

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. 4

Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017. 5

Jesse H. Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with wavenet autoencoders. *CoRR*, abs/1704.01279, 2017. URL http://arxiv.org/abs/1704.01279. 4

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 13

Zahra Ghodsi, Siva Kumar Sastry Hari, Iuri Frosio, Timothy Tsai, Alejandro Troccoli, Stephen W Keckler, Siddharth Garg, and Anima Anandkumar. Generating and characterizing scenarios for safety testing of autonomous vehicles. *arXiv preprint arXiv:2103.07403*, 2021. 4

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization, 2020. 5

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 5

Nick Jakobi. Running across the reality gap: Octopod locomotion evolved in a minimal simulation. In Philip Husbands and Jean-Arcady Meyer, editors, *Evolutionary Robotics*, pages 39–58, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49902-2. 9

Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *ICCV*, 2019. 5

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey, 2021. 4

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. *CoRR*, abs/2012.07421, 2020. URL https://arxiv.org/abs/2012.07421. 9

Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018. 4

B. G. Vijay Kumar, Ben Harwood, Gustavo Carneiro, Ian D. Reid, and Tom Drummond. Smart mining for deep metric learning. *CoRR*, abs/1704.01285, 2017. URL http://arxiv.org/abs/1704.01285. 3

Guillaume Leclerc, Hadi Salman, Andrew Ilyas, Sai Vemprala, Logan Engstrom, Vibhav Vineet, Kai Xiao, Pengchuan Zhang, Shibani Santurkar, Greg Yang, et al. 3db: A framework for debugging computer vision models. *arXiv preprint arXiv:2106.03805*, 2021. 5

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017. 5

Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Adversarial geometry and lighting using a differentiable renderer. *CoRR*, abs/1808.02651, 2018. URL http://arxiv.org/abs/1808.02651. 5

Elman Mansimov, Alex Wang, and Kyunghyun Cho. A generalized framework of sequence generation with application to undirected sequence models. *CoRR*, abs/1905.12790, 2019. URL http://arxiv.org/abs/1905.12790. 4

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020. 5

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. SSMBA: self-supervised manifold based data augmentation for improving out-of-domain robustness. *CoRR*, abs/2009.10195, 2020. URL https://arxiv.org/abs/2009.10195. 3, 4

Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, John Duchi, and Russ Tedrake. Scalable end-to-end autonomous vehicle testing via rare-event simulation. *arXiv preprint arXiv:1811.00145*, 2018. 4

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. 2020. 2, 5, 9

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019. 6

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*, pages 14866–14876, 2019. 4

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning, 2019. 5

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image, 2017. 9

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2020. 5

James Tu, Mengye Ren, Siva Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection, 2020. 5

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016. 4

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294. 4

Alex Wang and Kyunghyun Cho. BERT has a mouth, and it must speak: BERT as a markov random field language model. *CoRR*, abs/1902.04094, 2019. URL http://arxiv.org/abs/1902.04094. 3, 4

Chong Wang, Xue Zhang, and Xipeng Lan. How to train triplet networks with 100k identities? *CoRR*, abs/1709.02940, 2017. URL http://arxiv.org/abs/1709.02940. 3

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. *CoRR*, abs/1404.4661, 2014. URL http://arxiv.org/abs/1404.4661. 3

Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021a. 1, 4

Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X. Yu. Long-tailed recognition by routing diverse distribution-aware experts, 2021b. 5

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 2

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection, 2017. 1, 5

Xiaohui Zeng, Chenxi Liu, Yu-Siang Wang, Weichao Qiu, Lingxi Xie, Yu-Wing Tai, Chi Keung Tang, and Alan L. Yuille. Adversarial attacks beyond the image space, 2019. 5

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers, 2021. 7

A Appendix

A.1 Table of notation

Symbol	Description
ϕ	Detection model
x	Scene
G	Scene graph
$d \in \mathbb{N}$	Number of objects in scene
$S \in \mathbb{N}^O(d)$	Sequence encoding of scene graph
$I \in \mathbb{R}^3$	Scene image
$L \in \mathbb{R}^{4 \times d}$	Scene bounding box labels
$l_k \in \mathbb{R}^4, k < d$	The bounding box of the kth object
R	Scene generation process
$p_R(x)$	Distribution over scenes
$f : (\phi, I, L) \to \mathbb{R}$	Per-example scoring function
$\delta \in \mathbb{R}$	The change in score by intervention: $\delta = f(\phi, I', L') - f(\phi, I, L)$
$\tau \in \mathbb{R}$	Threshold value for classifying interventions as detrimental

Table 3: Table of notation

A.2 Dataset details

CARLA refuses to spawn agents that collide with the environment, including the ground. To ensure agents are grounded, for any asset that causes a spawn collision, we increase its Z coordinate and try to spawn again . This approach allows us to place every agent on the map, albeit some of the conflicting agents have to 'drop' from above, and consequently we wait for 50 timesteps so those agents can settle. In that duration, the autopilot policy guides the agents to satisfactory positions. After those 50 steps, we then record for another 150 steps and save every 15th frame. The resulting episodes each have ten frames following an initial distribution influenced by Nuscenes and CARLA, and a traffic policy influenced by only CARLA determining the final settled distribution.

We then need the 2D ground truth boxes for each asset. We found the existing suggested approach lacking because it frequently has trouble with occlusions and other challenging scenarios. See below for heuristics we developed to help filter the ground truth boxes. While they are not airtight, the resulting ground truths were qualitatively perceived as more reliable.

- Filter Height: We require that the final $2d$ box is at least 30 pixels. This is in between the easy (40) and medium/hard (25) settings on KITTI Geiger et al. (2012).
- Max Distance: We require that the ground truth detection not be more than 250 meters away. We enforce this through the use of a depth camera attached to the ego agent.
- Visible Pixel Percent (VPP) and Min Visible Count (MVC): The 2D box is attained by pairing the 3D box with the camera's calibration. With the latter, we get the closest point P to the ego agent. We then get the depth camera's output at the 2D box. VPP asks what percent t of that box is closer than P and filters it if $t \geq 80$, ensuring that at least 20% of the object is not occluded. MVC asks how many pixels q are further than P and filters it if $q < 1300$, ensuring that the occluded object is big enough.

A.3 Supporting charts

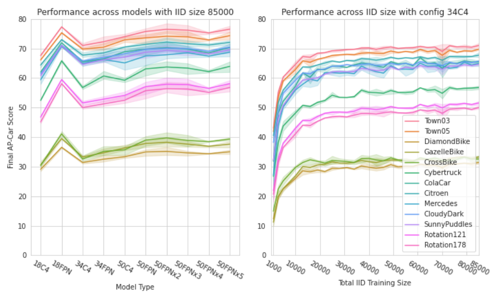

Figure 7: Charts showing increasing both data and model capacity at the same time. The left side ranges over model capacity with maximum IID data size (85000), while the right side ranges over IID data size with a bigger model - 34C4.

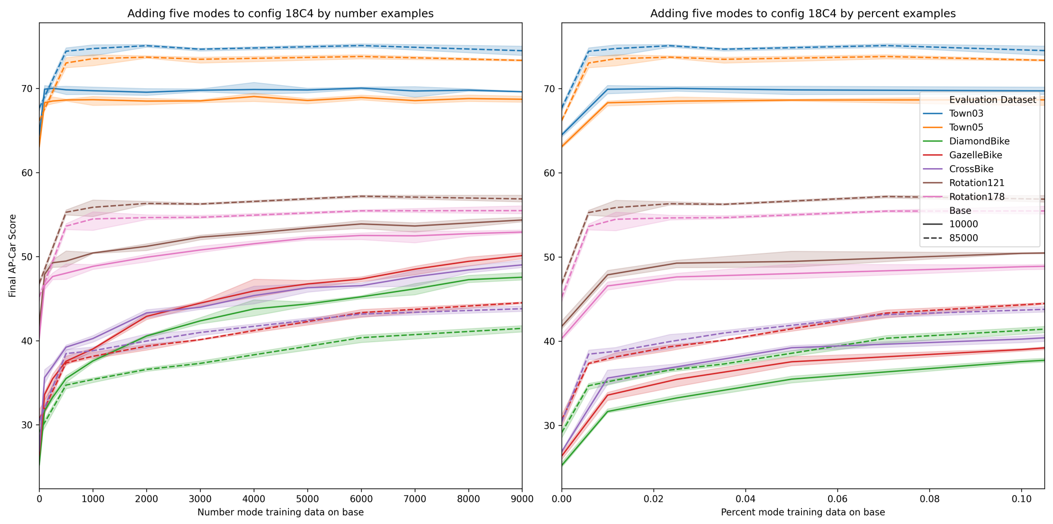

Figure 8: Performance of 18C4 on select test sets when adding mode data from the three bikes, the ColaCar, and the Cybertruck on top of either 10000 or 85000 base IID data. Towards improving the results, these two charts show that it is not the absolute count of the mode data that is important but rather the percent of it relative to the IID data. We see that in how the trendlines for the two bases are only consistent in the percent chart. The other modes are not shown for clarity but it holds in general.

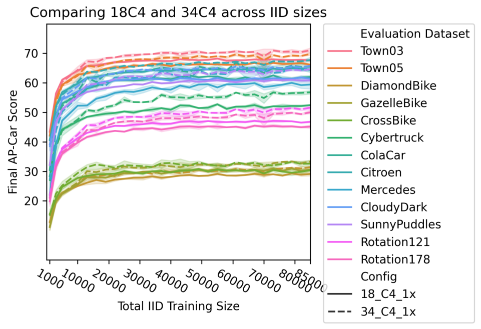

Figure 9: We can see that the model size does matter in that for every group the 34C4 model improves over the 18C4 model. However, the increase is quite small and the data quality and quantity appear to matter much more.

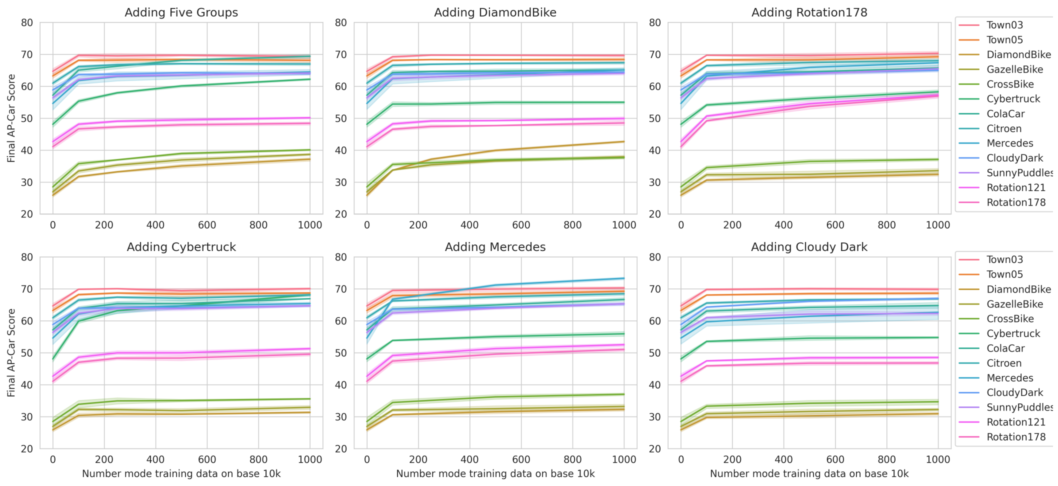

Figure 10: Results adding mode data to the base IID 10000 training set. This is the same as Figure 5 but zoomed into just $[0, 1000]$. The five modes in the top left are the Cybertruck, Cola Car, Diamondback, Gazelle, and Crossbike, each added in equal proportion.

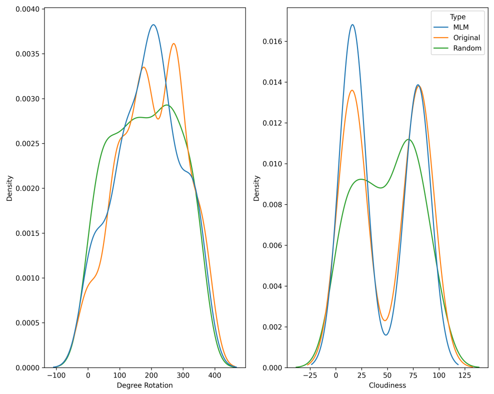

Figure 11: Comparing rotation and weather results for MLM and Random intervention strategies. We see that MLM fits with Original much better than Random does. Further, Random has a much wider berth of possible problematic modes, which is a concern given practical limits to model capacity and data budgets.