Causal BERT: Improving object detection by searching for challenging groups

Cinjon Resnick
NYU
cinjon@nyu.edu

Or Litany
NVIDIA

Amlan Kar
NVIDIA

Karsten Kreis
NVIDIA

James Lucas
University Toronto

Kyunghyun Cho
NYU

Sanja Fidler
NVIDIA

Abstract

Autonomous vehicles (AV) often rely on perception modules built upon neural networks for object detection. These modules frequently have low expected error overall but high error on unknown groups due to biases inherent in the training process. When these errors cause vehicle failure, manufacturers pay humans to comb through the associated images and label what group they are from. Data from that group is then collected, annotated, and added to the training set before retraining the model to fix the issue. In other words, group errors are found and addressed in hindsight. Our main contribution is a method to find such groups in foresight, leveraging advances in simulation as well as masked language modeling in order to perform causal interventions on simulated driving scenes. We then use the found groups to improve detection, exemplified by Diamondback bikes, whose performance we improve by 30 *AP points. Such a solution is of high priority because it would greatly improve the robustness and safety of AV systems. Our second contribution is the tooling to run interventions, which will benefit the causal community tremendously.*

1. Introduction

Our guiding motivation in this paper is to understand where a detection module has poor performance *before* it enters production. We focus on a realistic setting – autonomous vehicles (AV). These systems are becoming more apparent in our lives as they enter mainstream. It is vital that this module is extremely capable, however biases in the training process such as in the data distribution, the chosen architecture, or the hyperparameters will lead to the model having groups of data where the error is much higher than what it is in expectation over the full dataset. These gaps in its scene understanding lead to faulty operation, disengagements, and emergency consequences, and we would prefer that they be identified before deployment. We focus in this work on semantically meaningful groups such as specific weather patterns, vehicle types, and vehicle positioning. These are high priority problems to the community because finding challenging groups in advance informs efficient data acquisition, in turn improving the robustness and safety of AV systems. However, naïvely searching for them is difficult because the search space is so large.

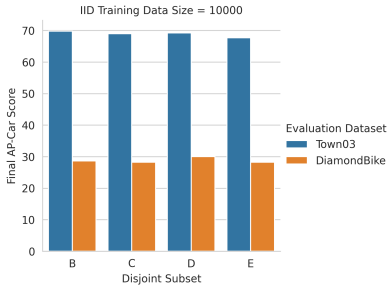

Figure 1: After training models on any of four disjoint 10000 IID subsets from CARLA's Town03 environment, they do well on the Town03 validation set and poorly on a validation set biased towards Diamondback bikes. This establishes that there is at least one challenging group for the model. Our method will now find other challenging groups.

Recent advances in masked language models [12] (MLM) have granted us understanding of sequences such that we now reliably generate fluent language (see Section 2). Advances in AV simulation frameworks (e.g. CARLA [13]) have granted us fine-grained control over scenes. We leverage both of these by first encoding simulated scenes from synthetic data as flattened sequential scene graphs of symbolic tokens. We then mask out select tokens and use the trained MLM to infer new ones. This

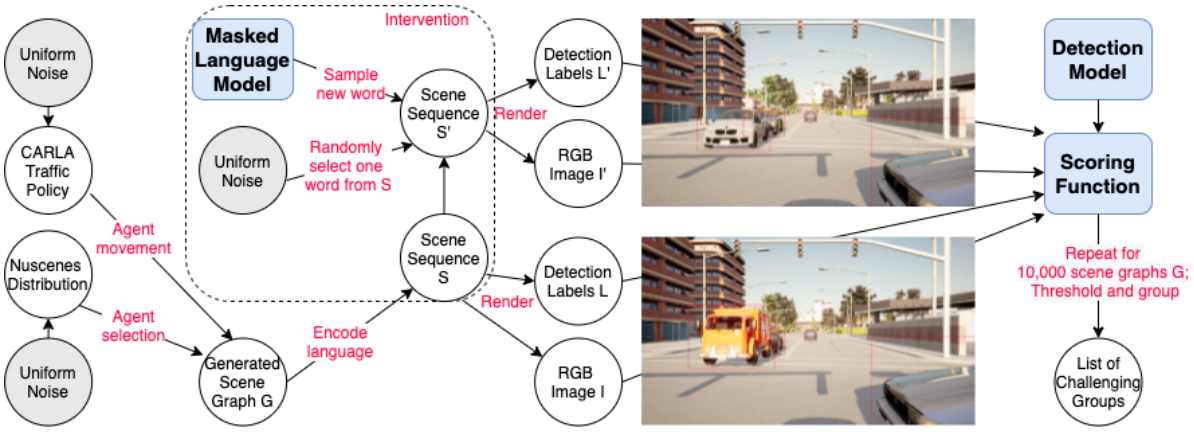

Figure 2: **Finding challenging groups for a detection model**: We generate 10000 scene graphs G and encode those graphs into sequences S using a pre-defined deterministic language. For each S, we intervene by randomly selecting a single node from either the weather or a random agent's location, rotation, or vehicle type, and then having our trained MLM resample that node. We render images and labels from S and S' with CARLA, and feed them to the scoring function f_ϕ for assessment. After applying a threshold and grouping these assessments, we attain an ordered list of challenging groups for our model. The area inside the dotted box is the intervention machinery. The MLM is a modular component, as we show in Section 5 in order to compare with a random approach.

intervention, e.g. changing the weather from sunny to dark, allows us to test causal interactions in a scene and see which are the most impactful.

Our primary contribution is a method for taking such interventions on the data distribution using MLMs operating on scene graphs. We employ this procedure to find which groups in the data distribution are most challenging for the model and confirm that the difficulty of these groups with respect to average precision (AP) aligns with our approach's predictions. We then show that adding independent and identically distributed (IID) data or more model capacity does not address the problem. Predictably, including data from the groups in training helps the model's performance on that group. Surprisingly, we find that including just one group in training helps every other group as well.

The MLM ensures that our interventions stay close to the true data distribution. This is critical, as we show via a comparison to random interventions that tend to produce examples which are highly unlikely under the true data distribution. Consequently, using random interventions requires a much higher budget for data collection and model capacity than using the MLM interventions.

Our second contribution is a toolbox for causal research. We developed software[1] that allows us to principally test questions of model capacity, dataset size, and dataset makeup. We did this in a simulated environment upon which we take causal interventions in a challenging and practical application setting. Beyond the AV community, we think this toolbox will benefit the causality community because the state of the art [21] involves static datasets with low complexity tasks.

[1]We will release this publicly after submission.

2. Background

We train detection models on synthetic datasets from simulated AV environments, and take interventions using an MLM on scene graphs of those environments.

For the detection models, which operate on RGB images, we use the Detectron2 library [44]. It provides a plethora of battle-tested architecture choices along with suggested training and testing configurations. We select six models: 18C4, 18FPN, 34C4, 34FPN, 50C4, and 50FPN. These are all common ResNet [18] architectures that include a litany of other attributes such as Feature Pyramid Networks [26]. See the Appendix for details. We create 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 allows us variation in model configuration, while the 18, 34, and 50 layer counts and their width increases provide variation 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 pretraining; none reached zero training loss.

For data, we use the ubiquitous CARLA simulator and produce synthetic data from the preset Town03 or Town05 maps. Besides RGB images and COCO [25] annotations for training the detectors, we also use *scene graphs* to let the MLM semantically manipulate the scene. These are representations of an image in a deterministic tree structure with symbolic elements that each have symbolic or numeric attributes. We flatten the tree and use absolute positioning.

MLMs are trained by receiving sequences of discrete tokens, a small number of which are masked, and predicting

what tokens should be in the masked positions. Through this process, they learn the data distribution [30, 39] of those sequences. At inference, they are fed a sequence with a chosen token masked and replace the mask with their prediction. We use FairSeq [33] and specifically the base MaskedLMModel transformer architecture[2] to train the MLM. We cannot use pre-trained models because our data is original, but instead train and validate on held out IID datasets of sequences converted from scene graphs (Section 3). These are converted to the FairSeq format before using the standard training pipeline.

For detection results on all charts, we report average precision (AP) over vehicles as returned by Detectron2.

3. Method

Figure 1 compares the performance of trained models on held-out data sampled from the training distribution and held-out data with increased occurrence of a challenging instance type, a brand of bikes called Diamondbacks. The performance gap between the sets is stark. We expected in advance that bikes would be difficult because they are underrepresented in the training distribution and are motivated to find similarly challenging groups with limited human involvement, especially groups that are not so easily hypothesized as the bikes.

One approach to doing this would be to iterate through the training set and gather examples for which the model ϕ gives poor results. Given a simulator R, if x is a triplet of (scene graph G, RGB image I, label L), and $p_{\mathrm{R}}(x)$ is the scene generation process, then we sample $(G, I, L) \sim p_{\mathrm{R}}(x)$. To ascertain the model's ability on this example, we also need a per-example scoring function $f : (\phi, I, L) \to y$. Repeat this over the full training set to attain tuples (G_i, y_i).

We immediately run into a problem when trying to glean causal factors from this data because each G_i has many scene constituents, such as the agents' attributes, the time of day, rain, etc, whose interactions scale combinatorially. The core issue is that we know the y_i at too high of a level to parse which constituents were problematic. Even if we associate a score per ground truth bounding box, we still would not be able to disambiguate the cause because there are a multitude of reasons why a specific detection could fail that are a consequence of the interactions within a scene. Understanding those causes is important because it would help us in both finding more data that would patch the deficiency but also downstream in grokking whether we have sufficiently addressed the issue.

Causal interventions on simulated scenes We take inspiration 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 G_i, such as a rotation or location of a specific car, render this new scene G_i' as RGB image I', and then compute the $\delta = f(\phi, I', L') - f(\phi, I, L)$. If this δ is sufficiently negative (or positive), then the target (or original) attribute in the intervention is potentially problematic for ϕ. We decide sufficiency by whether $|\delta| \geq \tau$, where $\tau > 0$ is a threshold parameter. After performing this procedure N times and grouping by the intervention type, we attain an ordered list of challenging groups at the semantic level of specific rotations, car types, or weather patterns.

What determines the intervention? Intervening randomly may produce data that is unlikely under the true distribution, which will make it challenging to draw conclusions about the difficulty of the example with respect to its actual utility in improving our model. In other words, we would like for the model to be able to detect flying cars, but given a computational budget, that out-of-distribution group is much lower priority than detecting motorbikes in the dark, and so we should favor finding the latter over the former. 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; it is also true in practice because detection labeling is expensive and our budget for data outside of the simulator is never infinite.

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. Sampling from $p_{\mathrm{R}}(x)$ is challenging because retrieving the same scene again with just one change is difficult. Instead, we train a denoising autoencoder (DAE) to sample faithfully from $p_{\mathrm{R}}(x)$ [6, 28, 38]. This sampling is satisfied by using a masked language model (MLM) as our DAE [30, 39].

The MLM is trained on flattened scene graphs where the discrete tokens are representations of weather, agent asset types, rotations, and locations (details below). Figure 2 shows how, at inference time, we convert G to a flat sequence S and re-sample masked tokens to create counterfactual scenes. Because the model was trained to a low perplexity on data drawn from the distribution, it faithfully samples from the original distribution $p_{\mathrm{R}}(x)$. Because our model is not the exact distribution and errors will accumulate when applying many interventions sequentially, we intervene for just one categorical step, equivalent to a single node change in the scene graph.

Scene-sequence encoding Encoding the scene graph language requires us to translate G with continuous node attributes into discrete S. The first 10 tokens correspond to weather attributes (cloudiness, precipitation, sun altitude

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

angle, etc), the next 5 are camera intrinsics, and the following 15 represent the ego agent. After these 30, we have a variable number of agents, each sequentially represented by 17 tokens. The two extra tokens for the non-ego agents are related to vehicle type, which is fixed for the ego agent. Although the 10 weather attributes are each continuous, we select these vectors from 15 weather choices during training and so, with regards to the encoding, they each correspond to discrete choices. Similarly, because the camera intrinsics were drawn from the (realistic) discrete Nuscenes [7] distribution, their encoding is discrete.

The agent tokens have a set order. First is the discrete type ('blueprint'), then the continuous (x, y, z) locations and $(\text{roll}, \text{yaw})$ rotations. To discretize the locations, we first subtract their minimum possible value. The resulting value is in $[0, 600)$ and we encode it with three tokens w_k: $w_0 \in [0, 5]$ represents the hundreds place, $w_1 \in [0, 99]$ the ones, and $w_2 \in [0, 9]$ the decimal. This small sacrifice of precision marginally impacts scene reproduction. Rotation uses the same encoding, albeit it is bounded in $[0, 360)$.

Scoring We require a per-example scoring function f to quantify the intervention delta $\delta = f(\phi, I', L') - f(\phi, I, L)$. Our goal was to replicate the 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 first get the model's predictions and order them by confidence, highest to lowest. We sequentially align each prediction with the highest IOU ground truth. If IOU $> .05$, 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.

Summarizing, we propose finding hard groups for a trained model ϕ by taking interventions on scene sequences through the use of an MLM and assessing the results with our own surrogate scoring function f. Our MLM infers a new scene close to the original distribution with semantic changes such as weather, agent asset type, location, or rotation, and then the scores from f delineates between interventions that were minimal and those that caused a high delta change. The assumption is that high negative delta changes imply that the intervention was deleterious to ϕ and high positive changes are the same in reverse. Asserting that a particular scene is hard does not provide any insights into why nor how to fix it; Asserting that a type of intervention is consistently hard narrows greatly where the model's difficulties lie without ever including humans in the process *and* suggests a route to fixing those difficulties. After finding this priority list, we address the groups via hard negative mining [40, 23, 41], a common technique for improving models by first mining the data for the hardest examples and then emphasizing those examples in the training set by either retraining or fine-tuning.

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. [30] generates from an MLM in order to augment training with generated examples. Mansimov et al. [28] and Wang and Cho [39] 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 We discover perception vulnerabilities in the AV detection system through scene manipulation using MLMs. Concurrently, 3DB [24] proposed a configurable system to diagnose vulnerabilities in perception systems through synthetic data generation. While theoretically possible to implement our proposal within their framework, we additionally show how to generate complex scene manipulations using the MLM and study scenes of significantly higher complexity. Multiple approaches test AV systems – usually the planning subsystem – through adversarial manipulation of actor trajectories. We believe we are the first to take causal interventions in static scenes to test AV detection systems. Ghodsi et al. [16] manipulate trajectories of actors to discover avoidable safety critical scenarios for an AV using their proposed scenario complexity metrics. Abeysirigoonawardena et al. [1] perform Bayesian optimization to discover adversarial driving scenarios and show their use in improving policy robustness through additional training. The Adaptive Stress Testing framework used for aircraft collision avoidance systems has also been extended to AVs [22, 9]. Done in simulation, deep RL is used to generate adversarial actor trajectories for an AV system. Wang et al. [42] also generate adversarial scenarios for AV systems by black-box optimization of actor trajectory perturbations. Key to their work is the simulation of LiDAR sensors in perturbed real scenes, allowing adversarial attacks and closed loop testing on a complete LiDAR-based AV system. O'Kelly et al. [31] also propose a closed loop simulated testing framework by importance sampling of safety critical AV scenarios using a learned distribution of driving behavior. We refer to [10] for a detailed survey.

Scene manipulation Ost et al. [32] learn neural scene graphs from real world videos via a factorized neural radiance field [29], while Kar et al. [20], Devaranjan et al. [11] 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. [14] 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. [45] 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. [27] use a differentiable renderer to find adverse lighting and geometry. Their renderer is consequently less capable and images appear stitched, a confounder with regards to the natural distribution. Athalye et al. [4] 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. [46], Tu et al. [37] synthesizes 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 [34, 2] do this by re-weighting or re-sampling the training set, with Chang et al. [8] focusing on detection. Sagawa et al. [36] uses regularization and Wang et al. [43] 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 [17] suggests that these approaches are not better than ERM, which is how we incorporate our found groups in Section 5.

5. Experiments

This section introduces a litany of experiments showing the virtue of our proposed method, as well as support for random interventions being less efficient.

5.1. Datasets

To create the datasets, we first select the CARLA preset map (Town03 or Town05). Then we randomly choose from among the pre-defined weather patterns. We randomly select the camera calibration and the number V of vehicle assets according to the distributions in Nuscenes. We place those V vehicles, the ego agent, and $P = 20$ pedestrian assets, at random town waypoints suitable for the asset type. Finally, we attach the calibrated camera to the ego agent and set every agent to autopilot.

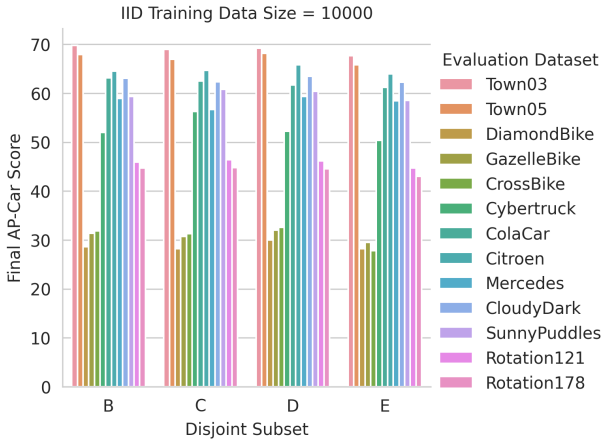

Figure 3: Test results with config 18C4 when training on disjoint IID subsets. Results are consistent, suggesting that the harder groups - bikes, rotations, and cybertruck - are ubiquitously hard.

We wait for 50 timesteps after spawning so the scene stabilizes. We then record for 150 steps and save every 15th frame. We need the 2D ground truth boxes for each asset, but found the suggested approach[3] lacking because it frequently has trouble with occlusions and other challenging scenarios. See the Appendix for heuristics we developed to help filter the ground truth boxes.

5.2. Interventions

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 is an 18C model trained on the base 10000 subset from Town03 and $\tau = 0.2$. We additionally filter the categories to have at least 20 entrants.

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 deleterious groups for the model ac-

[3] See client_bounding_boxes.py in the CARLA repository, commit hash 4c8f4d5f191246802644a62453327f32972bd536.

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 selected interventions, ordered by the percent of times that they were involved in a high magnitude δ event. The 2nd column shows that percent and the 3rd column the total number of observations. Section 5.3 suggests that between 6.0 (Carla Cola) and 6.4 (Cybertruck) is where our cutoff should lie.

cording to this metric. In particular, 'small bikes' have an outsized effect, as does cloudy weather and the rotations where a car is coming at the ego agent or turning to the left. Just after the last bike are two large vehicles, the Cybertruck and the Cola Car. The specificity of the listed weathers and rotations are due to our discretization gifting us an exact value that we translate in the table for semantic understanding. Practically speaking, there is a range of rotation and weather values around the group that would all suffice.

Finally, we do not include any location results in this table because the MLM would frequently re-position the asset somewhere out of the camera's purview. This says more about the asset than it does about the location, and it is also rife with confounders based on what was behind that asset. We could have localized the location interventions more by zeroing out MLM options, but leave that for future work.

5.3. Analysis

After obtaining candidate groups from the designed interventions, we investigate 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 sample $n_{\text{v}} \in [3, 6]$ as the number of vehicles selected from the scene. We then randomly choose vehicles $v_0, v_1, \ldots, v_{n_{\text{v}}}$ in that scene, including vehicles that may actually not be in the camera's purview, and change them to be the target mode. So as to not accidentally introduce a bias

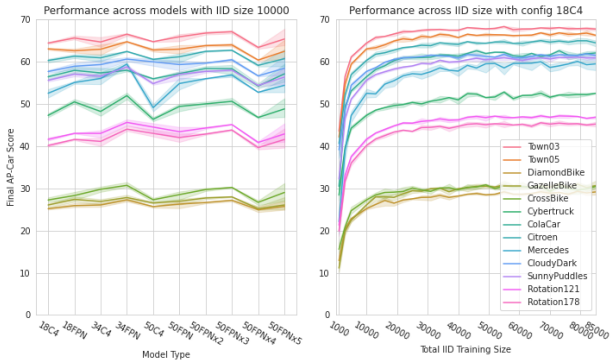

Figure 4: Charts showing independently increasing the model capacity (left) and increasing the data size (right). We see that no model distinguished themselves and that we quickly taper in how effectively the model utilizes the data. We consider the dip in the capacity chart to be an artifact of the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

through the random process, we select the same vehicles v_k for all group datasets. For rotation groups, we choose those same vehicles but rotate them to be the target rotation instead of switching their asset. For weather groups, we change 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 perform best on the IID data from Town03 and just a little bit worse on the same from Town05. Further, they do 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 do 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 of the model's issues, but that we have narrowed them greatly. The model's regression when changing a car to a bike may be because it performs 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 [3]. We restrict our scope to this level of understand-

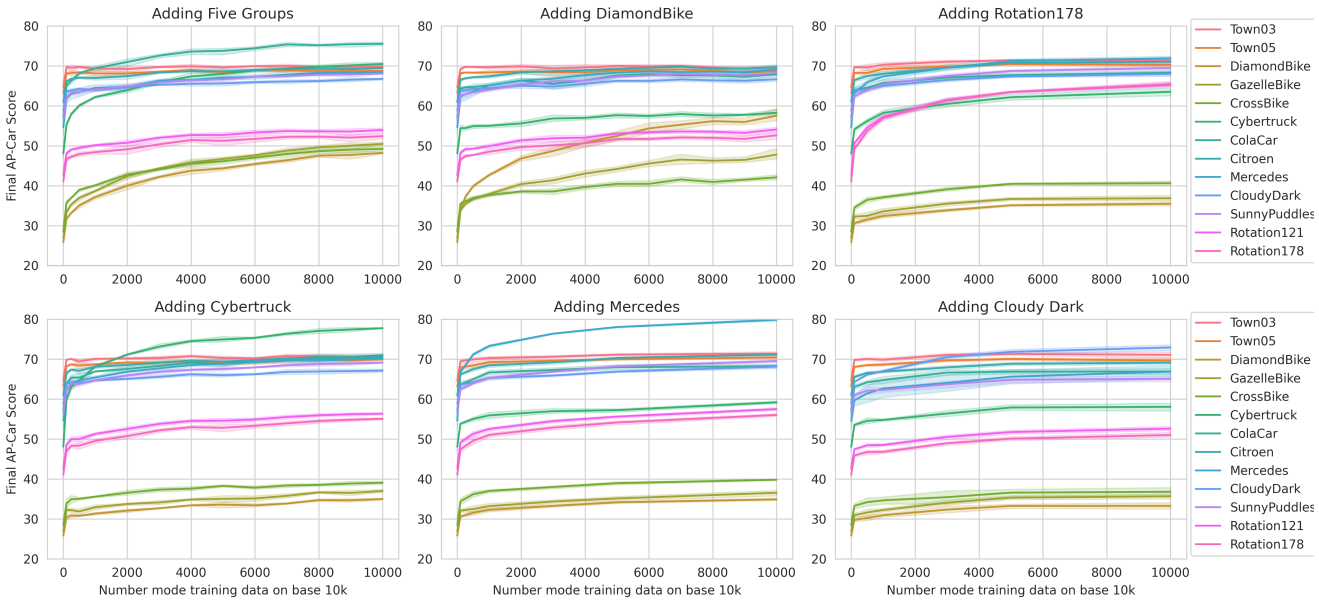

Figure 5: Results of training 18C4 on the base IID 10000 training set plus additional group data. The five groups in the top left (Cybertruck, Cola Car, Diamondback, Gazelle, and Crossbike) are added equally. Observe that, for all charts, adding any one group improves all of the other evaluation scores,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

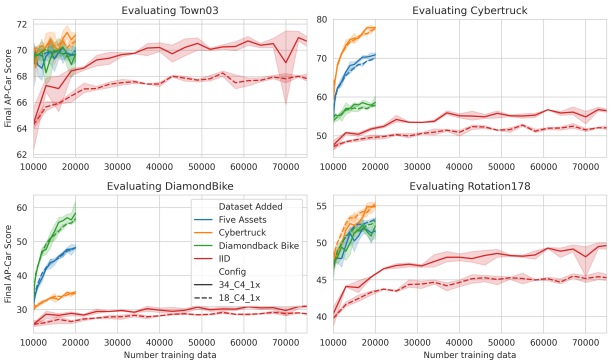

Figure 6: How much IID data is required to match a small amount of extra hard group data. The top left shows 20000 more IID data is required to reach par on IID with 250 group data. The bottom left shows that we never reach the same level on Diamondbacks with IID data as with adding Cybertrucks, let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

ing and leave improvements for future work.

Can we address these issues by increasing capacity? Recent papers [47, 5] 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 using 10000 IID examples.

What if we increased IID data? While we would have to collect data, this would be easier to attain than group specific data. The right side of Figure 4 suggests this too will not be sufficient. We see an initial jump from 1000 to 10000 IID data points, but efficacy slows precipitously across town and group data. We have no reason to think this will suddenly change, and Figure 8 (Appendix) affirms that by suggesting that the percentage of representation of the group is what matters, rather than absolute count.

What if we increased data and capacity simultaneously? Results remain 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 struggle on the harder groups. And the third, seen more clearly in Figure 9, is that more data yields a small accretive effect. This suggests that, 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 add an even amount of each group to the base 10000 IID subset and see that every group improves without impacting the Town03 and Town05 results.

Intervention	MLM	Random	Random 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 types are the same, Random places a) Cybertruck above Kawasaki Bike and b) Carla Cola well ahead of both. Its failure rate for the easy cars is much higher and, in general, posits 3% more failures than MLM.

The other plots in Figure 5 show what happens when we add in training data from any one group M. This predictably improves the model's results on M's validation set. It surprisingly also improves results on *all* of the other groups 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 are more pronounced for like groups - adding data from a biker group (Diamondback, Omafiets, Crossbike) improves the other biker groups a lot more than adding data from the heavy car groups (Cybertruck, Colacar), and similarly for the heavies. Adding rotation groups helps ubiquitously albeit not as much as adding a bike group does for the other bikes. The least effective fix is adding the CloudyDark weather mode. Figure 8 shows that this trend persists 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 change 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. One immediate observation is that the assets are ordered incorrectly with CarlaCola higher than both Cybertruck and Kawasaki Bike. Another observation is that Random has a higher percent of high threshold events. This holds 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 corresponds to sampling from a data distribution that does not faithfully represent the original training distribution.

Figure 11 (Appendix) shows density plots for rotation and cloudiness interventions, conditioned on the intervention being deleterious. We use density plots to demonstrate the differences between Random and MLM because these interventions are continuous for Random. For rotation, we see that there is a mostly steady plateau for Random while MLM shows a clear single group aligned with the bi-modal humps in Original. For weather, we see that Original and MLM are almost overlapping and, while Random is similarly bi-modal, its shape is less pronounced and more even as expected. These both further 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.

Qualitatively, why do these groups exist? With groups in hand, it is now easy to ascertain why our models were failing. For the bikes, it is because they are underrepresented in Nuscenes. For Rotation121, the model infrequently trains on turning cars due to the town layout. For Rotation178, the model infrequently trains on cars facing it due to the traffic policy and the quantity of cars. The Cybertruck is challenging because it is very large and causes occlusion issues in labeling. The ColaCar also has this issue, just not as severely as the Cybertruck. These issues can only be hypothesized without the groups in hand.

6. Conclusion

Combining causal interventions, MLMs, and an AV simulator, we presented a novel method that finds challenging groups for a detection model in foresight by having the MLM resample scene constituents. This direction is promising towards making more robust AV systems and inspiring confidence in downstream users.

Our method has limitations. Although we cannot apply it to real world data because we do not have fine control over scenes, Ost et al. [32] is a step towards overcoming this concern. Until then, the sim2real gap [35, 19] is everpresent. Another limitation is that our method only works on first-order interventions because otherwise the samples drift from the data distribution. However, taking multiple node changes 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.

References

[1] 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

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

[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020. 6

[4] 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. 5

[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

[6] 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. 3

[7] 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. 4

[8] 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

[9] 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

[10] 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

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

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 1

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[14] 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

[15] 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. 11

[16] 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

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

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[19] 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. 8

[20] 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

[21] 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. 2

[22] 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

[23] 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. 4

[24] 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. 4

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 2

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017. 2

[27] 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

[28] 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. 3, 4

[29] 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

[30] 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

[31] 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

[32] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. 2020. 5, 8

[33] 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. 3

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

[35] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image, 2017. 8

[36] 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

[37] 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

[38] 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. 3

[39] 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

[40] 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. 4

[41] 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. 4

[42] 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)*, 2021. 4

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

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

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

[46] 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

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