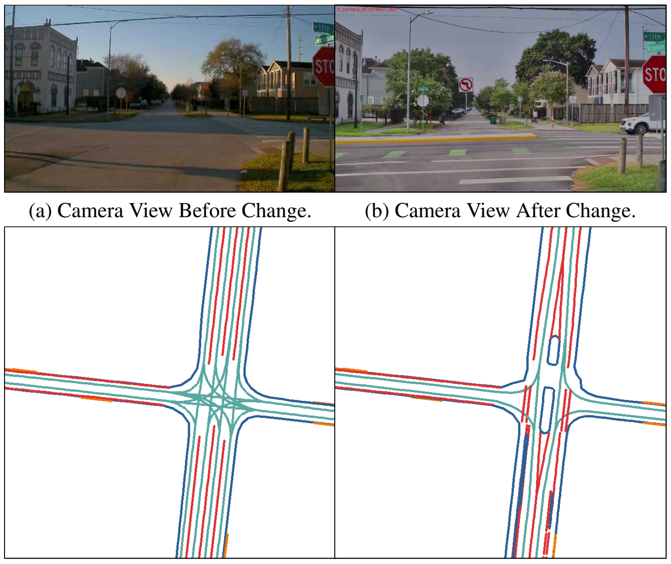
Exploring Real World Map Change Generalization of Prior-Informed HD Map Prediction Models

Samuel M. Bateman*†, Ning Xu*, H. Charles Zhao*, Yael Ben Shalom,
Vince Gong, Greg Long, Will Maddern
Nuro, Inc.
{sbateman, nxu, czhao, ybenshalom, vgong, glong, will}@nuro.ai

Abstract

Building and maintaining High-Definition (HD) maps represents a large barrier to autonomous vehicle deployment. This, along with advances in modern online map detection models, has sparked renewed interest in the online mapping problem. However, effectively predicting online maps at a high enough quality to enable safe, driverless deployments remains a significant challenge. Recent work on these models proposes training robust online mapping systems using low quality map priors with synthetic perturbations in an attempt to simulate out-of-date HD map priors. In this paper, we investigate how models trained on these synthetically perturbed map priors generalize to performance on deployment-scale, real world map changes. We present a large-scale experimental study to determine which synthetic perturbations are most useful in generalizing to real world HD map changes, evaluated using multiple years of real-world autonomous driving data. We show there is still a substantial sim2real gap between synthetic prior perturbations and observed real-world changes, which limits the utility of current prior-informed HD map prediction models.



(a) Camera View Before Change. (b) Camera View After Change.

(c) Example Map Before Change (d) Example Map After Change

Figure 1. A real-world map change from an autonomous vehicle dataset. In this paper we investigate which synthetic perturbations applied to a simulated prior map at training time best model a real prior map (left) for training a prior-informed online mapping model to produce the updated map (right), evaluated using a vast collection of real-world changes gathered over multiple years of autonomous vehicle operation.

1. Introduction

Large scale autonomous driving has long been a milestone for the robotics community and has been pursued for well over a decade now. Perception and mapping systems have formed core components of self driving systems, and mobile robots more generally, enabling comprehension and understanding of the world around them [10, 39]. In the field of mobile robotics, the goal of mapping systems at a high level is to predict semi-static geometry and affordances in a scene, i.e. elements of the world that rarely change over time. These semi-static elements are traditionally encoded with human labeled "High Definition" (HD) maps built atop a high resolution geometric world reconstruction utilizing many overlapping trajectories. Reducing this cost within the context of self driving would increase scalability, cost effectiveness, and robust safety. However, accurately and consistently constructing online HD maps from sensor data at sufficiently long ranges to facilitate safe, fully autonomous driving has proven challenging (outside of highways and simple road scenarios which have limited topological or geometric complexity).

∗ Equal Contribution.
† Corresponding Author.

Despite this, in the last few years, there has been a resurgence of interest in this problem. Equipped with modern machine learning techniques [7, 30], architectures [6, 8, 45], and open datasets [5, 47], the field has begun to pivot towards this online mapping approach as performance on other previously out of reach perception tasks has become more compelling. While exciting, the challenge of predicting highly accurate HD maps at suitable ranges remains, and the majority of recently published research on online mapping degrades in prediction performance with range from the vehicle.

One response to this problem, and one explored by concurrent work in [43], is integrating offline, potentially out of date or inaccurate HD Map priors into the prediction of online features. This is particularly compelling because low quality, out of date HD maps or sparse, low resolution Standard Definition (SD) maps are often available or cheaply labeled, and maps of roads are generally slow to change over time. The result would be a model that only rarely has to perform full online mapping, but most of the time is acting to clean up the discrepancies between the prior and reality.

A major caveat is that real world examples of meaningful map changes are relatively rare, even from the point of view of large, industrial deployments [15]. To step around this issue, MapEX [43] instead proposes that one could use synthetic mutations of labels to imitate these map changes over time. This allows one to generate a virtually limitless number of map changes to train their model to "fix" the prior, effectively training a change detection/map repair model rather than a full online mapping model. While this almost certainly would not be trained on the same noise distribution as real changes in the world, one would hope that sufficiently diverse perturbations of the prior map in training would minimize the sim2real transfer gap by acting akin to domain randomization [44], where the real distribution is in the support of the synthetic distribution applied to the labels. This alternative problem formulation where we have access to a prior could, in theory, significantly simplify the problem for perception systems, and has empirically been shown to outperform existing online-only models [43]. However, the broad applicability of such methods is predicated on its ability to effectively transfer from synthetic offline perturbation to real world map change events.

In this paper, we aim to answer the following two questions:

- Does training prior-informed online mapping models on synthetic prior mutations [43] generalize to real world map change examples?
- Considering a broad range of prior noise models, how do mixtures of prior noise models applied in training affect the generalization performance of these online mapping models to real world map changes?

In addition, we share details of how our model architecture differs from existing online mapping models in the literature to aid in reproducibility of our work.

2. Related Work

2.1. Birds Eye View Perception

3D perception is a core problem in mobile robotics and computer vision more broadly. The long term trend of 3D perception has been to leverage large, expressive backbones [8, 9, 18, 26–28, 41] trained on very large image datasets [5, 22, 40, 42, 47], to feed high quality image representations into various heads for specific tasks, such as image classification [13], object detection [6, 10, 24, 39], semantic segmentation [2, 48], keypoint estimation [10], and more. This trend has only strengthened with the rise of very large models pretrained on largely unsupervised objectives with internet scale data both within computer vision [32] and outside of it [4].

One direction that has advanced considerably in the last few years is the representation space used for 3D perception. Early 3D object detection approaches focused on a couple key approaches: one being detecting and tracking in 2D image space and reprojecting model outputs to 3D space using geometric information [33], and the other being early fusion through methods like [46]. With the rise of the use of LiDAR in mobile robotics perception tasks, efforts were made to develop better data representations and encoders for 3D object detection with LiDAR [16, 37, 38, 51]. Because of the natural 3D geometry of LiDAR data and the approximately 2.5D worlds that mobile ground robots generally perceive (i.e., generally much fewer detections coinciding along the z direction than in the x and y directions), the aforementioned publications cumulatively proposed laying out LiDAR information as features in a voxel grid represented as a top down feature image. This topdown representation, known as the Birds Eye View (BEV) representation, has attracted a large amount of attention in perception for mobile robotics over the past few years. In particular, a huge number of contributions have sought to develop BEV representations of other sensors as well so that they can be simply merged into a single unified feature representation. For example, much of this research focus has been on developing expressive BEV feature backbones utilizing both camera and LiDAR data [18, 28, 36, 50], which then can be consumed by downstream tasks through a relatively simple, single BEV image interface.

2.2. Online HD Map Construction

One specific downstream task which has been explored on top of these BEV models for the task of autonomous driving is online mapping. Early efforts in online mapping primarily focused on doing semantic segmentation from the perspective view of a camera [49], or as a semantic segmentation problem utilizing a BEV representation [17]. However, these methods struggle with two key problems: real world roads often have complex topologies and instance-level traits which are difficult to represent accurately with semantic segmentation, and many downstream behavior planning models (e.g. [31], [11]) consume vectorized map representations. Indeed, part of the reason why HD maps are generally labeled as vectorized features is that this removes much of the ambiguity regarding topology and overlapping elements and affordances of roads. Thus, it would be ideal to directly predict vectorized features. Earlier work identified this requirement, and reframed the polyline prediction problem in a few different ways, i.e. autoregressive transformer approaches [25] or classical, heuristic post processing of semantic segmentation [17]. A recent line of work [19, 20] similarly reframed the problem again by making the connection that vectorized online mapping can be represented as an unordered set detection problem, with a similar problem structure as the one-to-one bijective object detection transformer described in [6], and reported compelling empirical performance with this approach. This has sparked a flurry of renewed interest in the online mapping problem [20, 21, 29]. The work most related to ours is the concurrent work of MapEX [43] which proposes to add low quality HD map priors as inputs to MapTR [19] to improve detection performance despite occlusion or changes in the underlying map, attempting to solve the well known map change problem. This provides an alternative avenue for solving the change detection problem: rather than maintaining an up to date map, one can instead train a model to leverage both sensor data and out of date HD map data to reconstruct an accurate, live representation of the world around it, enabling lifelong deployment of mobile robots with significantly lower HD map maintenance expense.

3. Methodology

3.1. Map Detection Head

To predict vectorized map features, we follow in the footsteps of MapTR [19, 20] with a Deformable DETR [52] based polyline detection model, with one query per control point where this query is the sum of a per polyline embedding and a per point embedding as in [19]. In the following sections, we will primarily describe places where our methodology diverges from existing literature to aid in reproducibility of our results.

3.2. Single Stage Bipartite Matching Loss

We forgo the two-stage hierarchical matching loss from MapTR [19] and instead utilize a simpler one stage bipartite loss by computing a pairwise loss matrix before computing the bipartite matching itself. To do this, we first compute a set of loss tensors to model the positional error and the permutation symmetries introduced in [19]. We will use an indexed matrix notation to define the constructed tensors used to compute the losses, where $\hat{x} \in \mathbb{R}^{m_{pred} \times n_{points} \times p_{dim}}$ is the prediction from the decoder transformer, where m_{pred} is the maximum number of predictions, n_{points} is the number of control points in every polyline, and p_{dim} is the dimensionality of each control point. Similarly, $x \in \mathbb{R}^{m_{gt} \times n_{points} \times p_{dim}}$ is the ground truth polyline tensor padded with no-object classes to always have m_{gt} objects to match to, where m_{gt} is the max number of ground truth labels for a given frame. Since m_{pred} is fixed, we set $m_{pred} = m_{gt}$. Following this, we construct our loss matrices in the general form

$$\mathcal{L}_{p2p_c} = \Big[I_c(j) \min_{P_a \in Q_c} \sum_{k,l} |P_a \cdot \hat{x}_{i,k,l} - x_{j,k,l}| \Big]_{i,j} \tag{1}$$

where

$$Q_c = \{P_a | P_a \in \mathbb{M}^{m_{pred} \times m_{pred}}, \\ P_a \in \text{valid permutations for class c}\} \tag{2}$$

is the set of valid permutation matrices for a given invariance class $c \in$ {polygon, undirected polyline, directed polyline} and

$$I_c(j) = \begin{cases} 1 & \text{Invariance Class}(j) = c \\ 0 & otherwise \end{cases} \tag{3}$$

is a masking function which sets the jth ground truth label loss row to all 0 if the ground truth loss is not of that invariance class. As in [19], the valid permutations of polygons represent the set of all shift permutation maps with the polyline indexed in both directions (clockwise and counter-clockwise). Similarly, the valid set of permutations for undirected polylines is only swapping directions of the polyline and is simply the identity permutation for directed polylines. Using an actual permutation matrix would be computationally expensive, but these can be implemented inexpensively using `tf.roll`, `tf.reverse`, and `tf.tile` operations in Tensorflow [1] or similar operations available in most any comparable differentiable array computation library [3, 34]. Note that we optionally also add a scaled pairwise cosine similarity loss from [19] to this as well, though our experience suggests that weighting cosine similarity much lower than the pointwise positional loss helps with convergence when using this single step training objective.

With these 3 loss matrices, we can construct a single stage point2point matrix loss as:

$$\mathcal{L}_{p2p}(\hat{x}, x) = \mathcal{L}_{p2p_u}(\hat{x}, x) + \mathcal{L}_{p2p_d}(\hat{x}, x) + \mathcal{L}_{p2p_p}(\hat{x}, x) \tag{4}$$

where $\mathcal{L}_{p2p_u}$, $\mathcal{L}_{p2p_d}$, and $\mathcal{L}_{p2p_p}$ refers to the resulting pairwise loss matricies for each invariance class, those being

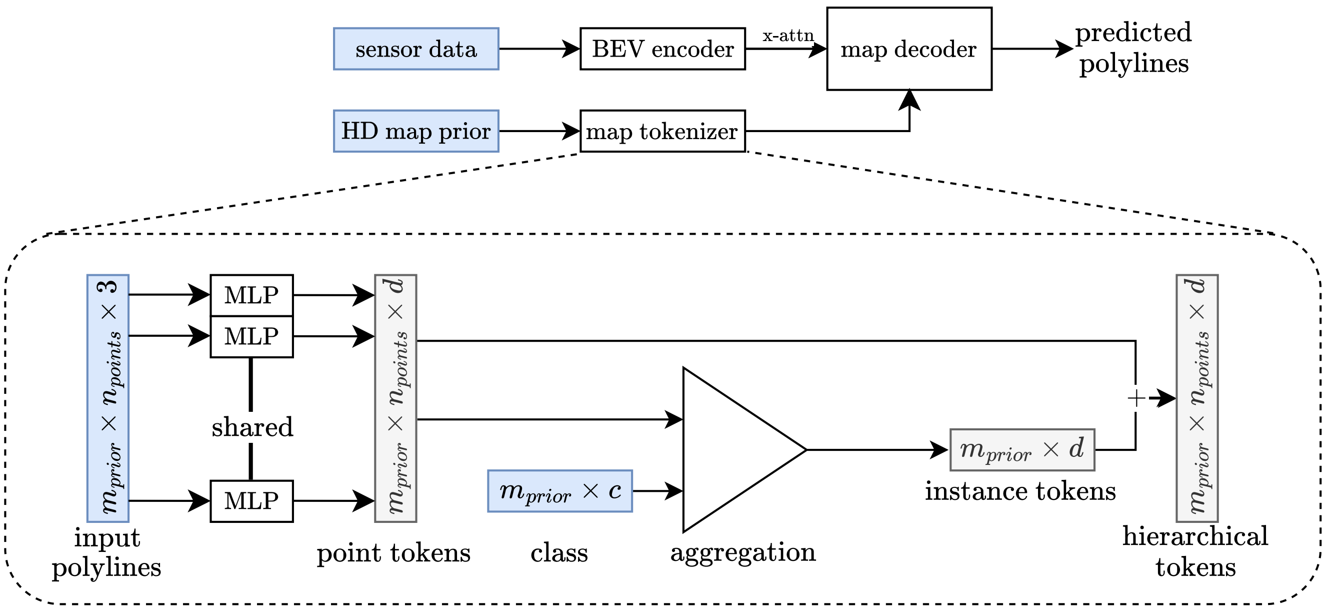


Figure 2. Overall architecture of our model, heavily influenced by [19, 43]. See these references for more details on implementation.

undirected polylines, directed polylines, and polygons respectively. Note that the resulting per batch polyline localization error matrix includes the pairwise positional error between every ground truth and predicted polyline, where each pairwise positional error is computed with the loss minimizing prediction permutation which is valid for a given pair's label. This matrix is identically 0 for any no-object ground truth class row, and correctly masks invalid permutations from matching with any given label as defined by the label's invariance class. We can then utilize a pairwise focal classification loss [23] matrix $\mathcal{L}_{focal}$ of a similar construction and sum them into a single loss matrix

$$\mathcal{L}_{pairwise}(\hat{x}, x) = w_c \cdot \mathcal{L}_{focal}(\hat{x}, x) + w_p \cdot \mathcal{L}_{point2point}(\hat{x}, x) \tag{5}$$

where $w_c, w_p \in \mathbb{R}$ are weighting terms to each respective loss type. With this full pairwise loss matrix, we can solve for the minimum loss label/prediction assignment using the Hungarian Algorithm [14] and directly sum up the resulting optimally matched losses rather than doing a hierarchical matching as in [19]. This should result in a similar final cost function, but performing Hungarian matching directly on the final losses ensures there is no divergence of objectives between point-wise and polyline-wise convergence and simplifies training code.

3.3. Map Tokenizer and Prior Integration

To incorporate an HD map prior, we use a map tokenizer similar to [43]. The map tokenizer is a lightweight learned module that converts an unordered set of polylines into tokens, the language of transformers (see Fig. 2). We want these tokens to encode as much useful information from the prior as possible, including both point-level and polyline-level information. We take inspiration from the idea of hierarchical queries in MapTR [19], and we set the tokens equal to the sum of a point token and an aggregated polyline token derived from the point tokens. The point tokens are generated using a Multi-Layer Perceptron (MLP) over the point coordinates, where this MLP is shared across all points. The polyline tokens are generated by max pooling over point tokens, concatenating this with a one-hot class vector, and passing through another MLP which is shared across all polylines. This weight-sharing scheme preserves permutation invariance among polylines, and the max pooling is a lightweight way to aggregate information.

One important case to handle is when $m_{prior} < m_{pred}$, where m_{prior} is the number of prior polylines, which is almost always the case. If we just naively add padding to the end of the prior up to m_{pred}, then some of the positional encodings in the transformer decoder will always be associated with a prior while some will almost never be associated with a prior, which could cause undesirable biases. To mitigate this, we shuffle the prior tokens after adding padding so that padding is effectively inserted randomly.

Once we have converted the prior into tokens, we still need a way to consume these tokens. One approach is to add an extra cross-attention step to the decoder layer that attends to these tokens. In this formulation, the prior is simply another modality to attend to, in addition to the BEV embedding. However, we found that this approach failed basic overfitting experiments. We suspect that cross-attention does not provide enough bandwidth for the model to incorporate the prior as strongly as it should, especially with a limited number of decoder layers.

Another approach is to directly replace the fixed hierarchical queries in MapTR [19] with these prior tokens. This formulation has a nice intuitive interpretation—the prior tokens provide the initial estimates, which are then refined through several decoder layers by attending to the BEV embedding to come up with a posterior. We found that, when tested against synthetic map perturbations, this approach works better and has more stable training, which is consistent with the approach in [43], and thus is the approach used in all of our results.

3.4. Types of Perturbations

We implement a number of synthetic map prior perturbation types, expanding on the experiments from [43]. These can be roughly classified as *discrete mutations* which change the number or types of polyline features, or *continuous, warping mutations* which change the position or shape of the features (see Fig. 3). Ultimately the goal of all these mutations is to prevent the trained model from simply passing through the vectorized polyline prior features while completely ignoring sensor observations.

3.4.1 Discrete Mutations

The primary goal of discrete mutations is to capture map changes to a scene which cannot be fully characterized simply by warping the geometries of the underlying features, and are more similar to a discete Bernoulli distribution of something about a map feature that has or hasn't changed, such as the class of a feature or the number of features in a scene.

- *Feature dropout* (Fig. 3b) – We use a full feature dropout mutation to model large scene changes or incomplete labels. We drop out each feature in the scene with a Bernoulli distribution with equal probability across each feature.
- *Feature duplication* (Fig. 3c) – We use a feature duplication mutation to model accidental label duplication, as well as to model large scene changes when mixed with warping perturbations which will warp each duplicated feature in a different way. This mutation works by again using a Bernoulli trial for each existing polyline feature, and truncating the newly added features to a maximum of m_{pred} features.
- *Wrong class* (Fig. 3d) – We use a wrong class mutation which, again by Bernoulli trial, mutates each polyline's class with some probability to a random other class. This has similar goals as previous mutations of decreasing model reliance on the prior, but is the only mutation which corrupts class information, modeling mislabeled features as well as map change events such as repainting of lines or the addition of new lanes.

3.4.2 Continuous, Warping Mutations

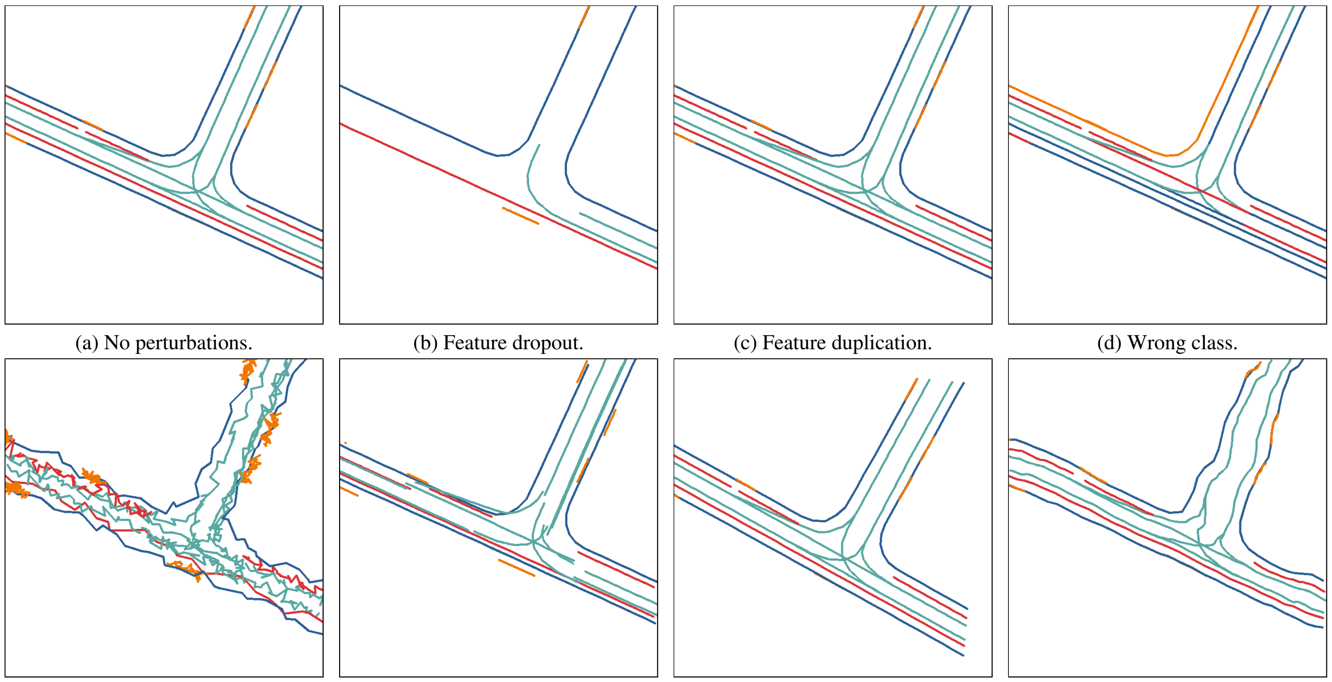
To complement our discrete mutations of category and cardinality, we also introduce a number of continuous, geometry warping mutations. Note that each of these are parameterized by a standard deviation which scales the variance of the resulting noise.

- *Control point perturbation* (Fig. 3e) – This is a simple per-control-point zero-mean Gaussian shift to help ensure the model cannot simply pass through the prior as an identity function.
- *Feature location perturbation* (Fig. 3f) – We hypothesize that the model may relatively easily overcome control point perturbation by simply smoothing out the prior rather than attending to sensor data, so we also use a zero-mean Gaussian per-feature location perturbation.
- *Global rotation and shift* (Fig. 3g) – We apply Gaussian perturbations to global yaw and position to simulate robot localization errors. We also hypothesize that even per-feature location perturbation may be relatively easily overcome by the model through self-attention without attending to sensor data, but this localization mutation can only be inverted by leveraging sensor data.
- *Perlin warp* (Fig. 3h) – We generate two 2D Perlin noise [35] images utilizing fractional Brownian motion [12], one each corresponding to warping x and y coordinates, and then renormalize the resulting noise distribution to be zero mean. We then sample from this image at the coordinates of the control points of the polylines to get structured noise. We hypothesize that such correlated warping will make it harder for the model to learn to simply denoise the zero-mean noise added to individual features, better simulating how curbs, lanes, and lane boundaries may all be moved together during a major, real map change event.

4. Experiments

4.1. Real-World Change Examples

To generate a large set of real world map change examples, we leverage an internal map database to generate a region change diff between a 2020 version of our internal HD map, and that same map from 2023. We only include changes to the map significant enough that they required a recollect of data in the area and subsequent changes to the HD semantic map. We then mine for 30 second scenes collected in the second half of 2023 which intersect these regions of change and occur after all computed map changes. To dump data for these scenes, we obtain the prior from the 2020 map version and the ground truth labels from the map version at data



(a) No perturbations. (b) Feature dropout. (c) Feature duplication. (d) Wrong class.

(e) Control point perturbation. (f) Feature location perturbation. (g) Global rotation and shift. (h) Perlin noise.

Figure 3. Types of perturbations. Note that Fig. 3c looks the same as Fig. 3a since features are exactly duplicated.

collection time in 2023, something not available from public change detection datasets [15] to the best of our knowledge. With this approach, every scene mined contains some change in the HD map of varying severity.

4.2. Experiment Setup

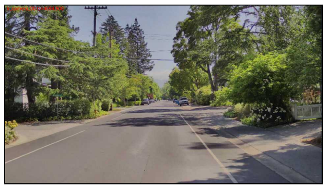
We perform all experiments with a BEV backbone similar to [28] with pre-trained LiDAR and camera BEV feature extractors which are trained for general perception tasks. We train each model for 75K steps on 32x Nvidia A100s on a large internal dataset collected from Houston, TX and Mountain View, CA (>13.7K scenes, >822K unique training frames), with a 90m square field of view centered on the vehicle and four different feature types: lane centers, lane dividers, road boundaries, and driveways. For evaluation, we utilize two major types of datasets, a synthetic evaluation set and a real world change dataset. First, for the synthetic evaluation set, we use a geographically split holdout test dataset (>3.3K scenes, 198K unique test frames) with low levels of synthetic prior noise utilizing all presented mutations (0.1m Std. Dev. for continuous warping mutations, and 0.1 probability for discrete mutations). Then, for real world change data, we mine a holdout test set (1240 scenes, 74k unique test frames) of scenes which have undergone real world HD map change, and provide an outdated map prior from multiple years of aggregated map changes in the geospatial map database as described in Sec. 4.1.

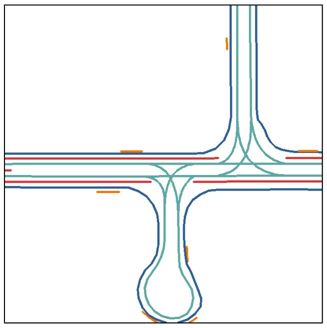
The value of utilizing our internal dataset is multifold. For one, it significantly reduces the risk of geospatial overfitting of map detection transformers [21] by holding out large geospatial regions for eval exclusively during the training and evaluation split. In addition, our dataset is significantly larger than the majority of open datasets (e.g. [15]), and is thus able to leverage the scaling behavior of map prediction models identified in [21] to further mitigate the noise caused by overfitting. Finally, we provide real world pairs of outdated and up to date maps, something not available in [15] as pointed out in [43]. As for metrics, we compute mean Average Precision (mAP) of predicted polylines using the same Chamfer distance based metrics and thresholds used in [19].

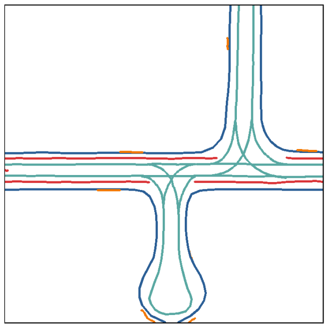
4.3. Experimental Design

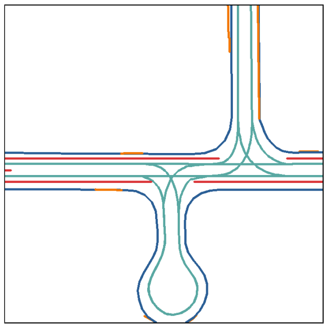
Due to the combinatorial hyperparameter space induced by so many different parameters, we first train a baseline model with a small amount of noise for each prior mutation, then tune each mutation parameter independently. Qualitative results for the baseline low noise model are shown in Fig. 4.

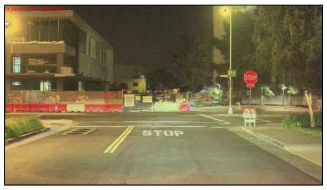
For the parameter search (Tab. 1, Tab. 2), we start with all mutations except Perlin warp enabled with a low noise level ("Low All Noise" in the table), where this low noise level is identical to the synthetic perturbation distribution (0.1m Std. Dev for continuous mutations, 0.1 probability of discrete mutations). We then test a number of increased levels of noise for their performance against the synthetic evaluation dataset as well as the real world evaluation dataset.

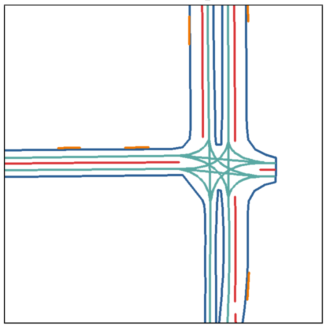

(a) New Driveway Geometry Scene

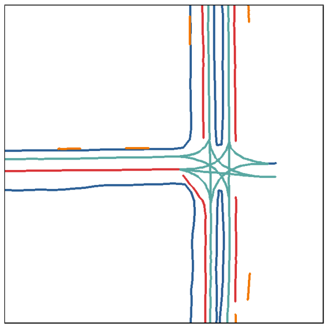

(b) Outdated Map Prior

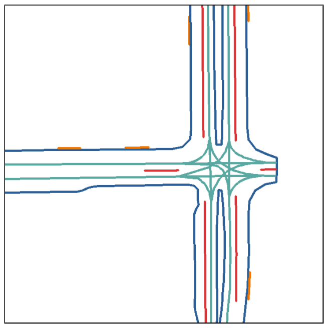

(c) Prior-Informed Prediction

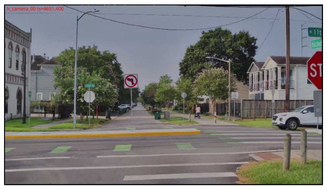

(d) Ground Truth

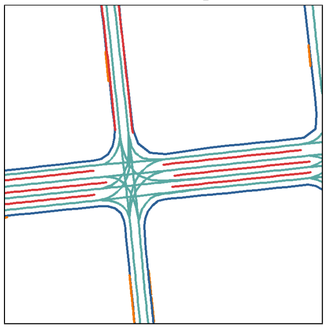

(e) New Curb Geometry Scene

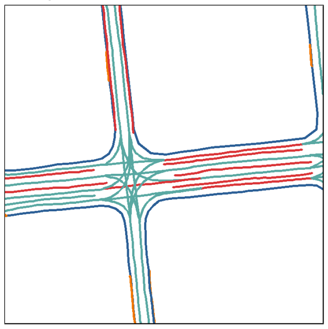

(f) Outdated Map Prior

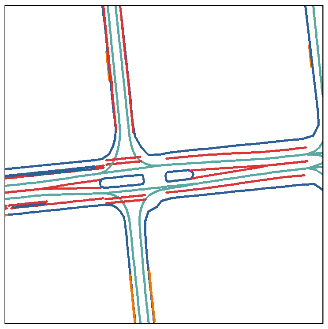

(g) Prior-Informed Prediction

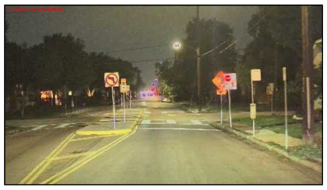

(h) Ground Truth

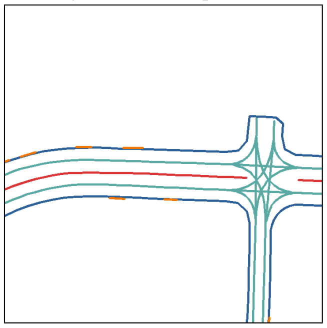

(i) New Median Intersection Scene

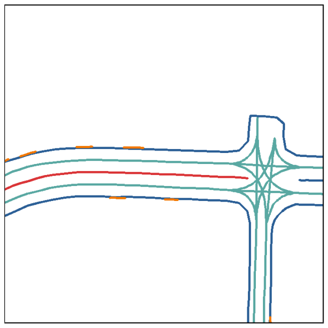

(j) Outdated Map Prior


(k) Prior-Informed Prediction


(l) Ground Truth


(m) New Road Construction Scene


(n) Outdated Map Prior


(o) Prior-Informed Prediction

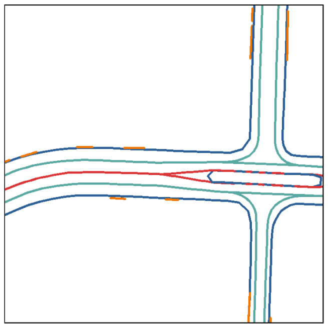

(p) Ground Truth

Figure 4. Qualitative results handling real world changes. For minor real-world changes, e.g. driveway geometry (a–d) and curb geometry (e–h), a model trained with prior perturbations correctly predicts changes to many of the real-world features. However, for substantial changes in road layout, e.g. additional medians (i–l) and new road construction (m–p), the model fails to meaningfully deviate from the prior to account for the new intersection geometry. Note that the topdown map is centered on the vehicle in all figures.

Mutation	**Probability**	**mAP**	
		Sim	Real
Baseline (No Noise)	0.0	0.8980	0.8239
Low All Noise	0.1	**0.9934**	**0.8571**
Drop Features	0.2	**0.9917**	**0.8564**
	0.4	0.9834	0.8435
	0.6	0.9677	0.8313
	0.8	0.6012	0.6074
	1.0	0.6580	0.6952
Duplicate Features	0.2	**0.9920**	0.8535
	0.3	0.9892	0.8570
	0.5	0.9917	**0.8604**
Wrong Class	0.2	0.9929	**0.8564**
	0.3	**0.9932**	0.8555
	0.5	0.9901	0.8542

Table 1. Discrete Mutation Parameter Search.

Mutation	**Std. Dev**	**mAP**	
		Sim	Real
Baseline (No Noise)	0.0 m	0.8980	0.8239
Low All Noise	0.1 m	**0.9934**	**0.8571**
Perturb Control Points	0.5 m	**0.9929**	**0.8640**
	1.0 m	0.9843	0.8512
	2.0 m	0.9531	0.8476
Shift Features	0.5 m	**0.9794**	**0.8612**
	1.0 m	0.9373	0.8461
	2.0 m	0.8781	0.8108
Localization Noise	0.5 m, 0.5°	**0.9936**	**0.8648**
	1.0 m, 1.0°	0.9913	0.8588
	2.0 m, 2.0°	0.9911	0.8604
Perlin Warp	0.5 m	**0.9854**	0.8606
	1.0 m	0.9711	**0.8623**
	2.0 m	0.9357	0.8407

Table 2. Continuous Warp Mutation Parameter Search.

5. Discussion

We note a number of interesting observations from our experimental results. First, consistent with [43], we note that training with no prior noise at all results in a "pass-through" model which learns to replicate the prior map without modifications. Since changed map features usually comprise a small percentage of the features in any given frame of data, we see that this pass-through model which is ignoring the sensor BEV information is capable of achieving a 0.824 mAP on the real world map changes regardless.

More substantially, we note a consistent sim2real gap between the simulated, low noise evaluation prior and the real world map change dataset, where the model has learned to effectively denoise the evaluation prior but is not sufficiently general to smoothly transfer to real change detection. Error is correlated between the simulated and real prior corruption, but our simulated evaluation is insufficient to model the complexities of real world changes. The qualitative results in Fig. 4 reinforce this observation, where only the simplest real-world changes are accurately predicted by the model, which reverts to the prior when the changes become too complex.

Somewhat surprising is that increasing prior dropout noise primarily serves to degrade model performance on real world map changes as its noise is increased, rather than cleanly trading off between real world sensor and prior contributions. Instead, we see that performance slowly degrades, until a bifurcation in response behavior when having too degraded of a prior causes the model to perform even worse than it does when trained with no prior and then tested with prior (Drop Features, p=1.0), which is completely out of distribution. Similarly interesting is that increased likelihood of feature duplication is somewhat more helpful than any other discrete mutation, with performance increasing with increased mutation likelihood for the values we tested on.

We see a slightly different story with continuous warping mutations, which all improve on real world performance with an increased level of noise from the baseline low noise level. Past that, however, they also see degradation behavior as perturbations get more exaggerated at higher noise levels, similar to that of the discrete features. This is likely due to the true distribution of map changes having a similar level of average displacement, where higher noise levels are unreasonable in nominal real world change scenarios (e.g. redoing a curb or driveway).

6. Conclusions

Robustness to real-world changes is critical for any map-based autonomous vehicle system. We confirm the conclusions of [43] in that prior maps are much better than no prior and that we need some noise in the prior to learn something more useful than a pass-through function.

However, we are able to expand on those results by observing that too corrupted or weak of a prior can actually harm performance of the model more than omitting the prior entirely. Most importantly, we show that there exists a considerable sim2real gap between real world change detection performance and performance on simulated prior noise. We observe through large-scale experiments that prior mutations are sufficient to capture only the simplest of real-world changes. We hope the results presented in this paper motivate future work in this area to address the sim2real gap for HD map prediction with prior.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016. arXiv:1605.08695 [cs]. 3

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. In *CVPR*, 2015. arXiv:1511.00561 [cs]. 2

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 3

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *NeurIPS*, 2020. arXiv:2005.14165 [cs]. 2

[5] 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, 2020. arXiv:1903.11027 [cs, stat]. 2

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020. arXiv:2005.12872 [cs]. 2, 3

[7] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020. arXiv:1909.13719 [cs]. 2

[8] 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. In *ICLR*. arXiv, 2021. arXiv:2010.11929 [cs]. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2015. arXiv:1512.03385 [cs]. 2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. arXiv:1703.06870 [cs]. 1, 2

[11] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. VAD: Vectorized Scene Representation for Efficient Autonomous Driving. In *ICCV*, 2023. arXiv:2303.12077 [cs]. 3

[12] Kenneth Falconer. *Fractal Geometry: Mathematical Foundations and Applications, 3rd Edition*. 2014. 5

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, 2012. 2

[14] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 4

[15] John Lambert and James Hays. Trust, but Verify: Cross-Modality Fusion for HD Map Change Detection. In *NeurIPS*, 2021. arXiv:2212.07312 [cs]. 2, 6

[16] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *CVPR*, 2019. arXiv:1812.05784 [cs, stat]. 2

[17] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. HDMapNet: An Online HD Map Construction and Evaluation Framework. In *ICRA*, 2022. arXiv:2107.06307 [cs]. 3

[18] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers. In *ECCV*. arXiv, 2022. arXiv:2203.17270 [cs]. 2

[19] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. MapTR: Structured Modeling and Learning for Online Vectorized HD Map Construction. In *ICLR*, 2023. arXiv:2208.14437 [cs]. 3, 4, 5, 6

[20] Bencheng Liao, Shaoyu Chen, Yunchi Zhang, Bo Jiang, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. MapTRv2: An End-to-End Framework for Online Vectorized HD Map Construction, 2023. arXiv:2308.05736 [cs]. 3

[21] Adam Lilja, Junsheng Fu, Erik Stenborg, and Lars Hammarstrand. Localization Is All You Evaluate: Data Leakage in Online Mapping Datasets and How to Fix It, 2023. arXiv:2312.06420 [cs]. 3, 6

[22] 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. In *ECCV*, 2014. arXiv:1405.0312 [cs]. 2

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 4

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In *ECCV*, pages 21–37, 2016. arXiv:1512.02325 [cs]. 2

[25] Yicheng Liu, Tianyuan Yuan, Yue Wang, Yilun Wang, and Hang Zhao. VectorMapNet: End-to-end Vectorized HD Map Learning. In *ICML*, 2023. arXiv:2206.08920 [cs]. 3

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*, 2021. arXiv:2103.14030 [cs]. 2

[27] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. arXiv, 2022. arXiv:2201.03545 [cs].

[28] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. In *IEEE International Conference on Robotics and Automation (ICRA)*. arXiv, 2022. arXiv:2205.13542 [cs]. 2, 6

[29] Zihao Liu, Xiaoyu Zhang, Guangwei Liu, Ji Zhao, and Ningyi Xu. Leveraging Enhanced Queries of Point Sets for Vectorized Map Construction, 2024. arXiv:2402.17430 [cs]. 3

[30] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. arXiv:1711.05101 [cs, math]. 2

[31] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023. arXiv:2207.05844 [cs]. 3

[32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, 2024. arXiv:2304.07193 [cs]. 2

[33] Su Pang, Daniel Morris, and Hayder Radha. CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. 2020. arXiv:2009.00784 [cs]. 2

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. arXiv:1912.01703 [cs, stat]. 3

[35] Ken Perlin. Improving noise. *ACM Transactions on Graphics*, 21(3):681–682, 2002. 5

[36] Jonah Philion and Sanja Fidler. Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D. In *ECCV*, 2020. arXiv:2008.05711 [cs]. 2

[37] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. arXiv:1612.00593 [cs]. 2

[38] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *CVPR*, 2018. arXiv:1711.08488 [cs]. 2

[39] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*, 2016. arXiv:1506.02640 [cs]. 1, 2

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*, 2015. arXiv:1409.0575 [cs]. 2

[41] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015. arXiv:1409.1556 [cs]. 2

[42] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *CVPR*, 2020. arXiv:1912.04838 [cs, stat]. 2

[43] Rémy Sun, Li Yang, Diane Lingrand, and Frédéric Precioso. Mind the map! Accounting for existing map information when estimating online HDMaps from sensor data, 2023. arXiv:2311.10517 [cs]. 2, 3, 4, 5, 6, 8

[44] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. arXiv, 2017. arXiv:1703.06907 [cs]. 2

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 2

[46] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. PointPainting: Sequential Fusion for 3D Object Detection. In *CVPR*, 2020. arXiv:1911.10150 [cs, eess, stat]. 2

[47] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*. arXiv, 2023. arXiv:2301.00493 [cs]. 2

[48] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified Perceptual Parsing for Scene Understanding. In *ECCV*, 2018. arXiv:1807.10221 [cs]. 2

[49] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In *CVPR*. arXiv, 2020. arXiv:1805.04687 [cs]. 2

[50] Yunpeng Zhang, Zheng Zhu, and Dalong Du. OccFormer: Dual-path Transformer for Vision-based 3D Semantic Occupancy Prediction. In *ICCV*, 2023. arXiv:2304.05316 [cs]. 2

[51] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, 2018. arXiv:1711.06396 [cs]. 2

[52] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *ICLR*, 2021. arXiv:2010.04159 [cs]. 3