Gradient-Free Adversarial Training Against Image Corruption for Learning-based Steering

Yu Shen
University of Maryland
College Park, MD 20742
yushen@umd.edu

Laura Zheng
University of Maryland
College Park, MD 20742
lyzheng@umd.edu

Manli Shu
University of Maryland
College Park, MD 20742
manlis@umd.edu

Weizi Li
University of Memphis
Memphis, TN 38152
wli@memphis.edu

Tom Goldstein
University of Maryland
College Park, MD 20742
tomg@cs.umd.edu

Ming C. Lin
University of Maryland
College Park, MD 20742
lin@cs.umd.edu

Abstract

We introduce a simple yet effective framework for improving the robustness of learning algorithm against (input) image corruptions for autonomous driving, due to both internal (e.g., sensor noises and hardware abnormalities) and external factors (e.g., lighting, weather, visibility, and other environmental effects). Using sensitivity analysis with FID-based parameterization, we propose a novel algorithm exploiting basis perturbations to improve the overall performance of autonomous steering and other image processing tasks, such as classification and detection, for self-driving cars. Our model not only improves the performance on the original dataset, but also achieves significant performance improvement on datasets with multiple and unseen perturbations, up to 87% and 77%, respectively. A comparative study drawn between our approach and other SOTA techniques confirms the effectiveness of our technique in improving the robustness of neural network training for learning-based steering and other image processing tasks.

1 Introduction

Autonomous driving is a complex task that requires many software and hardware components to operate reliably under highly disparate and often unpredictable conditions. In this work, we study "learning-based steering" as it contains both perception and control (two most critical components) for autonomous driving. While on the road, vehicles are going to experience day and night, clear and foggy conditions, sunny and rainy days, as well as bright cityscapes and dark tunnels. All these external factors in conjunction with internal factors of the camera (e.g., those associated with hardware) can lead to quality variations in input data for image-based learning algorithms. One can harden machine learning systems to these degradations by simulating them at training time [6]. However, an algorithmic tool for analyzing the sensitivity of real-world neural network performance on the properties of training images is lacking. More importantly, a mechanism to leverage such a sensitivity analysis for improving learning outcomes needs to be developed. In this work, we quantify the influence of image quality on the task of "learning to steer," study how training on degraded and low-quality images can boost robustness to image corruptions, and provide a systematic approach to improve the performance of the learning algorithm based on quantitative analysis.

Image degradations can be simulated by varying attributes such as blur, noise, distortion, color representations (such as RGB or CMY) hues, saturation, and intensity values (HSV). However, identifying the correct combination of the simulated attribute parameters to obtain optimal performance on

Preprint. Under review.

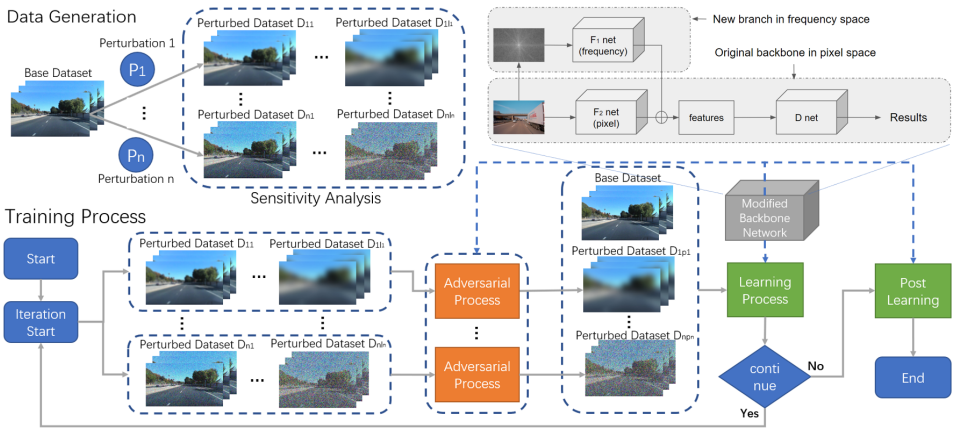

Figure 1: Pipeline of our method. **Data generation**. We generate perturbed datasets of each factor at multiple levels based on the FID-parameterized sensitivity analysis results. **Training process**. First stage: in each iteration, we first augment the training dataset with "adversarial images" given each perturbation; we then combine the base and perturbed datasets to train our model to maximize the overall performance. A frequency-space branch is added to the backbone when frequency-related perturbations (e.g., blur, noise) need to be handled. Second stage: in post-learning, the model is fine-tuned solely on clean data to boost accuracy, while performing special validation with an early break to maintain the performance on the perturbed data.

real data is a difficult—if not impossible—task, as it requires domain transfer and exploring a high dimensional parameterized space.

We then *design a systematic method for measuring the severity of image degradation and predicting the impact of such degradation on model performance*. Inspired by the use of image feature variance in sensitivity analysis [33], we measure the difference between real-world image distributions and simulated/degraded image distributions using Fréchet Inception Distance (FID). Our results confirm that FID can help predict the performance of a model trained using simulated data and deployed in the real world. Next, we use FID between different simulated datasets as a unified metric to parameterize the severity of various image degradations due to different factors.

Borrowing concepts from the adversarial attack literature [28, 35, 43], we *build a scalable training scheme for enhancing the robustness of autonomous driving against various combinations of image degradations, while* increasing *the overall accuracy of the steering task on clean data*. Our proposed method constructs a dataset of adversarially degraded images by applying optimization within the space of possible degradations during training. As shown in Fig. 1, the method begins by training on a set of real and simulated/degraded images using arbitrary degradation parameters. During each training iteration, the parameters are updated to generate a new degradation set so that the model performance is (approximately) minimized. The network is then trained on these adversarially degraded images to promote robustness. A post-training is applied as the last step to further improve the performance on clean data without weakening robustness. Our proposed algorithm uses our FID-based parameterization to discretize the search space of degradation parameters and accelerates the process of finding optimal parameters.

Experiments show that our algorithm improves the performance of "learning to steer" up to **97%** in mean accuracy over baselines, and especially improves the performance on datasets contaminated with complex combinations of perturbations (up to **87%**). It additionally boosts the test performance on degradations that are not seen during training, including simulated snow, fog, and frost (up to **77%**). We also compare our approach with other SOTA techniques (e.g., data augmentation and adversarial training) on visual processing tasks such as detection and classification. Our method consistently achieves higher performance. In addition, our method is easy to implement and can be readily integrated with other frameworks such as object detection, classification, regression, etc.

Furthermore, we propose a comprehensive robustness evaluation standard under four different scenarios: clean data, single-perturbation data, multi-perturbation data, and previously unseen data. While state-of-the-art studies usually conduct testing under one or two scenarios (e.g., ImageNet-C [20]), our work tests and verify results under four meaningful scenarios. We plan to release code

and datasets for benchmarking "autonomous driving under perturbations" using, unseen factors such as image corruptions in ImageNet-C [20], totaling **480** datasets and **26M** images.

2 Related Work

The influence of noise and distortion on real images for learning tasks has been well explored. For example, researchers have examined the impact of optical blur on convolutional neural networks and present a fine-tuning method for recovering lost accuracy using blurred images [41]. This fine-tuning method resolves lost accuracy when images are distorted instead of blurred [49]. While these fine-tuning methods are promising, Dodge and Karam [11] find that tuning to one type of image quality reduction effect would cause poor generalization to other types of effects. The comparison of image classification between deep neural networks and humans shows similar performance on good-quality images [12]. However, deep neural networks struggle significantly more than humans on low-quality, distorted, and noisy images. One study shows that adversarial perturbations are more prevalent in the Y channel in the YCbCr color space of images than the other two channels, while perturbations in RGB channels are equally distributed [29]. Wu et al. [42] studies the effect of Instagram filters on learning tasks, and introduces a lightweight de-stylization module that predicts parameters used for scaling and shifting feature maps to "undo" the changes incurred by filters.

Researchers have also explored how to improve the robustness of learning algorithms under various image quality degradations. One recent work provides a novel Bayesian formulation for data augmentation [39]. Cubuk et al. [9] proposes an approach to automatically search for improved data augmentation policies. Ghosh et al. [17] analyzes the performance of convolutional neural networks on quality degradations due to compression loss, noise, blur, and contrast, and introduces a method to improve the learning outcome. Another work [21] shows that self-supervision techniques can be used to improve model robustness and exceeds the performance of fully supervised methods. A recent method, AugMix [22] improves model robustness using data augmentation, where transformation compositions are used to create a new dataset that is visually and semantically similar to the original dataset. AugMix is compared to several other augmentation methods, which comprise Cutout [10], MixUp [45], and CutMix [44]. Gao et al. [14] proposes a technique to re-purpose software testing methods to augment the training data of DNNs, with the objective to improve model robustness. A recent work improves model generalizability by first augmenting the training dataset with random perturbations, and then minimizing worst-case loss over the augmented data [18].

Our work differs from these studies in several regards. First, we simulate adversarial conditions of image factors instead of using commonplace image conditions. Second, we conduct a systematic sensitivity analysis for preparing datasets that are representative of image degradations from multiple factors at various levels. Third, our algorithm can work with the discretized parameter space while generalizing well to the continuous parameter space. Another advantage of our approach is that we can augment the training dataset without the derivatives of the factor parameters, which may not exist or are difficult to compute.

3 Background and Setup

Task. Our target task is end-to-end steering: given a single image as input (e.g. captured by a front-facing camera on a self-driving car), output a steering angle that drives the car safely on the road [4]. A steering angle of 0 represents the forward direction. We use mean accuracy (MA) to evaluate the task since it represents overall performance under a variety of measures (See Sec. 5).

Datasets. We choose four real-world driving corpuses as our datasets: Audi [16], Honda [31], Waymo [37], and SullyChen [7]. The Audi dataset is the most recent (2020); the Honda dataset has 100+ long-time driving videos; Waymo includes many environmental conditions such as weather and lightening and is a large dataset (390k frames for perception); and the SullyChen dataset focuses on the steering task and has the longest continuous driving image sequence without road branching. The details of these datasets are provided in Appendix A.3. Other datasets, such as CIFAR-100, for various computer vision tasks are also used to demonstrate the generalizability of our technique.

Basis perturbation. We study nine basis perturbations: blur, noise, distortion, three-color (RGB) channels, and hues, saturation, and intensity values (HSV). Blur, noise, and distortion are among the most commonly used perturbations that can directly affect image quality. R, G, B, H, S, V channels are chosen because they are frequently used to represent image color spaces: RGB represent three basic color values of an image, while HSV represent three common metrics of an image. Other color spaces such as HSL or YUV have similar properties, hence are excluded.

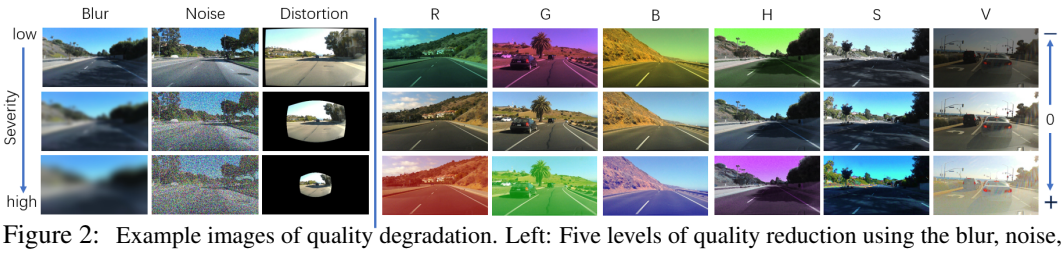

Figure 2: Example images of quality degradation. Left: Five levels of quality reduction using the blur, noise, and distortion effects (three levels are shown). Right: The original images are shown in the middle row. The top row shows examples in the lighter direction per channel for R, G, B, H, S, V, while the bottom row shows examples in the darker direction per channel.

We use Gaussian blur [2] (parameterized by standard deviation), additive white Gaussian noise (AWGN) (parameterized by standard deviation), and radial distortion [47, 48] (with radial distortion parameters $k1, k2$). For representing channel-level perturbations, we use a linear model: denote the value range of one channel C as $[a_C, b_C]$, in the darker direction, we set $v'_C = \alpha a_C + (1-\alpha)v_C$, while in the lighter direction, we set $v'_C = \alpha b_C + (1-\alpha)v_C$, where α is the severity parameter, v_C is the pixel value on clean image and v'_C is the perturbed pixel value. The default values are $a_C = 0$ and $b_C = 255$, with two exceptions. We set $a_V = 10$ to exclude a complete dark image on V channel, and $b_H = 179$ according to H channel definition. See examples in Fig. 2 and detailed description in Appendix A.2.

Test scenarios. While other studies usually test in only one or two scenarios [20], we test the performance of all methods in four scenarios with increasing complexity. Scenario 1: *Base dataset*. Test on the base dataset. Scenario 2: *Single perturbation*. Test on the datasets with perturbations, but each dataset only experiences one level of one perturbation (within blur, noise, distortion, R, G, ,B, H, S, V). Scenario 3: *Combined perturbations*. Test on the datasets with combinations of various perturbations at different levels, and each dataset has different levels of all perturbations. Scenario 4: *Unseen perturbation*. Test on the datasets with unseen perturbations at different levels. Specifically, we use "motion blur", "zoom blur", "pixelate", "jpeg compression", "snow", "frost", "fog" from ImageNet-C [20]. We apply the same perturbations/levels to all images within one dataset. (See details in Appendix A.1).

4 Improving Robustness of Learning-based Steering

In this section, we introduce our gradient-free adversarial training method to improve the robustness of learning-based steering using Sensitivity Analysis. Our method is among the first to *treat complex unforeseen perturbations as a functional combination of multiple simple "basis perturbations"*. As a result, the robustness of a model to several individual perturbations can lead to robustness against combined or unseen perturbations. In addition, *via discretized sensitivity analysis*, our approach is the first to *use FID to enable cross-factor performance comparison*, i.e., making task sensitivity w.r.t different perturbations comparable. Meanwhile, sensitivity analysis helps minimize the discretization level number, thus speed up the adversarial training process.

4.1 Basis Perturbations

There are various types of image corruptions due to different environmental effects or sensor variations, and often these corruptions can be approximated using a combination of basis perturbations, e.g., snow/frost may be simulated using water drop corruptions, Gaussian blur, and image whitening. Inspired by the concept of *basis function*, we explore the possibility of training a model on a few "basis perturbations" and the resulting model is robust against complex perturbations.

Our basis perturbation contains "basis" representations of the color space in RGB and of image metrics in HSV, which span color and intensity/brightness spaces, respectively. Blur and noise are designed to produce low and high frequency corruptions, while distortion captures the 2D positional transformation that may appear from small vibrations and motions of a camera. See comparison of different basis perturbations in Appendix. A.2. We also introduce two metrics to evaluate how well a training method can enable a model to perform on combined perturbations and previously unseen perturbations when learning only on single perturbations. Using basis perturbations to represent image corruption, we then design an algorithm to train neural networks to cope with quality-degraded images as input for autonomous driving.

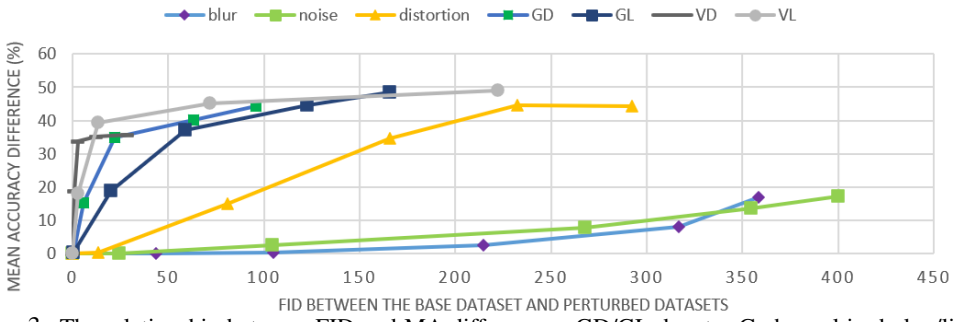

Figure 3: The relationship between FID and MA differences. GD/GL denotes G channel in darker/lighter direction, and VD/VL denotes V channel in darker/lighter direction, respectively. Sensitivity is represented by the first-order derivative of the curve. Note that the values in the near-zero FID range (i.e., < 50) are more commonly found in real-world scenarios.

4.2 Sensitivity Analysis

We use an adversarial training process in which we optimize corruption parameters to maximally disrupt model performance. If this was done using gradient optimization, it would require us to differentiate through the corruption operator to update the parameters describing the corruption. Not all gradients of basis perturbations can be easily derived, and corruptions such as distortions may have parameters with non-differentiable effects. Rather than rely on gradient optimization, we introduce an simple, discretized gradient-free method. We use Sensitivity Analysis (SA) to evenly discretize the space of parameters; we choose a discrete subset of values for each parameter that are equally "far apart" in terms of their impacts on the classifier. This is achieved using the Fréchet Inception Distance (FID) [23] to perform SA across a range of corruption types.

SA is commonly used to understand how the uncertainty of the model output (numeric or otherwise) can be apportioned to different sources of uncertainty of the model input [32, 33]. Here, we use SA to study how distortions to model input (blur, noise, distortion, and RGB/HSV brightness shifts) can be apportioned to model output. We also use SA to quantify the level of degradation caused by an corruption, and prepare datasets that are representative of image qualities at various degradation levels.

We adopt the Fréchet Inception Distance (FID) [23] as a unified metric for our SA (The reasons for adopting FID and a quantitative comparison between FID and other metrics such as L2 norm can be found in Appendix A.4). In addition, we focus on the changes in model performance according to the changes in input and define the sensitivity as the first-order derivative of MA w.r.t FID:

$$sensitivity = \frac{\partial MA^*(R \oplus F(\mathbf{p}))}{\partial FID(R, R \oplus F(\mathbf{p}))},$$

where $MA^*(D)$ is the MA test result on dataset D with the model trained on the base dataset R; $FID(A, B)$ is the FID between dataset A and dataset B (where FID is calculated in its standard formulation using a pretrained InceptionV3 network [23]); $F(\mathbf{p})$ is the perturbation with parameter $\mathbf{p}$ (e.g., the standard deviation of the Gaussian kernel), and $D \oplus F$ denotes the dataset obtained by applying perturbation F to D.

Starting from empirically-selected parameters of each factor, we generate perturbed datasets and compute their corresponding MA using the trained model on R (see numeric results in Appendix A.9). We then map the MAs and their corresponding parameter values into the FID space. By leveraging this new MA-FID space, we aim to minimize the number of sampled parameters for each perturbation for efficient training (see Sec 4.3), while preserving similar parameter curves. At a high level, we sample points more densely in the value range that has high sensitivity (closer FID values between sample points), while sampling points sparsely in the value range that has low sensitivity (See specific equation in Appendix. A.4). Examples of the resulting images are shown in Fig. 2 (more in Appendix A.1). Detailed descriptions of the final perturbed datasets are provided in Appendix A.2.

The final MA differences in the FID space are visualized in Fig. 3 (for blur, noise, distortion, and G and V channels; see the entire figure in Appendix A.4). We first observe that learning-based steering is more sensitive to the channel-level perturbations (i.e., R, G, B, H, S, V) than the image-level perturbations (i.e., blur, noise, distortion). Second, the task is least sensitive to blur and noise but most sensitive to the V channel (the intensity value). Third, for the same color channel, darker and

lighter levels appear to have different MAs at the same FID value. Compared to other "learning to steer" studies [38, 46], our method is the first to transfer perturbations from multiple parameter spaces into one space to enable cross-factor comparison.

4.3 Gradient-Free Adversarial Training

Our training process consists of two stages. In the first stage, we use iterative min-max training. At each iteration, we first choose one dataset from the datasets (with different quality degradations) of each basis perturbation (i.e., R, G, B, H, S, V, blur, noise, and distortion) to minimize validation MA, then we merge the nine chosen datasets with the base dataset to train our model while maximizing MA. The first stage stops when a pre-specified number of iterations is reached or the MA loss is below a certain threshold. Our method resembles conventional adversarial training: we improve model robustness by training the model to maximize accuracy using the base dataset plus the perturbed datasets with the minimum accuracy. The loss function is the following:

$$\max_{\theta} \min_{\mathbf{p}} MA(\theta, U_{\mathbf{p}}),$$

where $\mathbf{p}$ represents the union of all parameter levels of all basis perturbations; θ denotes the model parameters; and $U_{\mathbf{p}}$ is the training dataset. Furthermore, we add a branch in the frequency space to the backbone network to address frequency-related perturbations such as blur and noise (See details in Appendix. A.5). The effectiveness is demonstrated in our ablation study (see Table 1).

The second stage is to boost "clean" accuracy, where the model is fine-tuned solely on the base dataset. To maintain the performance on perturbed datasets, we terminate this stage if the overall validation accuracy start to decrease; otherwise, we continue this stage until it reaches the maximum number of iterations or the MA loss decreases to our expected threshold. See Algorithm 1. (Detailed explanations and the ablation study of the second stage are provided in Appendix A.6.)

Algorithm 1: Improve robustness of learning-based steering

Result: a trained model parameterized by θ
Pre-processing:
Conduct sensitivity analysis and discretize the parameters of n factors into their corresponding $l_{i=1,\ldots,n}$ levels (Sec. 4.2)
Generate new datasets for each factor with the discretized values from the base dataset R (Sec. 3 Basis Perturbation): $\{D_{i,j}\}_{i=1,2,..,n,j=1,2,..l_i}$
Initialization:
Initialize $t = 0$, the maximum number of iterations T, the number of epochs k_1 and k_2, and model parameters $\theta^{(0)}$
First stage:
while $t \leq T$ **do**
  For each factor, select D_{i,p_i} that can minimize the validation MA, where $p_i = \arg\min_j MA(\theta^{(t)}, D_{i,j})$
  Merge all selected datasets $U_{\mathbf{p}} = (\bigcup_{i=1}^{n} D_{i,p_i}) \bigcup R$
  Train the network for k_1 epochs and update $\theta^{(t+1)}$ = train($\theta^{(t)}$, $U_{\mathbf{p}}$, k_1) to maximize $MA(\theta^{(t+1)}, U_{\mathbf{p}})$
  Break if stop conditions are met; otherwise set $t = t + 1$
end
Second stage:
Train the network with $\theta^{(final)}$ = train($\theta^{(T)}$, R, k_2) and validate the network on $U_{\mathbf{p}}$ (for k_2 epochs or early break if conditions are met)

Our method offers several advantages: 1) the training data is augmented without re-train the model, thus improving efficiency; 2) it provides the flexibility to generate datasets at various discretized levels of the factor parameters; 3) it does not require the derivatives of factor parameters; other methods that optimize factor parameters in the continuous space require computing derivatives, which can be difficult (e.g., for distortion); 4) it generalizes to not only unseen parameters of individual factors but also the composition of unseen parameters of multiple factors; and 5) it is easy to implement and can be readily integrated with other frameworks such as object detection, classification, and regression.

5 Experiments and Results

Setups. All experiments are conducted using one Intel(R) Xeon(TM) W-2123 CPU, two Nvidia GTX 1080 GPUs, and 32G RAM. We use the Adam optimizer [24] with learning rate 0.0001 and batch size 128 for training. The maximum number of epochs is 2,000. The dataset setup is explained in Sec. 3. We use the maximum MA improvement (MMAI), the average MA improvement (AMAI), and mean Corruption Errors (mCE) [20] as the evaluation metrics.

Backbone. We choose the model described in [4] as the main backbone. We select this model as it has been used to steer an autonomous vehicle successfully in both real world [4] and virtual world [25]. In addition, six other networks are tested to show the generalizability of our method.

Evaluation metrics. We define the accuracy w.r.t a threshold τ as $acc_\tau = count(|v_{predicted} - v_{actual}| < \tau)/n$, where n denotes the number of test cases; $v_{predicted}$ and v_{actual} indicate the predicted and ground-truth value, respectively. We compute mean accuracy (MA) as $\sum_\tau acc_{\tau \in \mathcal{T}}/|\mathcal{T}|$, where $\mathcal{T} = \{1.5, 3.0, 7.5, 15, 30, 75\}$ contains empirically selected thresholds of steering angles. Lastly, we use the maximum MA improvement (denoted as MMAI), the average MA improvement (denoted as AMAI), and mean Corruption Errors (mCE) [20] as the evaluation metrics.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
Data Augmentation	-0.44	46.88	19.97	51.34	**36.1**	11.97	75.84	27.5	7.92	81.51
Adversarial Training	-0.65	30.06	10.61	74.42	17.89	6.99	86.82	16.9	8.17	89.91
MaxUp	-7.79	38.30	12.83	66.56	26.94	16.01	72.60	23.43	5.54	81.75
AugMix	-5.23	40.27	15.01	67.49	26.81	15.45	68.38	28.70	8.85	87.79
Ours w/o FS	0.93	48.57	20.74	49.47	33.24	17.74	63.81	29.32	9.06	76.20
Ours	**2.12**	**49.97**	**23.92**	**37.30**	33.15	**22.12**	**54.38**	**33.16**	**13.81**	**61.61**

Table 1: Performance of different methods against the baseline [4] on SullyChen dataset. We compare the basic data augmentation method (simply combine all perturbed datasets into training), an adversarial training method [36], MaxUp [18], and AugMix [22]. Overall, our method outperforms all other methods (i.e., highest MA improvements and lowest error in mCEs) in practically all scenarios. Notice ours w/o FS (without frequency space) also outperforms other techniques.

Comparison with different methods. We compare our method with four other methods: an adversarial training method [36], a basic data augmentation method, MaxUp [18], and AugMix [22], to see the performance improvement over the baseline method [4]. For the basic data augmentation method, we simply merge all perturbed datasets for model training.

From Table 1, we observe that our method outperforms other methods under all metrics in all scenarios: not only on the clean dataset but also on perturbed datasets. Notably, our algorithm improves the performance of "learning to steer" up to 50% in MMAI, while reducing mCE by 60% over the baseline (Scenario 2). Our method also improves the task performance using the combined datasets (Scenario 3) up to 33%. Lastly, when tested on unseen factors (Scenario 4), our algorithm maintains the best performance by 34% in MMAI, while reducing mCE to 62%.

Compared to AugMix [22], our adversarial approach can select the most challenging datasets for training, thus improving model robustness. MaxUp [18] selects only the worst case among all augmentation data, which may lead to the loss of data diversity. In contrast, our method selects the worst cases in *all* perturbation types (i.e., one dataset per factor), thus improving generalizability. Compared to [36], which performs an adversarial process on the entire pixel space with only norm constraints, our method is able to utilize vast prior information generated by sensitivity analysis to reduce the search space. Lastly, compared to the basic data augmentation method, which uses all generated data in training, our method selects the most useful data for training, and thus improves computational efficiency while minimizing data generation.

Comparison with different backbones. We test three backbones: Nvidia network [4], Comma.ai network [34], and ResNet152 [19], on the Honda dataset. The results shown in Table 2 indicate that our method outperforms AugMix in most cases. In general, our method achieves better performance on shallow networks than deep networks. But even on very deep networks such as ResNet152, our method achieves more than 5% improvement in all cases, except Scenario 1.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
AugMix+Nvidia	-0.12	40.64	10.94	76.48	25.97	16.79	64.41	**22.23**	5.99	84.95
Ours+Nvidia	**2.48**	**43.51**	**13.51**	**67.78**	**28.13**	**17.98**	**61.12**	16.93	**6.70**	**80.92**
AugMix+Comma.ai	-5.25	55.59	9.56	86.31	31.32	**0.77**	**106.1**	37.91	7.97	89.99
Ours+Comma.ai	**0.36**	**62.07**	**15.68**	**70.84**	**38.01**	0.74	108.32	**42.54**	**12.15**	**77.08**
AugMix+ResNet152	-4.23	20.84	1.45	96.24	12.21	6.71	80.19	15.40	2.87	97.62
Ours+ResNet152	**-0.96**	**24.29**	**5.19**	**79.76**	**16.05**	**8.02**	**75.16**	**16.58**	**5.33**	**85.68**

Table 2: Performance improvement of different backbones against the baseline performance using the Honda dataset. Our method outperforms AugMix in most cases. Notice that the methods with ResNet152 do not improve as much as the first two networks (since the ResNet152 baseline already has high performance).

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
AugMix on SullyChen	-5.23	40.27	15.01	67.49	26.81	15.45	68.38	28.70	8.85	87.79
Ours on SullyChen	**1.46**	**49.76**	**22.87**	**40.84**	**33.15**	**22.12**	**54.38**	**33.87**	**13.51**	**62.50**
AugMix on Audi	-8.24	81.89	32.22	55.27	75.49	50.23	41.98	73.06	27.39	77.51
Ours on Audi	**5.98**	**97.57**	**48.50**	**10.27**	**87.56**	**62.38**	**25.80**	**77.22**	**32.71**	**39.14**
AugMix on Honda10k	-0.12	40.64	10.94	76.48	25.97	16.79	64.41	**22.23**	5.99	84.95
Ours on Honda10k	**2.48**	**43.51**	**13.51**	**67.78**	**28.13**	**17.98**	**61.12**	16.93	**6.70**	**80.92**
AugMix on Honda100k	-11.41	63.85	14.08	70.64	**68.95**	47.69	40.12	**61.68**	16.32	88.36
Ours on Honda100k	**-2.55**	**67.35**	**19.88**	**53.26**	65.10	**48.60**	**36.94**	51.90	**18.29**	**72.84**
AugMix on Waymo	18.27	45.40	23.30	59.31	**22.95**	16.92	66.36	**57.65**	29.10	55.63
Ours on Waymo	**20.34**	**46.85**	**26.76**	**52.84**	21.34	**18.24**	**64.58**	56.98	**31.12**	**53.18**

Table 3: Performance improvement of different datasets with different sizes against the baseline using the Nvidia backbone. Our method outperforms AugMix considerably in most cases.

Comparison on different datasets. To demonstrate that our method does not overfit a particular dataset, we experiment four independent datasets: Audi [16], Honda [31], SullyChen [7], and Waymo [37]. We use the Nvidia network as the backbone for these experiments. Table 3 shows that our method achieves consistently better performance across all four datasets. Furthermore, our method obtains up to **97%, 87%, 77%** accuracy improvement on the single, combined, unseen perturbations, while achieving **90%, 74%, 61%** relative error reduction in some cases, respectively.

Performance on other image processing tasks. To show the generalizability of our method, we test it on classification using CIFAR-100. For a fair comparison, we use the same setting as of AugMix (i.e., same perturbations, training and testing data, backbone, and code). As shown in Table 4, our method consistently outperforms all backbones, reducing **16.3%** to **34.1%** in mean errors. The data for other methods come from the AugMix paper [22]. We also test our method on detection in driving. Specifically, we use the Audi dataset [16] and the Yolov4 network [3] as base settings, and implement ours based on Yolov4. The result shows that our algorithm improves robustness in most scenarios (about 3%-5% mAP on average). See Appendix A.7.

Decoupling Ratio and Generalization Ratio. We propose two new metrics to evaluate how well a training method can allow a model to perform on combined perturbations and unseen perturbations respectively, while only learning on single perturbations one at a time.

Definition 5.1 ***(Decoupling Ratio)*** *Let $MA(D_1, D_2)$ be the mean accuracy (MA) result of a model trained on dataset D_1 and test on dataset D_2. Let $P_i^*(D)$ be the ith ($i = 1, ..., n$) perturbation taking dataset D as input and producing a perturbed dataset. Let $NP^*(D) = P_n^*(P_{n-1}^*(...P_1^*(D))...)$ (datasets with combined/nested perturbations), $UP^*(D) = \bigcup_{i=1}^{n} P_i^*(D)$ (the union of datasets with single perturbations), D_{tr} be the training set, and D_{te} be the test set, then the decoupling ratio is*

$$r_d = \frac{MA(UP^*(D_{tr}), NP^*(D_{te}))}{MA(UP^*(D_{tr}), UP^*(D_{te}))}.$$

$r_d = 1$ means when the model is trained using the union of single-perturbation images, the test performance on combined-perturbation images (not training domain) can be as good as the test

	Standard	Cutout	Mixup	CutMix	AutoAugment*	Adv Training	AUGMIX	Ours
AllConvNet	56.4	56.8	53.4	56.0	55.1	56.0	42.7	**25.6**
DenseNet	59.3	59.6	55.4	59.2	53.9	55.2	39.6	**26.3**
WideResNet	53.3	53.5	50.4	52.9	49.6	55.1	35.9	**20.5**
ResNeXt	53.4	54.6	51.4	54.1	51.3	54.4	34.9	**15.4**
Mean	55.6	56.1	52.6	55.5	52.5	55.2	38.3	**22.0**

Table 4: Average classification error (in %) across several architectures. Our method is able to reduce mean corruption errors than the previous SOTA methods by 16.3%-34.1% on CIFAR-100-C data.

performance on the union of single-perturbation images (training domain). Note that different models and training methods may influence the MA function, thus potentially affecting the decoupling ratio.

Definition 5.2 ***(Generalization Ratio)*** *Keeping the definitions of $MA(D_1, D_2)$, $P_i^*(D)$, D_{tr}, D_{te}, and $UP^*(D)$ as of Decoupling Ratio, let $Q_j^*(D)$ be the jth ($j = 1, ..., m$) perturbation where $P_i^* \neq Q_j^*$, and $UQ^*(D) = \bigcup_{j=1}^{m} Q_j^*(D)$ (the union of datasets with unseen single perturbations), then the generalization ratio is*

$$r_g = \frac{MA(UP^*(D_{tr}), UQ^*(D_{te}))}{MA(UP^*(D_{tr}), UP^*(D_{te}))}.$$

$r_g = 1$ means when the model is trained using the union of single-perturbation images, the test performance on unseen-perturbation images (not training domain) can be as good as the test performance on the union of known single-perturbation images (training domain). Similarly, different models and training methods may influence the MA function, thus potentially affecting the generalization ratio.

Tested on our dataset ($UP^*(D_{tr})$ is the training set with our basis perturbations, $NP^*(D_{te})$ is the data in Scenario 3, and $UQ^*(D_{te})$ is the data in Scenario 4), our method can achieve a high decoupling ratio $r_d = \frac{0.7463}{0.8836} = 0.84$ and a high generalization ratio $r_d = \frac{0.8291}{0.8836} = 0.94$. We exclude these two ratios for other methods because they did not trained on only single perturbations.

Effectiveness visualization. Using the salience map on several combined samples in Fig. 7 shown in Appendix A.8, we demonstrate that our method can help the network to focus on important areas (e.g., the road in front) instead of random areas on perturbed images. We also show t-SNE [26] of feature embeddings from the baseline and our method in Fig. 8 shown in Appendix A.8. As a result, features from our method are more uniformly distributed, indicating the reduction of the domain gaps created by the perturbations, thus improving robustness.

Benchmarking datasets. We plan to release our perturbed datasets for benchmarking, which will contain augmented datasets from Audi [16], Honda [31], Waymo [37], and SullyChen dataset [7]. Each will include a base dataset and datasets with five levels of perturbation in blur, noise, and distortion, ten levels of variations in the channels R, G, B, H, S, V, multiple combined perturbations over all nine factors using our implementation, and five levels of each unseen simulated factor, including snow, fog, frost, motion blur, zoom blur, pixelate, and jpeg compression using ImageNet-C. In total, there are 480 datasets and about 26M images. The ground-truth steering angles (or angular velocity of the vehicle) for all images will also be provided for validation, along with the code to generate the perturbed datasets. The parameters for data generation can be found in Appendix A.2.

6 Conclusion and Future Work

In this paper, we first analyze the influence of different image-quality attributes on the performance of the task "learning to steer". We study nine image attributes and find that image degradations due to perturbation on these 9 attributes can impact task performance at various degrees. By using FID as a unified metric, we conduct sensitivity analysis in the MA-FID space. Leveraging the sensitivity analysis results, we propose an effective and efficient training method to improve the generalization of learning-based steering under various image perturbations. Our model not only improves the task performance on the base dataset, but also achieves significant performance improvement on datasets with a mixture of perturbations (up to 87%), as well as unseen adversarial examples including snow, fog, and frost (up to 77%).

Our method can be easily extended and applied beyond the set of factors and the learning algorithms analyzed in this study. It can also generalize to analyzing any arbitrarily high number of image/input factors, other learning algorithms, and multimodal sensor data. Lastly, other autonomous systems where the *perception-to-control* functionality plays a key role can possibly benefit from our technique

as well. We will release the generated datasets for benchmarking the robustness study of learning algorithms for autonomous driving, as well as the code. Our method currently uses discretization to achieve efficient training, but further optimization for our implementation is possible. Our framework is generalizable to other image factors, learning algorithms, multimodal sensor data, and other perception-to-control tasks.

References

[1] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous, and James McBride. Ford multi-av seasonal dataset. *arXiv preprint arXiv:2003.07969*, 2020.

[2] Isabelle Bégin and Frank Ferrie. Blind super-resolution using a learning-based approach. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 85–89, 2004.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[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. *arXiv preprint arXiv:1903.11027*, 2019.

[6] Qianwen Chao, Huikun Bi, Weizi Li, Tianlu Mao, Zhaoqi Wang, Ming C. Lin, and Zhigang Deng. A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. *Computer Graphics Forum*, 39(1):287–308, 2019.

[7] Sully Chen. A collection of labeled car driving datasets, https://github.com/sullychen/driving-datasets, 2018.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[9] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[10] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[11] Samuel Dodge and Lina Karam. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119*, 2017.

[12] Samuel Dodge and Lina Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th international conference on computer communication and networks (ICCCN)*, pages 1–7. IEEE, 2017.

[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.

[14] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 1147–1158, 2020.

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[16] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset. 2020.

[17] S. Ghosh, R. Shet, P. Amon, A. Hutter, and A. Kaup. Robustness of deep convolutional neural networks for image degradations. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2916–2920, 2018.

[18] Chengyue Gong, Tongzheng Ren, Mao Ye, and Qiang Liu. MaxUp: A Simple Way to Improve Generalization of Neural Network Training. *arXiv e-prints*, page arXiv:2002.09024, February 2020.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[21] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[22] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020.

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Weizi Li, David Wolinski, and Ming C. Lin. ADAPS: Autonomous driving via principled simulations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7625–7631, 2019.

[26] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[27] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[29] Camilo Pestana, Naveed Akhtar, Wei Liu, David Glance, and Ajmal Mian. Adversarial Perturbations Prevail in the Y-Channel of the YCbCr Color Space. *arXiv e-prints*, page arXiv:2003.00883, February 2020.

[30] Matthew Pitropov, Danson Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *arXiv preprint arXiv:2001.10117*, 2020.

[31] Vasili Ramanishka, Yi-Ting Chen, Teruhisa Misu, and Kate Saenko. Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7699–7707, 2018.

[32] Andrea Saltelli. Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590, 2002.

[33] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[34] Eder Santana and George Hotz. Learning a driving simulator. *arXiv preprint arXiv:1608.01230*, 2016.

[35] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*, 2019.

[36] Manli Shu, Zuxuan Wu, Micah Goldblum, and Tom Goldstein. Preparing for the worst: Making networks less brittle with adversarial batch normalization. *arXiv preprint arXiv:2009.08965*, 2020.

[37] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *arXiv*, pages arXiv–1912, 2019.

[38] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[39] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. In *Advances in neural information processing systems*, pages 2797–2806, 2017.

[40] Frederick Tung, Jianhui Chen, Lili Meng, and James J Little. The raincouver scene parsing benchmark for self-driving in adverse weather and at night. *IEEE Robotics and Automation Letters*, 2(4):2188–2193, 2017.

[41] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016.

[42] Zhe Wu, Zuxuan Wu, Bharat Singh, and Larry Davis. Recognizing instagram filtered images with feature de-stylization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:12418–12425, 04 2020.

[43] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *CVPR*, 2020.

[44] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision (ICCV)*, 2019.

[45] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[46] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 132–142. IEEE, 2018.

[47] Zhengyou Zhang. A flexible new technique for camera calibration. *Technical Report MSR-TR-98-71, Microsoft Research*, 1998.

[48] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[49] Yiren Zhou, Sibo Song, and Ngai-Man Cheung. On classification of distorted images with deep convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1213–1217. IEEE, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our method improves the robustness of target algorithms (methodology), and improve the safety for autonomous driving (application).

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sec.**??**.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Sec.**??**.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Data is provided as a dataset, see Sec. 5. Code will be provided after the paper is accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Sec. 5, Appendix. A.2, and Appendix. A.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We show experiment result on different backbones, different datasets, different tasks, compare to different methods, and perform consistently better. Also error bars for each of them will be too computationally expensive to get.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Sec. 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 3 and Sec. 5.
   (b) Did you mention the license of the assets? [No] We specify the source of the assets. People can find the asset's license in their official website.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We plan to provide a new benchmark, see Benchmarking Datasets subsection in Sec. 5.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use public data or code from their official website or git, no special data or code.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Benchmarking Datasets subsection in Sec. 5.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Dataset Samples

We show different kinds of perturbations in our benchmarks in Fig.4. Specifically, our benchmarks include 9 basic types of perturbations, including Gaussian blur, Gaussian noise, radial distortion, and RGB and HSV channels. Another type of datasets include multiple perturbations, where we create multiple random combinations of the basic perturbations. We also include 7 types of previously unseen perturbations (during training) from ImageNet-C [20], which are snow, fog, frost, motion blur, zoom blur, pixelate, and jpeg compression. For each type of perturbation, we generate 5 or 10 levels of varying intensity based on sensitivity analysis in the FID-MA space.

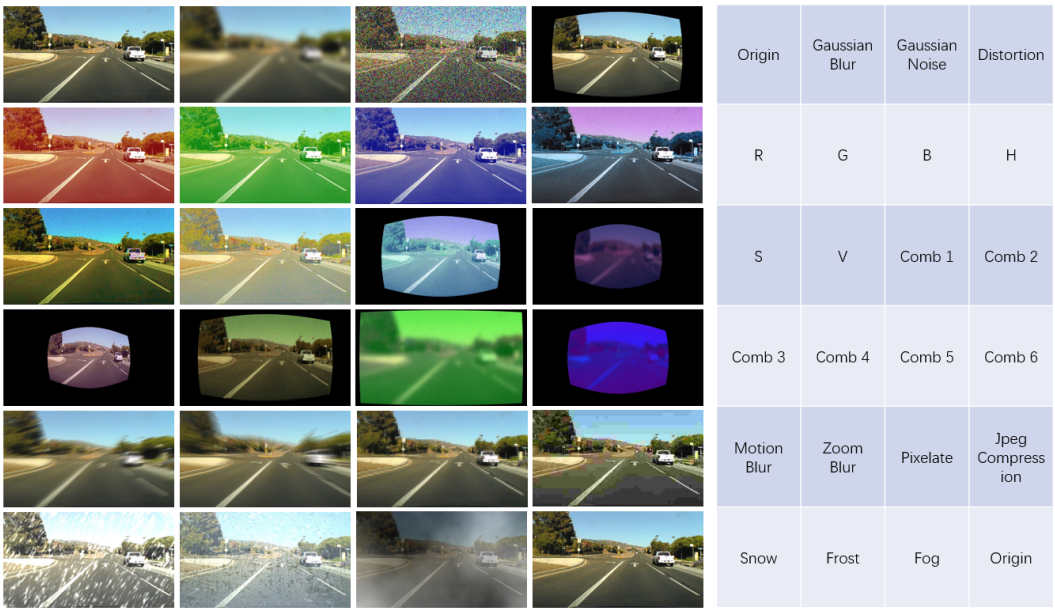

Figure 4: Sample images of our benchmark. We show our benchmark has 22 different types of perturbations. Also, we have 10 levels for R, G, B, H, S, V (5 levels in darker and 5 levels in lighter shades), and 5 levels for each of the other types of perturbations.

We show more image samples of unseen perturbations in Fig. 5.

Figure 5: Unseen perturbation examples in our experiments. We use "snow", "frost", "fog" (left to right; first row), and "motion blur", "zoom blur", "pixelate", "jpeg compression" (left to right; second row) from the corruptions in ImageNet-C [20].

A.2 Perturbed Datasets

We use Gaussian blur [2] (w.r.t standard deviation), additive white Gaussian noise (AWGN) (w.r.t standard deviation), and radial distortion [48] (w.r.t radial distortion parameter $k1, k2$), respectively. For representing channel-level perturbations, we use a linear model: denote the value range of one

channel as $[a, b]$, in the darker direction, we set $v_{new} = \alpha a + (1 - \alpha)v$; in the lighter direction, we set $v_{new} = \alpha b + (1 - \alpha)v$. The default values are $a_C = 0$ and $b_C = 255$, where C represents one channel. To exclude a complete dark image, we set $a_V = 10$ and $b_H = 179$.

In our sensitivity analysis experiments, we first select 10 samples for each of blur, noise, distortion, channel (R, G, B, H, S, V) darker, and channel lighter, then reduce to $n = 5$. We set $n = 5$ since a smaller number like $n = 2$ will decrease the algorithm performance greatly, while a larger number like $n = 8$ will decrease the efficiency dramatically.

The final representative datasets from the sensitivity analysis and used for improving the generalization of the learning task are introduced in the following.

- R: the base dataset, Audi [16], Honda [31], or SullyChen [7] dataset;
- $B1, B2, B3, B4, B5$: add Gaussian blur to R with standard deviation $\sigma = 1.4$, $\sigma = 2.9$, $\sigma = 5.9$, $\sigma = 10.4$, $\sigma = 16.4$, which are equivalent to using the kernel (7, 7), (17, 17), (37, 37), (67, 67), (107, 107), respectively;
- $N1, N2, N3, N4, N5$: add Gaussian noise to R with $(\mu = 0, \sigma = 20)$, $(\mu = 0, \sigma = 50)$, $(\mu = 0, \sigma = 100)$, $(\mu = 0, \sigma = 150)$, $(\mu = 0, \sigma = 200)$, respectively;
- $D1, D2, D3, D4, D5$: distort R with the radial distortion $(k_1 = 1, k_2 = 1)$, $(k_1 = 10, k_2 = 10)$, $(k_1 = 50, k_2 = 50)$, $(k_1 = 200, k_2 = 200)$, $(k_1 = 500, k_2 = 500)$, respectively. k_1 and k_2 are radial distortion parameters, the focal length is 1000, and the principle point is the center of the image.
- $RD1/RL1, RD2/RL2, RD3/RL3, RD4/RL4, RD5/RL5$: modify the red channel of R to darker (D) / lighter (L) values with $\alpha = 0.02$, $\alpha = 0.2$, $\alpha = 0.5$, $\alpha = 0.65$, $\alpha = 1$.
- $GD1/GL1, GD2/GL2, GD3/GL3, GD4/GL4, GD5/GL5$: modify the green channel of R to darker (D) / lighter (L) values with $\alpha = 0.02$, $\alpha = 0.2$, $\alpha = 0.5$, $\alpha = 0.65$, $\alpha = 1$.
- For B, H, S, V channels, we use similar naming conventions for notation as for the red and green channels.
- $Comb1$: $R_\alpha = -0.1180$, $G_\alpha = 0.4343$, $B_\alpha = 0.1445$, $H_\alpha = 0.3040$, $S_\alpha = -0.2600$, $V_\alpha = 0.1816$, $Blur_\sigma = 3$, $Noise_\sigma = 10$, $Distort_k = 17$
- $Comb2$: $R_\alpha = 0.0420$, $G_\alpha = -0.5085$, $B_\alpha = 0.3695$, $H_\alpha = -0.0570$, $S_\alpha = -0.1978$, $V_\alpha = -0.4526$, $Blur_\sigma = 27$, $Noise_\sigma = 7$, $Distort_k = 68$
- $Comb3$: $R_\alpha = 0.1774$, $G_\alpha = -0.1150$, $B_\alpha = 0.1299$, $H_\alpha = -0.0022$, $S_\alpha = -0.2119$, $V_\alpha = -0.0747$, $Blur_\sigma = 1$, $Noise_\sigma = 6$, $Distort_k = 86$
- $Comb4$: $R_\alpha = -0.2599$, $G_\alpha = -0.0166$, $B_\alpha = -0.2702$, $H_\alpha = -0.4273$, $S_\alpha = 0.0238$, $V_\alpha = -0.2321$, $Blur_\sigma = 5$, $Noise_\sigma = 8$, $Distort_k = 8$
- $Comb5$: $R_\alpha = -0.2047$, $G_\alpha = 0.0333$, $B_\alpha = 0.3342$, $H_\alpha = -0.4400$, $S_\alpha = 0.2513$, $V_\alpha = 0.0013$, $Blur_\sigma = 35$, $Noise_\sigma = 6$, $Distort_k = 1$
- $Comb6$: $R_\alpha = -0.6613$, $G_\alpha = -0.0191$, $B_\alpha = 0.3842$, $H_\alpha = 0.3568$, $S_\alpha = 0.5522$, $V_\alpha = 0.0998$, $Blur_\sigma = 21$, $Noise_\sigma = 3$, $Distort_k = 37$

The datasets $Comb1$ through $Comb6$ are generated by randomly sampling parameters of each type of perturbation, e.g. blur, noise, distortion, and RGB and HSV channels, and combining these perturbations together. The parameters listed here are the parameters for the corresponding examples used in the experiment.

We also show the comparison when choosing different basis perturbations in Table. 5. The final set we used are better than other sets (e.g., V channel only, or V channel + B channel + Blur) on clean and unseen perturbation scenarios. Notice we don't compare them on single perturbation and combined perturbation scenarios, as the basis perturbations are different.

A.3 Dataset details

We use Audi dataset [16], Honda dataset [31], Waymo dataset [37], and SullyChen dataset [7]. Among the autonomous driving datasets, Audi dataset is one of the latest dataset (2020), Honda dataset is one of the datasets that have a large amount of driving videos (over 100+ videos), Waymo dataset includes many environmental conditions such as weather and lightening and is a large dataset

	Scenarios			
	Clean	Unseen Perturbation		
Basis Perturbations	AMAI↑	MMAI↑	AMAI↑	mCE↓
V channel	-3.33	14.61	1.64	98.54
V channel, B channel, Blur	-0.83	15.77	2.79	95.64
Ours	**2.12**	**33.16**	**13.81**	**61.61**

Table 5: Performance of different basis perturbations. The set used is better than other sets on clean and unseen perturbation scenarios. We do not compare them on single perturbation and combined perturbation scenarios, as the basis perturbations are different.

(390k frames for perception), and SullyChen dataset is collected specifically for steering task and has a relatively long continuous driving image sequence on a road without branches and has relatively high turning cases.

For Audi dataset [16], we use the "Gaimersheim" package which contains about 15,000 images with about 30 FPS. For efficiency, we adopt a similar approach as in [4] by further downsampling the dataset to 15 FPS to reduce similarities between adjacent frames, keep about 7,500 images and align them with steering labels. For Honda dataset [31], which contains more than 100 videos, we first select 30 videos that are most suitable for learning to steer task, then we extract 11,000 images for Honda10K and 110,000 images for Honda100K from them at 1 FPS, and align them with the steering labels. For SullyChen dataset [7], images are sampled from videos at 30 frames per second (FPS). We then downsample the dataset to 5 FPS. The resulting dataset contains approximately 10,000 images. All of them are then randomly split into training/validation/test data with an approximate ratio 20:1:2. For Waymo dataset [37], we use the data from the perception part directly, which is already split into train/validation/test folders (exclude the domain adaptation data). The training set is about 163k frames, while the test set is about 33k frames. The Waymo dataset doesn't contain steering labels directly, but it contains angular velocity data from IMU. Instead of predicting steering angle, we predict the angular velocity w.r.t. the gravity axis. The only modification is we scale up the angular velocity to meet the range of steering angle, to make it a similar regression problem as steering regression.

There are several good autonomous driving datasets, but not all of them are suitable for the end-to-end learning to steer task. For example, KITTI [15], Cityscapes [8], OxfordRoboCar [27], Raincouver [40], etc., do not contain steering angle labels. Some well-known simulators like CARLA [13] can generate synthetic datasets, but our work focuses on real-world driving using real images. There are also several other datasets that contain steering angle labels (e.g., nuScenes [5], Ford AV [1], Canadian Adverse Driving Conditions [30], etc), but we didn't use them all because the results on the three datasets we chose can already show the effectiveness of our method.

A.4 FID-MA and L2D-MA

We adopt the Fréchet Inception Distance (FID) [23] as a unified metric for our sensitivity analysis (instead of using the parameter values of each image factor) for three reasons. First, given the autonomous driving system is nonlinear, variance-based measures would be more effective for sensitivity analysis of the network. FID can better capture different levels of image qualities than the parameter values of each factor, because the correspondence between the parameter values and image quality of each factor is not linear. Second, using FID, we can map the parameter spaces of all factors into one space to facilitate the sensitivity analysis. Lastly, FID serves as a comprehensive metric to evaluate the distance between two image datasets: image pixels and features, and correlations among images—these meaningful factors to interpret the performance of a learning-based task—are all taken into consideration.

We first empirically select m parameter values for blur, noise and distortion perturbation severity, and $2m$ parameter values for R, G, B, H, S, V (m in the darker direction and m in the lighter direction), generate perturbed datasets using these parameter values, and compute their corresponding MA using the trained model on R (see numeric results in Appendix A.9). At this point, we can obtain the relationship between MA and parameter values for each factor, but **the parameter values for each factor are not directly comparable to each other**. Notice for each perturbation, one parameter value can generate one perturbed dataset. Thus, we can calculate the FID between this new dataset and clean dataset. In this way, we map the correspondences between the MAs and the

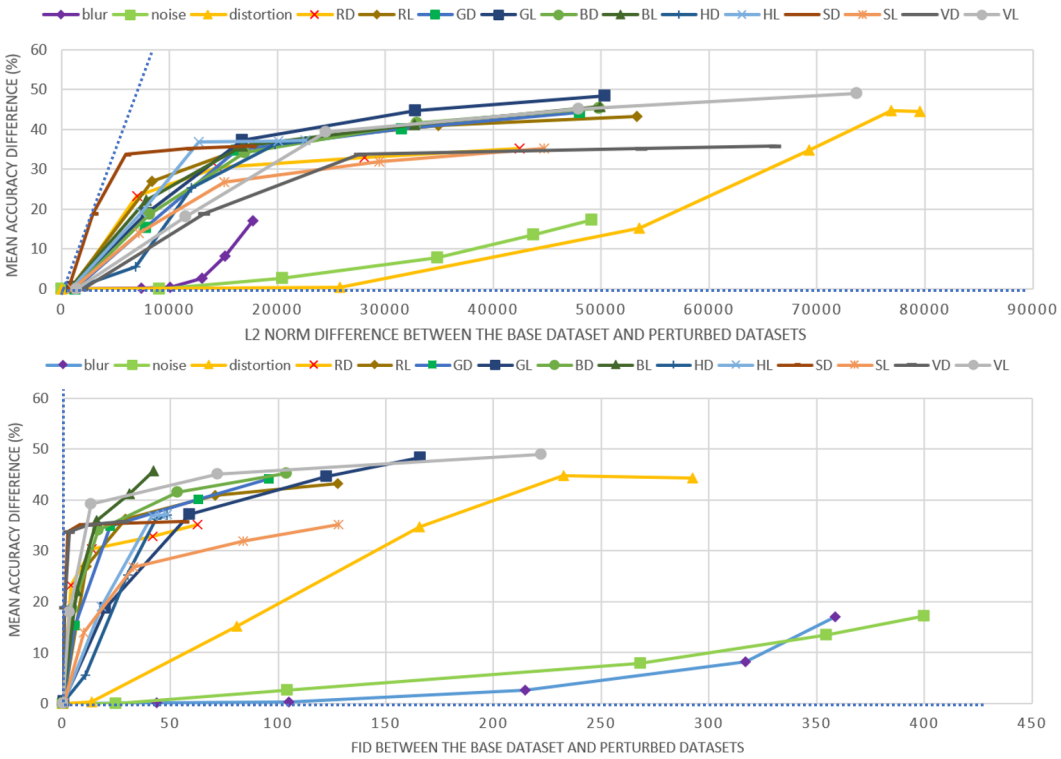

Figure 6: The relationship between L2 norm distance and MA difference (top), and the relationship between FID and MA difference (bottom). The FID space can better capture the difference among various factors affecting image quality better than the L2D space, i.e., the range of the curves' first-order derivative is larger in FID space than in L2D space (see the angle between the two dot lines).

parameter values into the FID space, i.e., FID-MA relationship. By leveraging this new FID-MA space, we can minimize the number of parameter samples for each perturbation, while maintaining a similar FID-MA curve between the sampled dataset and the original one to improve the computational efficiency of training (see Sec 4.3). **A high-level guideline is, in the range that has high sensitivity (See definition in Sec. 4.2) there are denser sample points (closer FID values between sample points), while in the range that has low sensitivity there are sparser sample points**. Specifically, we retain the min and max parameter values, and pick $n-2$ other parameter values to maximize the minimum 'MA and normalized FID difference' between adjacent sample points (here we assume the FID-MA curve is approximately monotonic):

$$\max_{\substack{Q=\{p_{q_1},p_{q_2},\ldots,p_{q_n}\}\subseteq P \\ p_{q_1}<p_{q_2}<\ldots<p_{q_n}}} \min_i |MA(p_{q_{i+1}}) - MA(p_{q_i})| + \alpha |FID(p_{q_{i+1}}) - FID(p_{q_i})|$$

where $P = \{p_1, p_2, ..., p_m\}$ is the m parameter values we chose in the beginning, $Q = \{p_{q_1}, p_{q_2}, ..., p_{q_n}\} \subseteq P$ is the n parameter values we want to keep, $MA(p_j)$ and $FID(p_j)$ are the mean accuracy value and FID value related to the parameter value p_j, and α is the normalization parameter which equals to the reciprocal of the largest FID (within the FID values related to the chosen parameter values). Notice this is not the only criterion to achieve the goal of "sample denser points in high sensitivity range and sample sparser points in low sensitivity range". We also experimented with other criteria, like replacing $|MA(p_{q_{i+1}}) - MA(p_{q_i})| + \alpha|FID(p_{q_{i+1}}) - FID(p_{q_i})|$ with $|MA(p_{q_{i+1}}) - MA(p_{q_i})|$ (MA difference only), which achieves slightly worse results but still works.

We set $m = 10$, try $n = 3, 5, 8$, and find $n = 5$ is the best one ($n = 5$ can lead to similar robustness of the model as $n = 8$ while taking less time, and more robust than $n = 3$). Examples of the resulting

images are shown in Fig. 2 (more in Appendix A.1). Detailed descriptions of the final perturbed datasets are provided in Appendix A.2. We also provide a detailed analysis of MA differences in FID space in Appendix A.4.

The final MA differences in the FID space are visualized in Fig. 6. Since FID aligns different factors into the same space, we can compare the performance loss of all factors at various levels. Notice that the values in the near-zero FID range (i.e., FID< 50) are more commonly found in real-world applications. We first observe that the learning-based steering task is more sensitive to the channel-level perturbations (i.e., R, G, B, H, S, V) than the image-level perturbations (i.e., blur, noise, distortion). Second, the task is least sensitive to blur and noise but most sensitive to the V channel, the intensity value. Third, for the same color channel, darker and lighter levels appear to have different MAs at the same FID values. Compared with other analysis studies on the "learning to steer" task, e.g., [38] and [46], our method is the first to transfer perturbations from multiple parameter spaces into one unified space to enable the cross-factor comparison, e.g., the task is more sensitive to the V-channel perturbation than perturbations in other attributes.

We illustrate the relationship between FID and Mean Accuracy (MA) Difference, and the relationship between L2 norm distance (L2D) and Mean Accuracy (MA) Difference in Fig. 6. As shown in the figure, the FID space can better capture the difference among various factors affecting image quality better than the L2D space, i.e., the range of the curves' first-order derivative is larger in FID space than in L2D space (see the angle between the two dot lines).

A.5 Frequency Branch of Our Method

When there are frequency-related perturbations (e.g., using Gaussian noise is increasing high frequency component of the image, while using blur operation will reduce high frequency component), a frequency branch can be added to our architecture. Formally, we do a standard 2-D Fourier Transform, and using the absolute value of each complex number and form another channel of the image, thus the input image of the network has 4 channels. The transform equation is:

$$Y_{p,q} = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \omega_m^{jp} \omega_n^{kq} X_{j,k}$$

where ω_m and ω_n are complex roots of unity:

$$\omega_m = e^{-2\pi i/m}$$
$$\omega_n = e^{-2\pi i/n}$$

i is the imaginary unit. p and j are indices that run from 0 to $m-1$, and q and k are indices that run from 0 to $n-1$.

Notice this frequency branch is optional, our method can already outperform others without this branch. It can help improve the accuracy, but also has limitation: about 1.5x training time. Thus this is an option depends on whether users want the model to be more accurate or more efficient.

A.6 Second Stage of Our Method

As introduced in Sec. 4.3, the second stage is designed to boost clean accuracy, where the model is fine-tuned solely on clean data. To meanwhile maintain the performance on the perturbed data, we terminate this stage if the overall validation accuracy on clean and perturbed data decreases. Otherwise, this stage continues until it reaches the maximum number of iteration or the MA loss decreases to our expected threshold. In most cases, the network can already learn well the first stage on both clean and perturbed data, thus it will converge very fast in the second stage and do early break without influencing the performance or increasing the training time, as shown in the first row of Table. 6. But we found in some cases, the first stage can make the network learn well on perturbed data, but can not perform on clean data as well as a network trained on only clean data. In this case, adding the second stage can help the network perform better, while doesn't reduce the performance on perturbed data too much. Moreover, our method can perform better than AugMix even without the second stage. See Table. 6 for details.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AMAI↑	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓	MMAI↑	AMAI↑	mCE↓
AugMix on Audi	-8.24	81.89	32.22	55.27	75.49	50.23	41.98	73.06	27.39	77.51
Ours on Audi w/o 2	5.86	94.95	47.78	11.83	**88.42**	60.31	27.18	75.16	**32.91**	**39.04**
Ours on Audi	**5.98**	**97.57**	**48.50**	**10.27**	87.56	**62.38**	**25.80**	**77.22**	32.71	39.14
AugMix on Honda100k	-11.41	63.85	14.08	70.64	68.95	47.69	40.12	**61.68**	16.32	88.36
Ours on Honda100k w/o 2	-7.40	66.69	17.77	58.31	**69.98**	47.43	37.88	58.27	17.64	80.50
Ours on Honda100k	**-2.55**	**67.35**	**19.88**	**53.26**	65.10	**48.60**	**36.94**	51.90	**18.29**	**72.84**

Table 6: Ablation study for the second stage. "w/o 2" stands for "without the second stage training". On Audi dataset, the network is already well trained on both clean and perturbed data, thus the second stage won't make great differences. But on Honda100k dataset, the performance on the clean dataset for the first stage is not well, and adding the second stage can improve the performance on clean data while doesn't influence performance on perturbed data too much. But even without the second stage, our method can perform better than AugMix.

	Scenarios									
	Clean	Single Perturbation			Combined Perturbation			Unseen Perturbation		
Method	AmAPI↑	MmAPI↑	AmAPI↑	mCE↓	MmAPI↑	AmAPI↑	mCE↓	MmAPI↑	AmAPI↑	mCE↓
AugMix	-2.23	10.63	1.54	97.35	3.84	1.12	96.18	3.06	1.95	98.74
Our method	**-1.12**	**16.21**	**3.40**	**95.72**	**7.53**	**4.94**	**94.92**	**5.88**	**2.93**	**96.86**

Table 7: Performance comparison for detection task against the baseline performance [3]. Our method outperforms the AugMix in all cases, with about 1%-3% mAP improvement on average, while reducing the mCE by 1%-2%.

A.7 Performance on Detection

We also test our algorithm on the detection task in autonomous driving. We use the Audi dataset [16] (3D Bounding Boxes) and the Yolov4 network [3] as base settings, and then implement our algorithm based on Yolov4. Table 7 shows that our algorithm also improves the model robustness in most scenarios (about 3%-5% mAP improvement on average), and is consistently better than AugMix (about 1%-3% mAP improvement on average).

A.8 Visualization

We show the saliency map visualization in Fig. 7. Our method can help the network to focus on important areas (e.g., the road in front) instead of random areas on perturbed images.

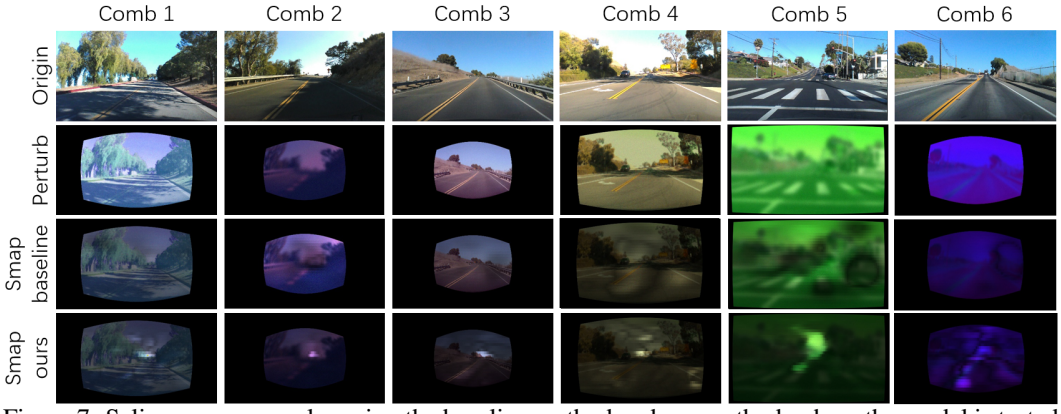

Figure 7: Saliency map samples using the baseline method and our method, where the model is tested on different combinations of perturbations shown as columns. We show the original image, perturbed image with a chosen effect, saliency map of the baseline model, and saliency map of our method from top to bottom rows. Using our method, the network focuses more on the important areas (e.g., road in front) instead of random areas on the perturbed images.

We also show the t-SNE [26] visualization of feature embeddings for the baseline method and our method in Fig. 8. The features from the baseline method are more clustered by color (e.g., the left

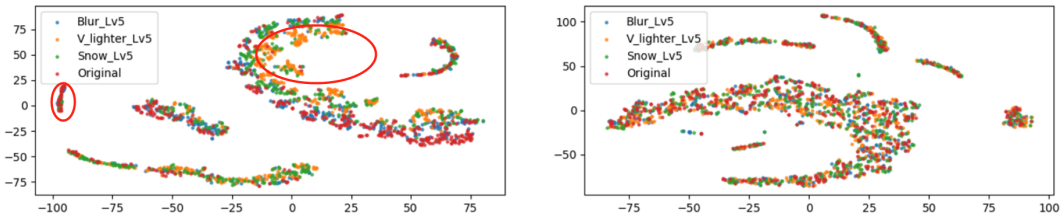

Figure 8: t-SNE [26] visualization for features achieved from networks trained by baseline method (left) and our method (right). The features from the baseline method are more clustered by color (e.g., the left circle in the left image mainly contains red dots, and the right circle in the left image mainly contains yellow dots), indicating there are domain gaps between the perturbed data and original data, while the features from our method are more uniformly distributed, suggesting that our method is able to reduce the domain gaps due to perturbations, i.e., improving the robustness.

circle in the left image mainly contains red dots, and the right circle in the left image mainly contains yellow dots), indicating there are domain gaps between the perturbed data and original data; while the features from our method are more uniformly distributed, suggesting that our method is able to reduce the domain gaps from perturbations, i.e., improve the robustness.

A.9 Experiment data

To quantify our results, we collected mean accuracy (MA) measurements from each experiment, for each pairwise factor and level across methods. Table 8 shows the mean accuracy measurements for blur, noise, and distortion factors. The same is of table 9, where mean accuracy is measured across levels of RGB or HSV color channels, where each channel serves as a single corruption factor. Table 10 presents the MA measurements for scenarios with a combination of factors, and Table 11 presents the MA measurements for scenes with previously unseen factors.

Method	Factor	L1	L2	L3	L4	L5
baseline	blur	88.2	88.1	86.1	81.2	73.3
	noise	88.3	86.0	81.4	76.4	73.2
	distortion	88.6	75.0	57.7	**48.8**	49.2
ours	blur	90.6	90.6	90.3	90.7	88.6
	noise	89.7	89.2	86.1	84.2	79.4
	distortion	90.3	89.8	87.0	**84.2**	83.3

Table 8: Mean Accuracy of training (in %) using the baseline model and ours, tested on datasets with different levels of blur, noise, and distortion. Levels range from L1 to L5. We achieve up to 35.4% in performance gain (see bold number pair).

Method	Factor	DL5	DL4	DL3	DL2	DL1	LL1	LL2	LL3	LL4	LL5
baseline	R	53.2	55.4	57.9	65.1	87.8	87.7	61.4	52.1	47.4	45.1
	G	44.2	48.2	53.5	73.0	88.5	87.9	69.6	51.2	43.7	**40.0**
	B	43.0	46.8	54.3	69.7	88.2	87.7	66.2	52.5	47.1	42.6
	H	51.3	52.1	63.1	82.8	88.1	88.2	69.3	51.5	51.3	51.2
	S	58.4	63.8	72.6	83.9	88.1	88.3	74.5	61.6	56.5	53.2
	V	52.6	53.2	54.6	69.4	88.5	88.4	70.4	49.1	43.2	39.4
ours	R	88.6	90.1	90.6	90.5	90.5	90.4	90.6	90.6	90.2	89.4
	G	90.0	90.6	90.6	90.5	90.5	90.4	90.5	90.6	90.3	**89.9**
	B	89.1	90.0	90.5	90.4	90.3	90.4	90.4	90.6	90.0	89.3
	H	89.7	90.2	90.2	90.4	90.4	90.7	90.5	90.0	89.2	89.7
	S	88.9	90.0	90.4	90.5	90.6	90.6	90.7	90.7	89.7	86.9
	V	87.8	89.5	90.7	90.8	90.7	90.5	90.6	90.0	84.4	76.4

Table 9: Mean accuracy (MA) of training (in %) using the baseline model and ours, tested on datasets with different levels of R, G, B, and H, S, V channel values. DL denotes "darker level", which indicates a level in the darker direction of the channel, while LL indicates "lighter level", which indicates the lighter direction, on levels 1 to 5. We achieve up to **49.9%** in performance gain (see bold number pair).

Method	Comb1	Comb2	Comb3	Comb4	Comb5	Comb6
baseline	59.7	54.0	40.9	**50.0**	54.0	56.3
ours	73.4	68.6	70.9	**83.3**	86.2	65.3

Table 10: Mean accuracy (MA) of training (in %) using the baseline model and ours, tested on datasets with several perturbations combined together, including blur, noise, distortion, RGB, and HSV. We achieve up to **33.3%** in performance gain (see bold number pair).

Method	Unseen Factors	L1	L2	L3	L4	L5
baseline	motion_blur	76.4	69.7	62.6	61.1	60.3
	zoom_blur	85.6	83.7	81.8	80.0	78.2
	pixelate	88.2	88.2	88.0	88.3	88.1
	jpeg_comp	88.4	88.0	87.4	85.4	82.2
	snow	62.8	**50.7**	54.9	55.5	55.3
	frost	55.8	52.1	51.7	51.7	51.2
	fog	58.7	55.0	52.4	50.8	48.1
ours	motion_blur	88.0	86.9	84.7	81.4	78.5
	zoom_blur	88.3	86.9	85.5	84.1	82.5
	pixelate	90.3	90.3	89.9	90.5	90.6
	jpeg_comp	90.1	90.2	90.0	89.5	90.0
	snow	87.1	**83.8**	82.0	77.4	76.7
	frost	86.0	83.9	81.1	81.7	80.1
	fog	77.4	71.6	65.6	61.5	56.1

Table 11: Mean accuracy (MA) of training (in %) using the baseline model and ours, tested on datasets with previously unseen perturbations at 5 different levels. These types of unseen perturbations do not appear in the training data, and include motion blur, zoom blur, pixelate, jpeg compression loss, snow, frost, and fog, on intensity levels L1 to L5. We achieve up to **33.1%** in performance gain (see bold number pair).