Adversarial Differentiable Data Augmentation for Autonomous Systems

Manli Shu[1], Yu Shen[1], Ming C. Lin[1,2] and Tom Goldstein[1]

***Abstract*— Autonomous systems often rely on neural networks to achieve high performance on planning and control problems. Unfortunately, neural networks suffer severely when input images become degraded in ways that are not reflected in the training data. This is particularly problematic for robotic systems like autonomous vehicles (AV) for which reliability is paramount. In this work, we consider robust optimization methods for hardening control systems against image corruptions and other unexpected domain shifts. Recent work on robust optimization for neural nets has been focused largely on combating adversarial attacks. In this work, we borrow ideas from the adversarial training and data augmentation literature to enhance robustness to image corruptions and domain shifts. To this end, we train networks while augmenting image data with a battery of image degradations. Unlike traditional augmentation methods, we choose the parameters for each degradation *adversarially* so as to maximize system performance. By formulating image degradations in a way that is differentiable with respect to degradation parameters, we enable the use of efficient optimization methods (PGD) for choosing worst-case augmentation parameters. We demonstrate the efficacy of this method on the *learning to steer* task for AVs. By adversarially training against image corruptions, we produce networks that are highly robust to image corruptions. We show that the proposed differentiable augmentation schemes result in higher levels of robustness *and* accuracy for a range of settings as compared to baseline and state-of-the-art augmentation methods.**

I. Introduction

Autonomous systems like drones and self-driving cars have the potential to improve the efficiency of transportation, while reducing the risks of accidents. However, it is challenging to develop a reliable and robust autonomous system that can make correct decisions under different adversarial conditions, such as changing ambient lighting during day and night, unpredictable weather conditions, and various hardware uncertainties, including camera distortion and noises. Recently Deep Neural Networks (DNNs) have achieved notable success in autonomous systems. In particular, vision-based autonomous driving offers the advantage of relatively low hardware costs, with cameras on board instead of delicate sensors. These vision-based autonomous driving systems are typically trained in an end-to-end fashion, thereby reducing the overall system complexity.

However, DNN-based vision models are known to be highly brittle to small image corruptions, changes in lighting conditions, and other seemingly small changes in the data distribution [1], [2]. For safety and reliability, it is crucial to build robust DNN-based vision systems that perform well under unforeseen conditions. In this work, we propose to improve the robustness of vision-based models using ideas inspired by the adversarial training literature [3]. We augment the training datasets with common image transformations, like Gaussian blur, noise corruption, and color shifts. Unlike standard augmentation methods, we choose the image transformations and their parameters using adversarial optimization to find the worst-case scenarios during training. By doing so, we produce vision systems that are more robust to image transformations than traditional methods.

In the proposed framework, transformations are applied to each batch of training images. We find the optimal parameters for each transformation that results in worst-case model performance using Projected Gradient Descent (PGD) [3], in which we take the derivatives of the model loss with respect to the parameters that control the data augmentation (i.e. the amount of blur, the level of noise, etc...). Once the worst-case transformation has been identified via gradient ascent on the augmentation parameters, the model is trained on these "adversarial examples" using standard SGD.

With our proposed adversarial data augmentation strategy, we observe a significant performance gain in terms of both improved accuracy on the original dataset and generalization to a broad range of image degradations, including corruptions that are not seen during training. When tested on a set of unseen corrupted images that simulate challenging real-world scenarios, our model shows significant improvement on the *mean Corrupted Error* (mCE) [2] over a baseline model [4]. Compared with the existing work [5] that also adopts a similar idea of adversarial training but in an approximate way that does not exploit differentiable augmentations, we show that our formulation for adversarial training achieves superior results. We demonstrate the efficacy of the adversarial training framework by comparing to a model trained with randomly chosen augmentations. We also provide an ablation study on how the strength of the adversary can affect the behavior of our training strategy.

Main Results: We introduce a differentiable and tunable data augmentation framework for adversarial training of autonomous systems, specifically self-driving cars in this paper, to improve the robustness of computer vision-based models. With this formulation, we can substantially improve model performance and robustness against challenging real-world conditions that pose threats to safety and reliability of autonomous vehicles. In principle, the method is quite general, and it can be applied to a wide range of vision-based tasks, model architectures, and datasets.

The authors are with [1]Department of Computer Science and [2]Maryland Robotics Center, University of Maryland at College Park, MD, U.S.A. {manlis,yushen,lin,tomg}@umd.edu

II. Related Works

The work presented in this paper builds on existing needs to enhance the robustness of vision systems, and draws inspiration from the data augmentation and adversarial training literature. We discuss these connections here.

Robustness in Autonomous Driving. The robustness of autonomous systems is a topic of interest and great importance. Vision-based models rely heavily on training data, which can be problematic in the task of autonomous driving: training data are usually high quality images collected under good weather conditions; models trained on this may fail to generalize well to various real-world situations [6], [7]. For this reason, [8] collects a new dataset that contains complicated real-world traffic conditions. [9], [10] propose to enhance the robustness of autonomous systems by training with synthetic data that models different weather conditions. Instead of modeling any specific real-world situations, or collecting new datasets. We exploit common image-quality factors that affect the decision-making process of vision-based models. Our work studies a regression problem in the field of autonomous driving , *learning to steer* [11], [12]. We propose to improve the robustness of autonomous systems by introducing the above mentioned factors to our model through data augmentation at the training time, incorporating adversarial training for enhanced robustness.

Data Augmentation. Data augmentation is a widely-adopted technique to prevent machine learning models from overfitting. Commonly used augmentation methods rely on basic image processing operations like random scaling, random cropping, etc. More complex data augmentation techniques have been studied in recent works. Unlike traditional data augmentations that consider each image separately, [13], [14] train on random pair-wise combinations of images. [15] trains using random averages of images after augmentations are applied, resulting in higher diversity in the augmented data. Rather than designing new augmentation operations, another direction is to find the best choices and order of operations for basic augmentation operations, i.e., an optimal data augmentation policy, for a given learning task. [16] aims to find the optimal augmentation policy by solving a discrete search problem through reinforcement learning, whereas [17] proposes to reduce the cost of the searching procedure by relaxing this discrete optimization problem into a differentiable one. In our work, instead of finding the optimal augmentation policy for a learning task, we aim to find the optimal setting of each basic augmentation operation for each batch of input data using adversarial optimization. With all of our operations made differentiable, we can solve the optimization at relatively low cost.

Adversarial Training. Adversarial training and its variants [3], [18] were first proposed to defend against adversarial attacks, where adversarially crafted input data with imperceptible perturbation, i.e., adversarial examples [1], [19], are used to cause the malfunction of machine learning models. In order to improve the adversarial robustness of a model, adversarial training solves a min-max optimization problem, in which the inner maximization perturbs the input data to cause maximal loss of the model, while the outer minimization problem updates the model so as to increase performance on the perturbed input.

Recent works have shown that adversarial training can be effectively applied to tasks other than defending against adversarial attacks. [20] use adversarial training to improve performance on non-adversarial data by disentangling the feature statistics of the two during training. [21] improves the generalization ability of a model through adversarial training, where the adversary perturbs the midway feature representations inside the model rather than the input data. [5] proposes an approximate adversarial training for data augmentation by selecting the "worst" augmentation among a group of randomly generated augmentations, and use the selected augmentation for training the model. We implement a differentiable framework where gradient feedback from the downstream task can be obtained to direct the search for the "worst" augmentation.

III. Background

Vision-based Learning to Steer. The vision-based *learning to steer* task in autonomous driving aims to predict the ideal steering angles of the autonomous vehicle (AV) from vision-based inputs, i.e., one or more images, which are captured by the camera device(s) installed on board of the AV. Usually the input images are photographs from road scenes [11], but other image formats are also possible like event camera data [22]. The task can be solved using different learning methods, e.g., through Convolutional Neural Networks (CNN) [4], or through Reinforcement Learning [12]. The proposed approach in this paper is agnostic to the learning methods used and it can be applied on the preparation of training data for any AV and/or computing platform.

Robustness and Image Quality Attribute Factors. Robustness in autonomous driving refers to how stable the autonomous system is under changes to input data. In vision-based learning to steer task, when the input image quality of the autonomous driving system is degraded by certain environment factors, e.g., bad weather condition like snow/fog, or changes in lightening conditions that occur due to transitions from day to night or from cloud cover, the robustness of the system is critical to ensure the AV drives safely. In this paper we propose to train the networks by augmenting training data with adversarial image degradations. We choose eight common attributes that affect the image quality in the training data for autonomous driving. They are Gaussian blur, Gaussian noise, red/green/blue color balance (3 channels in RGB space), hues/saturation/intensity values (3 channels in HSV space) – together they capture many complicated or even unseen factors in the scenes. Gaussian blur and noise are two of the most commonly seen image-level perturbations, while RGB and HSV space are among the most widely used color model for image representation. We adopt the metric of *mean corrupted error* (mCE) from [2] to evaluate robustness, as will be specified in section V-D.

IV. Method

A. Overview

Our method features two major components: a differentiable data augmentation framework, and the adversarial training procedure that tunes the augmentation parameters to improve the robustness of a model against image corruptions. An overview diagram of the overall algorithmic framework is shown in Figure 1. With each batch, we do adversarial training for each augmentation operation one at a time. The backbone network is used for both the adversarial process and learning process. Only the training images are updated during the adversarial process, and only the backbone parameters are updated during the learning process.

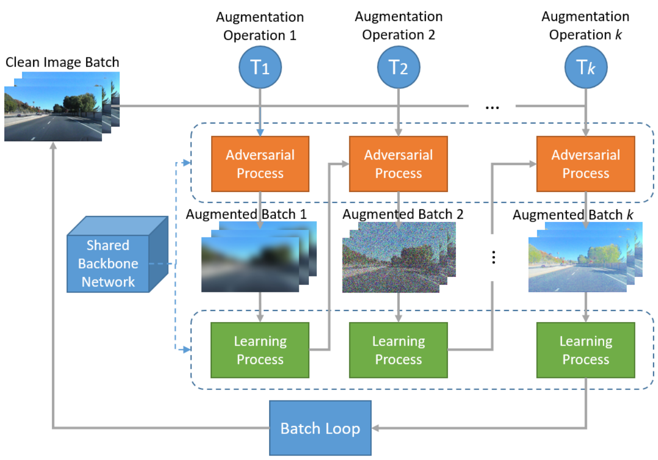

Fig. 1. **System Pipeline of Our Method:** With each batch, we do adversarial training for each augmentation operation one at a time. The backbone network is used for both the adversarial process and learning process. Only the training images are updated during the adversarial process, and only the backbone parameters are updated during the learning process.

B. Differentiable Data Augmentation

We first propose a differentiable data augmentation framework, where augmentation operations are differentiable w.r.t associated parameters. Let T_δ denote an augmentation operations in our framework parameterized by δ, given input data x, this framework outputs augmented data $\hat{x} = T_\delta(x)$, where δ controls the strength of operation T.

After feeding $\hat{x}$ into a downstream DNN model and doing a forward and backward propagation within the model, we can get an upstream gradient for the input of the model, i.e., $\nabla_{\hat{x}} J$, where J is the objective of the downstream model. Thus we can tune our augmentation operation by taking the derivative of J w.r.t. δ, which can be easily obtained using the chain rule given that the operation T_δ in our framework is differentiable:

$$\nabla_\delta J = \nabla_\delta \hat{x}_i \nabla_{\hat{x}_i} J, \tag{1}$$

We consider a set of transformations drawn from the traditional image processing literature. Specifically, our augmentation set consists of 8 operations: Gaussian blur, Gaussian noise, shifts in the R/G/B channel, and shifts in the H/S/V channel.

Gaussian Blur. We tune the Gaussian blur operation through a parameter δ controlling the standard deviation of the Gaussian kernel by $\sigma = 1 + \delta$, in which δ can be negative or positive centering around 0. To make this operation differentiable with respect to δ, we allocate a 21×21 array to hold the Gaussian kernel. We then populate this array using the analytical formula for a 2D Gaussian with radius $1 + \delta$, i.e., we evaluate the formula

$$f(z) = e^{-\|z\|^2/2(1+\delta)^2} \tag{2}$$

on the 21×21 grid of integer coordinates with the origin at the center, and then normalize the array to sum to 1. We can then perform convolution with this kernel, and then use automatic differentiation to obtain the derivative of the downstream loss with respect to δ.

Gaussian Noise. The Gaussian noise augmentation in our framework consists of two parts: a random array n of noise sampled from a standard Gaussian and a scalar multiplier δ being used to tune the magnitude of the noise.

$$n \sim \mathcal{N}(0, 1) \tag{3}$$

$$T_\delta(x) = clip(x + \delta \cdot n, 0, 1), \tag{4}$$

where the function $clip(x, a, b)$ clips the value of x into range $[a, b]$. This is to make sure that the value of each pixel in the transformed image falls in the valid range, which is $[0, 1]$ in our framework. Note that for each instance of Gaussian noise augmentation, the base noise n is fixed, but it will be resampled for each instance, i.e., each time this augmentation is applied.

Color Systems. We perturb image colors by considering 6 channels, consisting of solid color representation in RGB space and the hue, saturation and value in HSV channels. We can simply perturb values of a single channel to change the perception of colors in an image. All 6 types of color shift augmentations can be tuned in the same manner where parameter δ functions as a scalar to a chosen channel (out of 6 possible) and controls its magnitude. Equation 5 gives an example of shifting the value in the first channel of an image $x \in \mathbb{R}^{H \times W \times 3}$, $x_{i,j,k} \in [0, 1]$.

$$T_\delta(x) = clip\left(\begin{pmatrix} 1+\delta \\ 1 \\ 1 \end{pmatrix} \circ \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ x^{(3)} \end{pmatrix}, \ 0, \ 1 \right), \tag{5}$$

where $x^{(i)} \in \mathbb{R}^{H \times W}$ is the i-th channel of the image ($i = 1, 2, 3$) in either RGB or HSV format, and $\circ$ denotes element-wise matrix multiplication.

C. Adversarial Training

With the aforementioned differentiable data augmentation framework, we can apply PGD adversarial training to tune these augmentation operations "adversarially". The goal of tuning the data augmentation is to degrade the performance of the downstream model, which can be formulated as a min-max optimization problem:

$$\min_\theta \mathbb{E}_{(X,y)\sim\mathcal{D}} \left[\max_{\|\delta\|_p \leq \epsilon} \mathcal{L}(f_\theta(T_\delta(X)), y) \right], \tag{6}$$

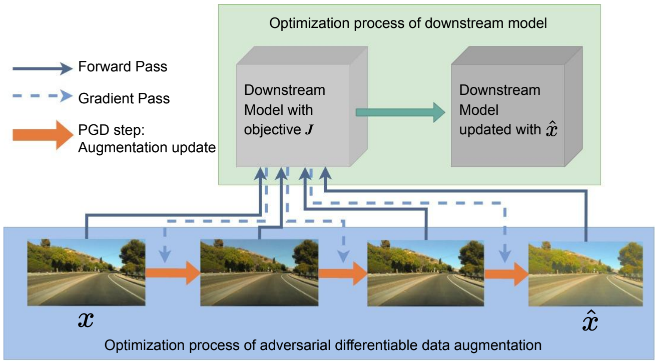

Fig. 2. **Optimization process of differentiable adversarial data augmentation** and the model in our system. The figure shows one augmentation operation (value shift in a color channel) as an example. For each data batch, the gradient of the data augmentation is used to perform PGD steps, and a batch of images is crafted that maximize model error. Then, the adversarial images are used to update model parameters.

where (X, y) is the (image, label) pair drawn from distribution $\mathcal{D}$, f_θ is the model with parameter θ, and T_δ is our differentiable data augmentation module parameterized by δ. $\mathcal{L}$ denotes the objective function, which is the mean square error loss in our task. We solve the inner maximization problem by running projected gradient descent (PGD) [3], where the ℓ_p norm of the perturbation (i.e., δ) will be bounded by ϵ. The outer minimization is solved by running gradient descent. Fig. 2 provides an illustration of our optimization process.

We summarize a formal description for adversarial training with differentiable data augmentation in Algorithm 1.

Algorithm 1: Adversarial Differentiable Data Augmentation

Input: Training data, model f_θ, PGD parameters: $\{$bound ϵ, step size α, repeats m$\}$, data augmentation Ops: $\mathcal{T}_\delta = \{T_\delta^1, \ldots, T_\delta^k\}$, Loss function: $\mathcal{L}$;

Output: Updated model parameters: θ;

for *each training step* **do**
  Sample mini-batch x with label y;
  for *augmentation op T =* $T^1, \ldots, T^k$ **do**
    Initialize augmentation parameter: $\delta = 0$;
    for *adversarial step = 1, …, m* **do**
      $\delta \leftarrow \delta + \alpha \cdot sign(\nabla_\delta \mathcal{L}(f_\theta(T_\delta(x)), y))$;
      $\delta \leftarrow \text{clip}(\delta, -\epsilon, \epsilon)$;
    end
    $\hat{x} = T_\delta(x)$;
    Minimize the total loss w.r.t. model parameter:
    $\theta \leftarrow \arg\min_\theta \mathcal{L}(f_\theta(\hat{x}), y)$;
  end
end

V. EXPERIMENTS

We validate our method on the vision-based learning to steer task. System inputs are images (often photos taken of the road scene) and the output is a steering angle for the AV.

A. Datasets

We train and evaluate our models on three different real-world driving datasets: SullyChen [23], HDD [24], and A2D2 dataset [25], all of which are collected under good weather conditions during the daytime. In each dataset, images are extracted from videos at certain "frames per second" (FPS), associated with their steering angles as the label information. For training efficiency, we adopt a similar approach in [4] by using low FPSs to reduce similarities between adjacent frames. Specifically, we use 15 FPS for A2D2, 5 FPS for SullyChen, and 1 FPS for HDD dataset. For each dataset, we use approximately 10,000 images for training, and 1,000 for testing.

B. Models

Model Architecture. We use the network architecture from [4], which is known to be effective in both real [4] and virtual [26] domains, and extensible to transfer learning on "sim-to-real" [27]. The model is a 5-layer convolutional network with 3 fully connected layers, which takes single images as input, and outputs steering angle predictions.

Implementation Details. We use the Adam [28] optimizer with learning rate 1×10^{-4} and batch size 128. The baseline model is trained for 1,000 epochs. The differentiable data augmentation module in our method is implemented in PyTorch [29], which supports automatic differentiation for our chosen augmentation operations. Following the preprocessing for the baseline model in [4], input images are rescaled to 66×200 resolution and converted to YUV format.

Based on the architecture and hyper-parameter settings described above, we implemented 4 different methods including the naive baseline, and two alternatives that are conceptually related to ours for ablation study, as specified below:

- **Baseline.** Our baseline model is trained without any data augmentation.
- **Ours.** We fix the adversarial step-size as $\alpha = 0.2$, and for a m-step PGD attack, we set the perturbation bound to be $\epsilon = \alpha \cdot m - 0.1$. We use $m = 3, 4, 5$ for A2D2, SullyChen and HDD dataset respectively.
- **Random Augment.** This approach follows the same training process as ours, except that the PGD adversarial step is replaced with a random sampling of δ. The distribution from which we draw δ is a uniform distribution with range $[-\epsilon, \epsilon]$, i.e., the PGD bound in the adversarial setting.
- **MaxUp.** [5] approximates adversarial steps by selecting the worst augmentation from N randomly generated candidates directly without gradient feedback. Our MaxUp implementation generates random candidate augmentations from the 8 operations introduced in Section IV-B, with δ randomly sampled from a uniform distribution with range $[-\epsilon, \epsilon]$. MaxUp models are trained for 9,000 epochs to match the number of parameter updates in our method. We follow the training policy in [5] by not running the adversarial step in the first 5 epochs, and using $N = 4$ for selecting candidates.

Fig. 3. **Example images from "single factors" test sets**. Each column corresponds to a factor. For the left 2 columns, corruption severity increases from top to bottom. In the 6 rightmost columns, channel values range from decreased (top) to increased (bottom).

C. Test Scenarios

When evaluating a model's performance on a dataset, in addition the test set sampled from the original dataset, we generate a range of variants of this test set, each one featuring a challenging domain for an autonomous system.

We classify testing scenarios into four categories for a systematic evaluation, with the first scenario being the original test data with no domain shift, denoted as "Original Data" in Table I. The second scenario, "Single Factors", consists of the 8 transformations we use for data augmentation (Gaussian blur, Gaussian noise, R/G/B shift, and H/S/V shift), applied one-at-a-time to create 8 categories of test sets under this scenario. For each category, we apply the transformation with 5 different severity levels (for the 6 color shift transformations, 5 severity levels are applied at both negative and positive directions), creating a total of 70 test sets. See Fig. 3 for sample images from these test sets.

In the third scenario, we consider "multi-factor" augmentations that combine all the 8 transformations in a random manner. For each multi-factor test set, we randomly sample a parameter vector that controls each of the augmentation operations, and sample a random permutation that decide the order for these operations. The parameter vector we use here is drawn from a normal distribution with $\sigma = 0.33$ and $\mu = 0$. Model performances are averaged over 25 combinations to reduce the randomness.

In the final testing scenario, we evaluate model performance on more complicated image corruptions that models have not seen during training, denoted as "unseen factors." We include 8 image corruptions, simulating real-world situations that autonomous system may encounter due to hardware or weather conditions: "radial distortion", "zoom blur", "motion blur", "jpeg compression", "pixelate", "snow", "frost", "fog." For each factor, we generate 5 test sets with increasing severity levels. This set of corruptions was originally proposed for benchmarking the robustness of ImageNet classifiers in [2]. See Fig. 4 for visual effects of each unseen factors.

D. Evaluation Metrics

Mean Accuracy. To measure the regression accuracy, we use *mean Accuracy* (mAcc), similar to the *mean Average Precision* (mAP) used in classification tasks [30] [31], which evaluates the performance with different thresholds.

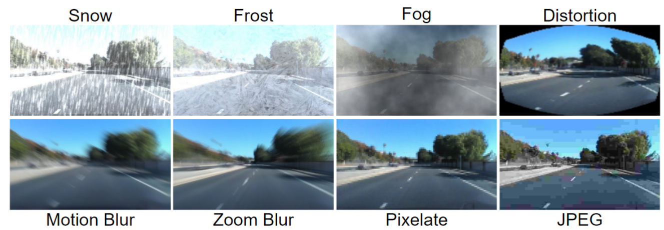

Fig. 4. **Example images for 8 "unseen" quality corruptions** that simulate real-word weather challenges and hardware-related artifacts.

The accuracy for a threshold τ is defined as:

$$\text{acc}_\tau = \frac{1}{n}\text{count}(|v_{pred} - v_{gt}| < \tau), \tag{7}$$

where v_{pred} and v_{gt} are the predicted and ground-truth value respectively, and n denotes the number of test cases. The mAcc is computed as $\frac{1}{|\mathcal{T}|}\sum_\tau acc_{\tau \in \mathcal{T}}$, where $\mathcal{T} = \{1.5, 3.0, 7.5, 15, 30, 75\}$ contains empirically selected thresholds of steering angles.

Mean Corrupted Error. To evaluate the domain generalization ability of a model, we adopt the metric of *mean corrupted error* (mCE) from [2], where we compute a weighted average of error rates among a range of test data, each of which features a set of image corruptions. The weights used for averaging come from the performance of a baseline model: denote $Err_A^{C_i}$ as the error rate of model A on a test set of corruption type c_i. Then the mean Corrupted Error of model A among test sets $\mathcal{C} = \{c_1, c_2, \cdot, c_m\}$ will be

$$\text{mCE}_A = \frac{1}{m}\sum_i^m \frac{\text{Err}_A^{c_i}}{\text{Err}_{baseline}^{c_i}}, \tag{8}$$

where $\text{Err}_A^{c_i} = 1.0 - mAcc_A^{C_i}$, which is derived from the mean accuracy of model A on the test set c_i. This weighted average balances different corruption types in alignment with their difficulties as determined by a baseline model, and reflects the overall generalization ability of a model.

E. Comparing Results

Results in Table I show that our proposed method can improve the performance of the baseline model under all test scenarios and on all three datasets. By comparing to the random augmentation, we prove the advantage of our adversarial training strategy. We notice that even though MaxUp also achieves non-trivial improvements over the baseline, it does not compete well with other methods. We believe the

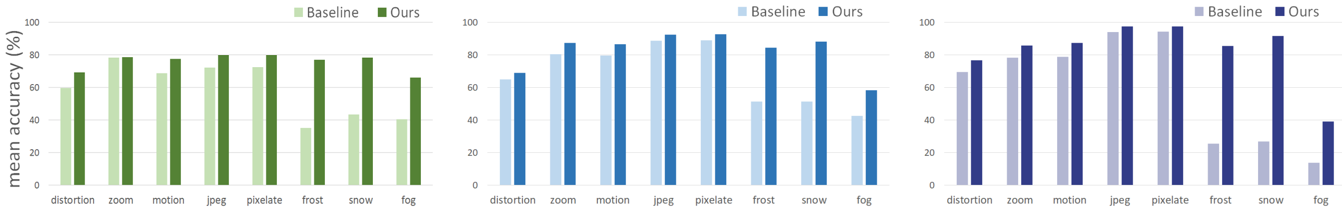

Fig. 5. **Detailed improvement on each category of the unseen factors**. Presented results are mean accuracies (mAcc) on HDD[24] (left), SullyChen [23] (center), and A2D2 [25] (right) test data. Our model consistently outperforms the baseline method on all 3 datasets with previously unseen factors.

TABLE I

EVALUATION OF ROBUSTNESS OF OUR VS. OTHER MODELS.

Performance of ours vs. other models under different scenarios with single- and multi-factor image quality corruption, and complex ones that simulate real-word weather challenges and hardware-related artifacts. Ours outperforms other alternatives on 3 different driving datasets [23], [24], [25] on the original test data and augmented data with image corruption due to single, multiple, or previously unseen factors.

	Original Data	Single Factors	Multi Factors	Unseen Factors
Models	mAcc(%)↑	mCE(%)↓	mCE(%)↓	mCE(%)↓
Baseline	72.25	100	100	100
MaxUp [5]	79.04	70.78	72.51	73.02
Random augment	78.81	61.96	54.75	63.49
Ours	**79.91**	**60.00**	**52.05**	**61.73**

(a) Results on HDD dataset [24].

	Original Data	Single Factors	Multi Factors	Unseen Factors
Models	mAcc(%)↑	mCE(%)↓	mCE(%)↓	mCE(%)↓
Baseline	89.12	100	100	100
MaxUp [5]	89.05	58.18	53.44	84.23
Random augment	91.04	38.55	24.39	65.51
Ours	**92.38**	**34.11**	**22.35**	**61.67**

(b) Results on SullyChen dataset [23].

	Original Data	Single Factors	Multi Factors	Unseen Factors
Models	mAcc(%)↑	mCE(%)↓	mCE(%)↓	mCE(%)↓
Baseline	95.17	100	100	100
MaxUp [5]	97.42	38.26	43.42	69.68
Random augment	96.98	19.94	13.40	57.31
Ours	**97.55**	**14.80**	**9.12**	**48.36**

(c) Results on A2D2 dataset [25].

selection procedure in MaxUp can be systematically biased toward augmentations that are inherently more deleterious to performance, while models trained on such imbalanced data cannot generalize well to our diverse test scenarios.

In Fig. 5, we break down the evaluation for the test scenario of "unseen factors". Our method constantly improve the performance on each of the unseen factor, especially on weather-related image quality degradation, where baseline models are shown to be significantly vulnerable.

F. Ablation Study

In this section, we study how adversarial strength impacts the performance of our method on the SullyChen dataset. We evaluate performance of our models trained with different levels of "adversarial strength", controlled by the number of PGD adversarial steps m, as we fix the step size to be $\alpha = 0.2$ and set the perturbation bound $\epsilon = \alpha \cdot m - 0.1$.

As shown in Figure 6, when a model is trained with small perturbations, the improvement decreases under all test scenarios. This drop may be due to the lack of diversity in the data augmentation, as small perturbations cause augmented data to be near the original one. Models trained with large perturbation also show smaller performance gain. This is expected because training data becomes extremely noisy and distorted when the adversarial strength is excessively strong.

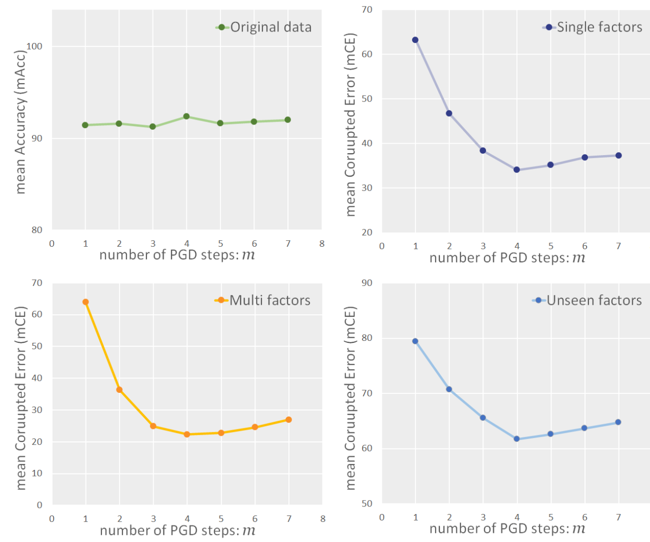

Fig. 6. **Ablation study on the effect of PGD adversarial strength in our method**, evaluated on SullyChen [23] test data under all test scenarios.

VI. CONCLUSION

This work focuses on the reliability of machine learning models for robotic systems like autonomous vehicles. Our methods improve the robustness of models by training with data augmentation, where each operation is optimized adversarially in a differentiable framework. To evaluate the performance of our framework, we consider different scenarios that cover a wide range of image degradations, including simulation of real-world situations related to autonomous systems. Models trained with our method demonstrate consistent robustness and outperform other recent works under various testing scenarios, including previously unseen factors (e.g. those from poor weather conditions and camera artifacts).

Note that our method functions as a base framework for training neural networks and it can be combined with various data augmentation methods. Although we have only focused on the learning to steer task for AVs in this paper because of its relevance to robotics, this method is generalizable and can be applied to a wide range of vision-based tasks, model architectures, and datasets that we hope to further investigate.

Acknowledgements: Shu and Goldstein were supported by the ONR MURI program. Additional support was provided by DARPA GARD.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in *International Conference on Learning Representations, ICLR*, 2014.

[2] D. Hendrycks and T. G. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” in *International Conference on Learning Representations, ICLR*, 2019.

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *International Conference on Learning Representations, ICLR*, 2018.

[4] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016.

[5] C. Gong, T. Ren, M. Ye, and Q. Liu, “Maxup: A simple way to improve generalization of neural network training,” 2020.

[6] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, “Benchmarking robustness in object detection: Autonomous driving when winter is coming,” *arXiv preprint arXiv:1907.07484*, 2019.

[7] W. Zhou, J. S. Berrio, S. Worrall, and E. Nebot, “Automated evaluation of semantic segmentation robustness for autonomous driving,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 1951–1963, 2019.

[8] Z. Che, M. G. Li, T. Li, B. Jiang, X. Shi, X. Zhang, Y. Lu, G. Wu, Y. Liu, and J. Ye, “D^2-city: A large-scale dashcam video dataset of diverse traffic scenarios,” *CoRR*, vol. abs/1904.01975, 2019.

[9] C. Sakaridis, D. Dai, S. Hecker, and L. V. Gool, “Model adaptation with synthetic and real data for semantic dense foggy scene understanding,” in *Computer Vision - ECCV*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018, pp. 707–724.

[10] G. Volk, S. Müller, A. von Bernuth, D. Hospach, and O. Bringmann, “Towards robust cnn-based object detection through augmentation with synthetic rain variations,” in *2019 IEEE Intelligent Transportation Systems Conference, ITSC*, 2019.

[11] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, “Learning robust control policies for end-to-end autonomous driving from data-driven simulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[12] K. Wu, M. Abolfazli Esfahani, S. Yuan, and H. Wang, “Learn to steer through deep reinforcement learning,” *Sensors*, vol. 18, no. 11, p. 3650, 2018.

[13] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” in *International Conference on Learning Representations, ICLR*, 2018.

[14] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in *ICCV*, 2019.

[15] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “Augmix: A simple data processing method to improve robustness and uncertainty,” in *International Conference on Learning Representations, ICLR*, 2020.

[16] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, “Autoaugment: Learning augmentation strategies from data,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

[17] Y. Li, G. Hu, Y. Wang, T. M. Hospedales, N. M. Robertson, and Y. Yang, “DADA: differentiable automatic data augmentation,” *CoRR*, 2020.

[18] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. P. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, “Adversarial training for free!” in *NeurIPS*, 2019.

[19] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: A simple and accurate method to fool deep neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

[20] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, “Adversarial examples improve image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2020.

[21] M. Shu, Z. Wu, M. Goldblum, and T. Goldstein, “Prepare for the worst: Generalizing across domain shifts with adversarial batch normalization,” 2020.

[22] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, “Event-based vision meets deep learning on steering prediction for self-driving cars,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.

[23] S. Chen, “A collection of labeled car driving datasets, https://github.com/sullychen/driving-datasets,” 2018.

[24] V. Ramanishka, Y. Chen, T. Misu, and K. Saenko, “Toward driving scene understanding: A dataset for learning driver behavior and causal reasoning,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2018, pp. 7699–7707.

[25] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, *et al.*, “A2d2: Audi autonomous driving dataset,” *arXiv preprint arXiv:2004.06320*, 2020.

[26] W. Li, D. Wolinski, and M. C. Lin, “ADAPS: Autonomous driving via principled simulations,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7625–7631.

[27] S. Akhauri, L. Zheng, and M. Lin, “Enhanced transfer learning for autonomous driving with systematic accident simulation,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[28] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[30] K. Li, Z. Huang, Y.-C. Cheng, and C.-H. Lee, “A maximal figure-of-merit learning approach to maximizing mean average precision with deep neural network based classifiers,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4503–4507.

[31] P. Henderson and V. Ferrari, “End-to-end training of object class detectors for mean average precision,” in *Asian Conference on Computer Vision*. Springer, 2016, pp. 198–213.