DispBench: Benchmarking Disparity Estimation to Synthetic Corruptions

Anonymous CVPR submission

Paper ID 0006

Abstract

Deep learning (DL) has surpassed human performance on standard benchmarks, driving its widespread adoption in computer vision tasks. One such task is disparity estimation, estimating the disparity between matching pixels in stereo image pairs, which is crucial for safety-critical applications like medical surgeries and autonomous navigation. However, DL-based disparity estimation methods are highly susceptible to distribution shifts and adversarial attacks, raising concerns about their reliability and generalization. Despite these concerns, a standardized benchmark for evaluating the robustness of disparity estimation methods remains absent, hindering progress in the field.

To address this gap, we introduce DISPBENCH*, a comprehensive benchmarking tool for systematically assessing the reliability of disparity estimation methods.* DISPBENCH *evaluates robustness against synthetic image corruptions such as adversarial attacks and out-of-distribution shifts caused by 2D Common Corruptions across multiple datasets and diverse corruption scenarios. We conduct the most extensive performance and robustness analysis of disparity estimation methods to date, uncovering key correlations between accuracy, reliability, and generalization. Upon acceptance,* DISPBENCH *will be open-sourced to facilitate further research in this direction.*

1. Background

The vision task of disparity estimation, also commonly known as stereo-matching is used to estimate the disparity between matching pixels in stereo image pairs. Mayer et al. [29] proposed the first Deep Learning (DL) based method for disparity estimation called DispNet. This led to disparity estimation becoming primarily a DL-based task [16, 27, 39]. However, DL-based methods are known to be unreliable [13, 33], they tend to learn shortcuts rather than meaningful feature representations [14] and can be easily deteriorated even by small perturbations, causing the evaluation samples to not be independent and identically distributed (i.i.d.) w.r.t. the training samples. This shift from i.i.d. samples can be caused due to changes in the environment, changes in weather conditions, or image corruption due to sensor noise [21]. Such shifts cause the evaluations to be Out-Of-Distribution (OOD), and robustness to such shifts is called OOD Robustness. OOD Robustness is often used as a metric for the generalization ability of a method [22, 23]. Another possible cause of distribution shifts could be either accidental or malicious adversarial attacks. Here, the perturbations made to an image are optimized to fool the method while the semantic meaning of the images remains the same for a human observer. When adversarial attacks are optimized with full information about a model and its loss, they are called white-box adversarial attacks. Since these white-box attacks can potentially simulate the worst-case scenario for a method, they are often used as a proxy for measuring their reliability [1, 15, 25].

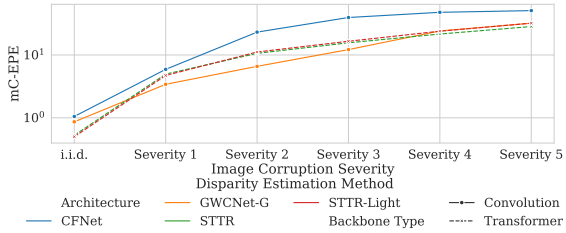

Figure 1. Analyzing the generalization ability of some Disparity estimation methods: GWCNet [16], CFNet [39], and STTR and STTR-light [27] proposed over time. The y-axis represents the mean End-Point-Error (EPE) on Syntheticc Corruptions (2D Common Corrruptions [21]) at different severalties (severity=0 is i.i.d. performance) using the FlyingThings3D [29], i.e., lower is better. We observe that disparity estimation methods lack the generalization ability to common corruptions and, thus, are not safe for real-world deployment.

In Fig. 1, we provide an overview of the i.i.d. performance, generalization ability, and reliability of disparity estimation methods proposed over time on the FlyingThings3D dataset [29]. We include old popular methods such as GWCNet and CFNet and new large transformer-based STTR and its lightweight version STTR-light, which, due to its training regime, are proposed as zero-shot disparity

estimation methods. Here, we observe a disturbing pattern: while the i.i.d. performance has improved over time, since this improvement has been the focus of most works, the models still lack robustness. This is particularly concerning as disparity estimation is often used in the real world, especially for safety-critical scenarios such as medical surgery procedures [34, 42], including invasive surgeries such as laparoscopy [32] and in autonomous driving [6]. Here, safety is paramount, and to ensure the safe deployment of recent DL-based disparity estimation methods, their reliability and generalization ability need to be guaranteed. However, no such guarantees can be provided currently since no works focus on OOD and the adversarial robustness of disparity estimation methods. This is primarily due to a lack of datasets that enable such studies. Capturing corruptions in the wild and then annotating for disparity estimation is a time and resource intensive process.

Some prior works have focused on other kinds of robustness; for example, a recent work [43] looks into the robustness of disparity estimation works to domain shifts, while [28, 44] studies the robustness of methods to occlusions. Currently, there exists no unified framework to evaluate disparity estimation methods for safe deployment in the real world. Guo et al. [17] recently proposed a benchmarking tool for disparity estimation methods. However, this tool is limited to i.i.d. performance evaluations. This is a significant limitation impeding the community's ability to ensure safe, reliable, and generalizable DL-based disparity estimation methods for the real world.

To bridge this gap, we propose DISPBENCH, the first robustness benchmarking tool for disparity estimation. DISPBENCH is easy to use and extending it to future disparity estimation methods and datasets, when they are proposed, is straightforward. It is inspired by similar popular benchmarks for the image classification tasks [8, 40] and object detection [4, 9, 10, 20, 31]. It enables i.i.d. evaluations of various DL-based disparity estimation methods across multiple commonly used disparity estimation datasets. It also facilitates research in the reliability and generalization ability of disparity estimation methods, as it enables users to use synthetic image corruptions, specifically, 5 diverse adversarial attacks and 15 established common corruptions. This will help researchers build better models that are not limited to improved performance on identical and independently distributed (i.i.d.) samples and are less vulnerable to adversarial attacks while generalizing better to image corruptions. Our proposed DISPBENCH facilitates this, streamlining it for future research to utilize.

The main contributions of this work are as follows:

- We provide a benchmarking tool DISPBENCH to evaluate the performance of most DL-based disparity estimation methods over 2 different datasets and synthetic corruptions.
- We benchmark the aforementioned models against commonly used adversarial attacks and common corruptions that can be easily queried using DISPBENCH.
- We perform an in-depth analysis using DISPBENCH and present interesting findings showing methods that perform well on i.i.d. are remarkably less reliable and generalize worse than other non-well-performing methods.
- We show that synthetic corruptions on synthetic datasets do not represent real-world corruptions; thus, synthetic corruptions on real-world datasets are required.

2. DISPBENCH Usage

There exists no standardized tool for evaluating the performance of disparity estimation methods. Thus, the codebase for such a tool had to be written from scratch. In the following, we describe the benchmarking tool, DISPBENCH. Currently, it supports 4 unique architectures (new architectures to be added to DISPBENCH with time) and 2 distinct datasets, namely FlyingThings3D [29] and KITTI2015 [30] (please refer Appendix B for additional details on the datasets). It enables training and evaluations on all aforementioned datasets, including evaluations using SotA adversarial attacks such as CosPGD [1] and other commonly used adversarial attacks like BIM [26], PGD [25], FGSM [15], under various Lipshitz (l_p) norm bounds and APGD [41] under the ℓ_∞-norm bound. Additionally, it enables evaluations for Out-of-Distribution (OOD) robustness by corrupting the inference samples using 2D Common Corruptions [21].

We follow the nomenclature set by RobustBench [8] and use "threat_model" to define the kind of evaluation to be performed. When "threat_model" is defined to be "None", the evaluation is performed on unperturbed and unaltered images, if the "threat_model" is defined to be an adversarial attack, for example "PGD", "CosPGD" or "BIM", then DISPBENCH performs an adversarial attack using the user-defined parameters. Whereas, if "threat_model" is defined to be "2DCommonCorruptions", the DISPBENCH performs evaluations after perturbing the images with 2D Common Corruptions. If the queried evaluation already exists in the benchmark provided by this work, then DISPBENCH simply retrieves the evaluations, thus saving computation. Please refer to Appendix D for details on usage.

Following, we show the basic commands to use DISPBENCH. We describe each attack and common corruption supported by DISPBENCH in detail in Appendix D. Please refer to Appendix F for details on the arguments.

2.1. Model Zoo

It is challenging to find all checkpoints, whereas training them is time and compute-exhaustive. Thus, we gather available model checkpoints made available online by the respective authors. The trained checkpoints for all models

available in DISPBENCH can be obtained using the following lines of code:

```
from dispbench.evals import load_model
model = load_model(model_name='STTR',
    dataset='KITTI2015')
```

Each model checkpoint can be retrieved with the pair of 'model_name', the name of the model, and 'dataset', the dataset for which the checkpoint was last fine-tuned.

2.2. Adversarial Attacks

To evaluate a model for a given dataset on an attack, the following lines of code are required.

```
from dispcbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015'
  ↪  retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model, for example, when the threat model is a PGD attack, 'config.yml' could contain 'threat_model=*"PGD"*', 'iterations=*20*', 'alpha=*0.01*', 'epsilon=*8*', and 'lp_norm=*"Linf"*'. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False' then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the adversarial attack to be used, given as a string.
  - 'iterations' are the number of attack iterations, given as an integer.
  - 'epsilon' is the permissible perturbation budget ϵ given a floating point (float).
  - 'alpha' is the step size of the attack, α, given as a floating point (float).
  - 'lp_norm' is the Lipschitz continuity norm (l_p-norm) to be used for bounding the perturbation, possible options are 'Linf' and 'L2' given as a string.
  - 'target' is false by default, but to do targeted attacks, either the user can set 'target'=True, to use the default target of $\overrightarrow{0}$, or can pass a specific tensor to be used as the target.

The adversarial attacks supported by DISPBENCH are *FGSM, BIM, PGD, APGD, and CosPGD*.

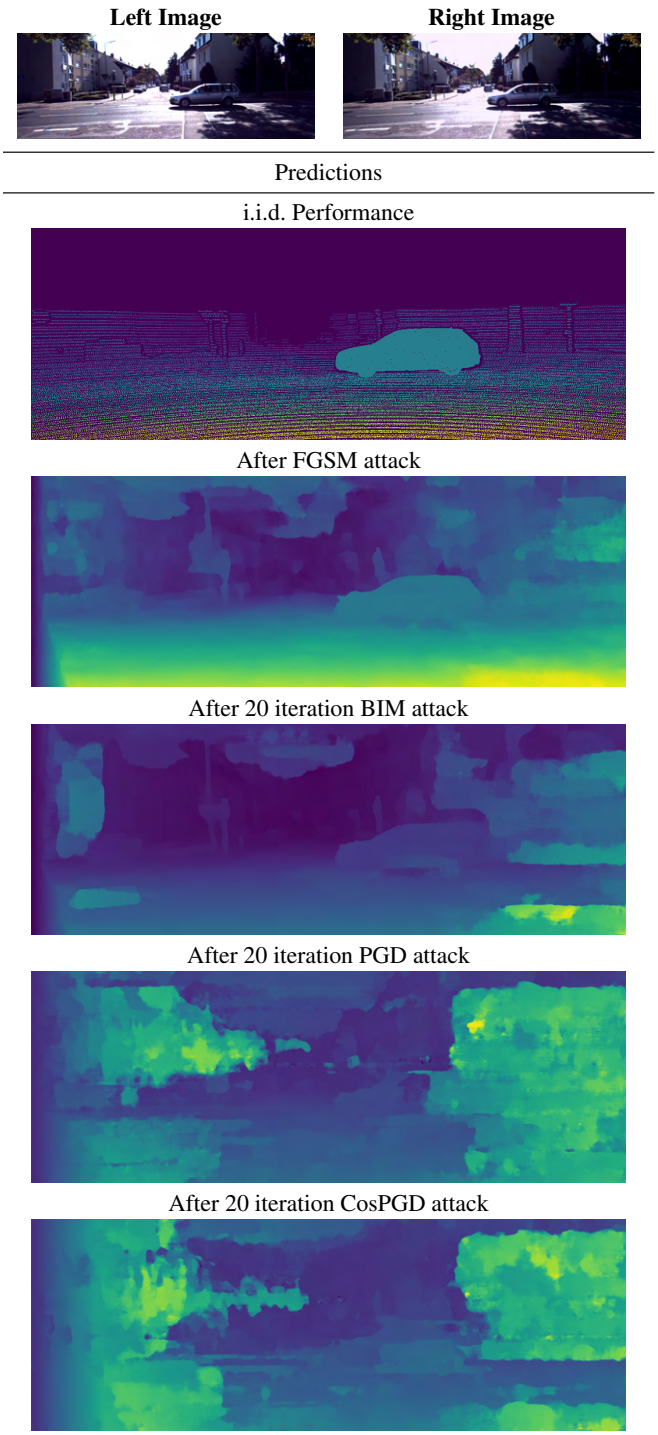

Figure 2. Example of performing adversarial attacks on STTR using KITTI2015 dataset under different attacks. We show the samples before and after the attacks and the predictions before and after the respective adversarial attacks.

In Fig. 2, we show example images perturbed using different adversarial attacks and the change in disparity esti-

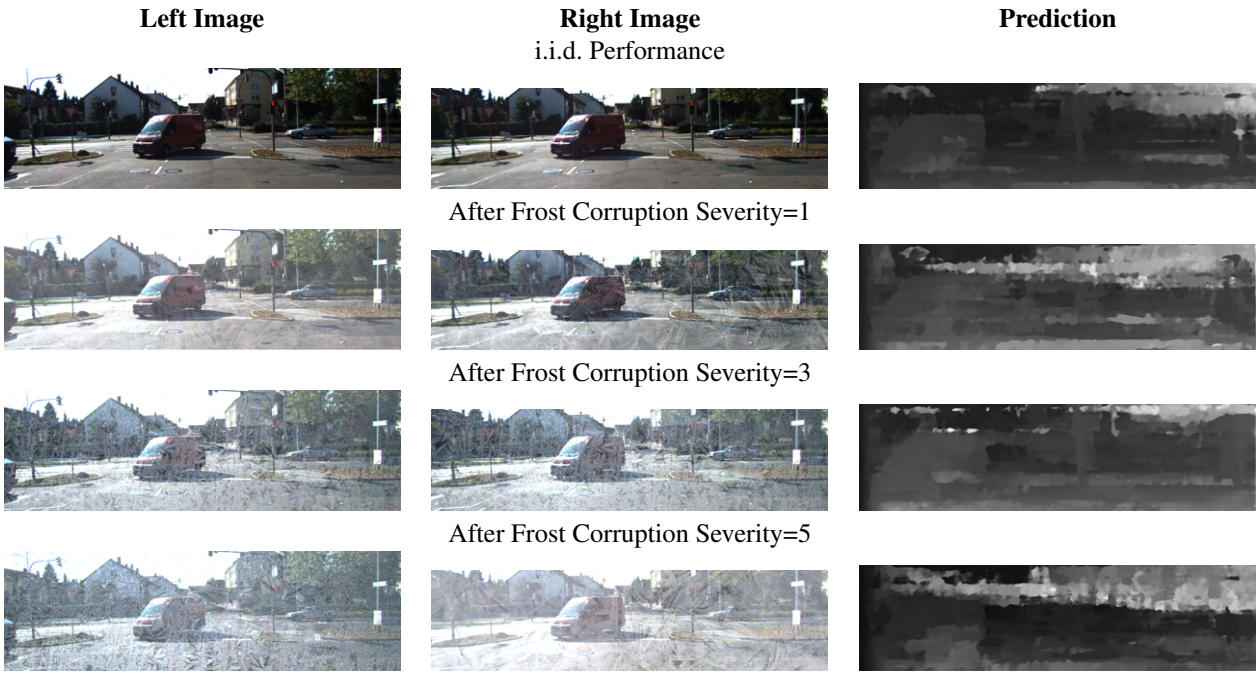

Figure 3. Example of predictions using STTR on KITTI2015 dataset under different severities of the 2D Common Corruption: Frost.

mation performed by STTR. Here, all attacks are optimized for 20 attack iterations, with α=0.01 and $\epsilon = \frac{8}{255}$ under the ℓ_∞-norm bound.

2.3. 2D Common Corruptions

To evaluate a model for a given dataset with 2D Common Corruptions, the following lines of code are required.

```
from dispbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015',
  ↪  retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model; for example, when the threat model is 2D Common Corruption, 'config.yml' could contain 'threat_model=*"2DCommonCorruption"*', and 'severity=*3*'. Please note, when the 'threat_model' is the common corruption, DISPBENCH performs evaluations on all corruptions under the respective 'threat_model' and returns the method's performance on each corruption at the requested severity. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False' then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the common corruption to be used, given as a string, i.e. '2DCommonCorruption'.
  - 'severity' is the severity of the corruption, given as an integer between 1 and 5 (both inclusive).

DISPBENCH supports the following 2D Common Corruption: 'gaussian_noise', shot_noise', 'impulse_noise', 'defocus_blur', 'frosted_glass_blur', 'motion_blur', 'zoom_blur', 'snow', 'frost', 'fog', 'brightness', 'contrast', 'elastic', 'pixelate', 'jpeg'. For the evaluation, DISPBENCH will evaluate the model on the validation images from the respective dataset corrupted using each of the aforementioned corruptions for the given severity and then report the mean performance over all of them.

In Fig. 3, we show example images perturbed using the 2D Common Corruption: Frost and the change in disparity estimation performed by STTR over different severity strengths.

3. Initial Evaluations using DISPBENCH

We use DISPBENCH to perform some initial benchmarking and make some interesting observations. Following, we discuss the details of the benchmarking process. Please note,

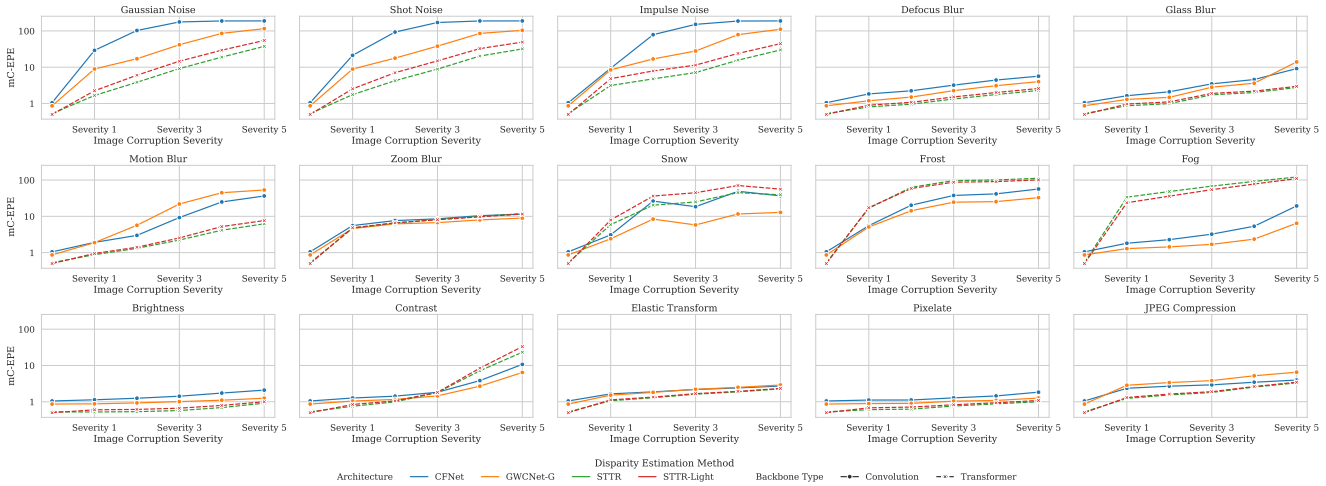

Figure 4. Using the FlyingThings3D dataset for disparity estimation, we perform an initial benchmarking of i.i.d. performance and generalization abilities of four popular disparity estimation methods. CFNet and GWCNet are traditional CNN-based stereo matching methods, whereas STTR and STTR-light are newly proposed transformer-based large models capable of zero-shot disparity estimation. Here, we use their fine-tuned versions for the FlyingThings3D dataset. The y-axis reports the mean EPE over the entire validation set for the respective corruption, and the x-axis denotes the severity of the 2D Common Corruption used to corrupt the input images. We report the i.i.d. performance at severity=0. Here we observe that while all four methods are highly vulnerable to Noise and Weather corruptions, newly proposed STTR and STTR-light are surprisingly less robust than the older CNN-based methods against weather corruptions. This finding is interesting and concerning as weather corruptions are the most likely real-world domain shift.

we use the FlyingThing3D and the KITTI2015 dataset for the benchmarking. However, very few pretrained architectures are available for KITTI2015, and thus our evaluations using KITTI2015 are limited to these. While DISPBENCH enables the training of architectures of multiple datasets, doing so is beyond our resource capabilities.

For additional details on the datasets, please refer to Appendix B.

Measuring Generalization Ability. Inspired by multiple works [8, 22, 23] that use OOD Robustness of methods for evaluating the generalization ability of the method, even evaluate over every common corruption, that is the 15 2D Common Corruptions: 'Gaussian Noise', Shot Noise', 'Impulse Noise', 'Defocus Blur', 'Frosted Glass Blur', 'Motion Blur', 'Zoom Blur', 'Snow', 'Frost', 'Fog', 'Brightness', 'Contrast', 'Elastic Transform', 'Pixelate', 'JPEG Compression'. Then, we find the mean EPE w.r.t. the ground truth for a given method, across all corruptions at a given severity and report use this to measure the Generalization Ability. We corrupt the pair of stereo images with the same corruption at the same severity when evaluating.

Ideally, one would like to evaluate the generalization ability and reliability of methods using real-world samples captured in the wild. However, annotation of these samples is a challenging and time-consuming task, and thus, no such dataset is available for disparity estimation. Sakaridis et al. [36] captured such data in the wild with domain shifts due to changes in time of day and changes in weather conditions like snowfall, rain, and fog. They also provide pixel-level annotations for their images, however, these annotations are only available for semantic segmentation, and these images are monocular and not stereo. They propose this as the Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding (ACDC) dataset. Interestingly, in their work, Anonymous [3] showed a very strong positive correlation between the performance of most methods on the ACDC dataset and their performance against in-domain images corrupted with the 2D Common Corruptions to cause a synthetic domain shift. This is an important finding as it proves that 2D Common Corruptions can be used as a proxy for real-world domain shifts. We discuss this in Appendix A.

For details on the dataset, please refer to the appendix.

Measuring Reliability Under Adversarial Attacks. Adversarial attacks, especially white-box attacks, serve as a proxy for the worst-case scenario and help understand the quality of the representations learned by a model [1, 37, 41]. DISPBENCH provides the ability to evaluate the models against some popular adversarial attacks, as discussed in Sec. 2.2. However, we focus this work towards realistic corruptions possible in the real world. For evaluations over adversarial attacks, please refer to Appendix G.

Architectures Used. Disparity estimation networks essentially estimate optimal correspondence matching between pixels on epipolar lines in the left and right images to infer depth. Most disparity estimation architectures used a cost volume with cross-correlation or contamination of feature representations for the left and right images. However, **GWCNet-G** [16] proposed using group-wise correlations to construct the cost volume. This leads to a significant boost in i.i.d. performance and inference speed. **CFNet** [39] proposed fusing on multiple low-resolution dense cost volumes to enlarge the receptive field, enabling extraction of robust structural representations, followed by cascading the cost volume representations to alleviate the unbalanced disparity estimation. It was proposed to be robust to large domain differences and was SotA when proposed. **Stereo-Transformers (STTR)**, Li et al. [27] proposes to replace the cost volume construction with dense pixel matching using position information and attention to enable sequence-to-sequence matching. This relaxes the limitation of a fixed disparity range and identifies occluded regions with confidence estimates. STTR generalizes across different domains, even without fine-tuning. However, in our evaluations, we use fine-tuned checkpoints for a fair comparison of reliability and generalization capabilities. **STTR-light** is the lightweight version of STTR proposed for faster inference with only a marginal drop in i.i.d. performance. We use the publicly available pre-trained checkpoints for our evaluations.

4. Key Findings

Following, we present the key findings made using the initial benchmarking using the DISPBENCH.

4.1. FlyingThings3D

Following, we discuss the observations made in the robustness benchmark created using DISPBENCH. We report the evaluations in Fig. 4. Here, we observe that indeed the i.i.d. performance of the new methods like STTR and STTR-light is better than the older CNN-based CFNet and GWCNet-G, however, the same is not always true for their generalization abilities. All four considered methods appear to be robust to digital corruptions such as changes in brightness, contrast, elastic transform, pixelated, and JPEG compression to a significant extent. While, all four methods appear to be extremely non-robust to additive noise, possible in the real world due to sensor error, causing the mean errors to go as high as 100. Please note, compared to the single-digit EPE values for i.i.d. performance, these errors are significantly high.

The most interesting behavior is seen under different weather corruptions: Snow, Frost, and Fog. Here, all four methods are non-robust, however, the newer transformer-based methods STTR and STTR-light are significantly more non-robust. This is quite alarming, as weather corruptions are the most natural domain shifts possible in the real world, and here, the large models fail significantly worse. Especially under Frost and Fog corruptions, the larger STTR performs worse than its lightweight counterpart, STTR-light. This raises some interesting concerns that warrant further study and deeper analysis.

4.2. KITTI2015

There are very limited pre-trained architectures available on KITTI2015 for the disparity estimation task, namely GWCNet-G and STTR. We perform our analysis using these and report the evaluations in Fig. 5. We observe that the newly proposed STTR is less robust than GWCNet-C across all corruptions and severities. This does not align with the observations made with synthetic corruptions on the synthetic dataset FlyingThings3D. This suggests that further analysis is required.

5. Synthetic Corruptions on Synthetic Dataset vs Synthetic Corruptions on Real World Dataset

Following the findings from Sec. 4.2, we investigate whether the performance of models on synthetic corruptions (2D Common Corruptions) on synthetic dataset (FlyingThings3D) can serve as a proxy for the performance of models on synthetic corruptions (2D Common Corruptions) on real-world data (KITTI2015). We report this analysis in Fig. 6 and observe that synthetic corruptions on synthetic datasets do not represent synthetic corruptions on real-world datasets. As known from [3], synthetic corruptions on real-world datasets represent real-world corruptions. By extension, synthetic corruptions on synthetic datasets do not represent real-world corruptions. This crucial finding eliminates the possibility of using synthetic simulators like CARLA [12], LGSVL (SVL Simulator) [35], AirSim [38], and others for possible applications in the real world.

6. Conclusion

Evaluating a model's robustness is vital for real-world applications. However, capturing corruptions in the real world is time and resource intensive. Here, synthetic corruptions appear to be an attractive alternative. Thus, we propose DISPBENCH, the first robustness benchmarking tool and a novel benchmark on synthetic corruptions for disparity estimation methods. We discuss the unique features of DISPBENCH in detail and demonstrate that the library is user-friendly, such that adding new methods or performing evaluation is very intuitive. We use DISPBENCH to evaluate the i.i.d. performance and OOD generalization of some popularly used disparity estimation methods. We ob-

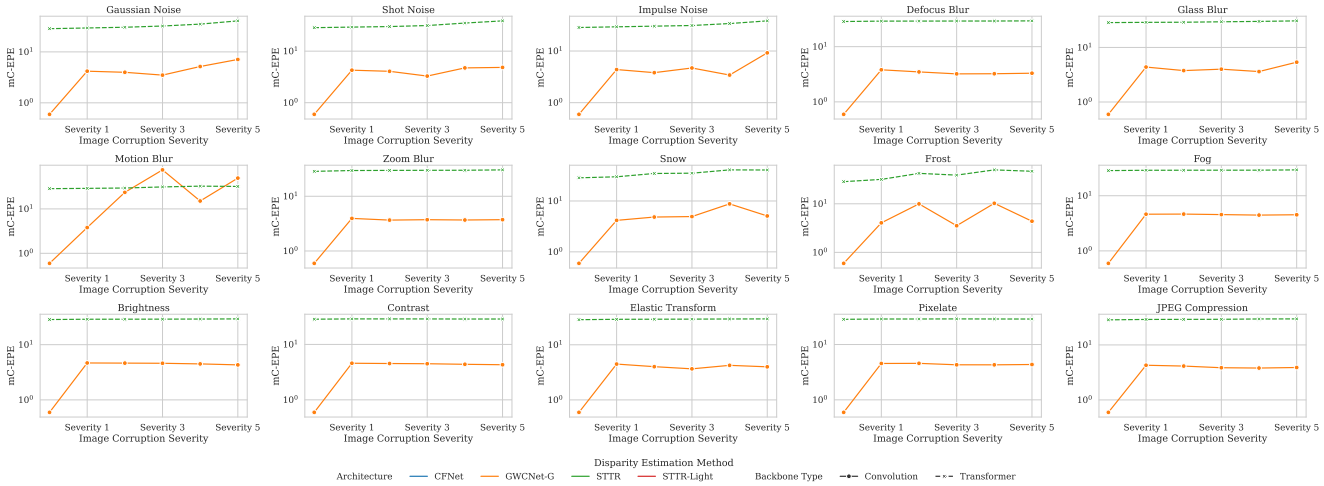

Figure 5. Using the KITTI2015 dataset for disparity estimation, we perform an initial benchmarking of i.i.d. performance and generalization abilities of the two popular and available disparity estimation methods. GWCNet is a traditional CNN-based stereo matching method, whereas STTR is a newly proposed transformer-based large model capable of zero-shot disparity estimation. Here, we use their fine-tuned versions for the KITTI2015 dataset. The y-axis reports the mean EPE over the entire validation set for the respective corruption, and the x-axis denotes the severity of the 2D Common Corruption used to corrupt the input images. We report the i.i.d. performance at severity=0. Here, we observe that while both the methods are highly vulnerable to Noise and Weather corruptions, the newly proposed STTR is surprisingly less robust than the older CNN-based method against all corruptions. This finding is interesting and concerning as it contradicts the findings on the Synthetic Dataset FlyingThings3D in Fig. 4.

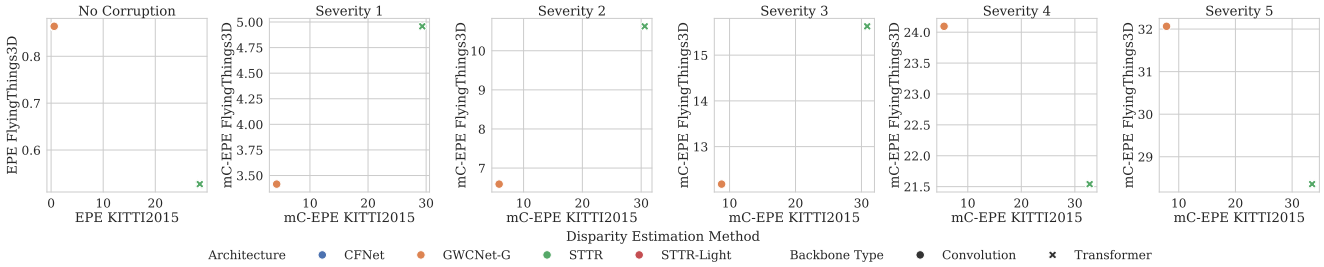

Figure 6. For the same architecture, we evaluate checkpoint pretrained on Flyingthings3D against synthetic 2D Common Corruption on Flyingthings3D and correlate its performance with the checkpoint trained on KITTI2015 against synthetic 2D Common Corruptions on KITTI2015. Here, we report the mean EPE across all corruptions for a given severity level. For individual corruptions, please refer to Fig. 9. We observe no correlation in performance, indicating that synthetic corruptions on synthetic datasets cannot be used as a proxy for real-world corruptions.

serve that under realistic scenarios, recently proposed large transformer-based methods known to be SotA on i.i.d. samples do not generalize well to image corruptions, demonstrating the gap in current research when considering real-world applications. Lastly, we show experimentally that synthetic corruptions on synthetic datasets do not represent real-world corruptions, thus, synthetic corruptions on real-world datasets present a more promising path. DISPBENCH enables a more in-depth understanding of reliability and generalization abilities to disparity estimation methods, and its consolidated nature would make future research more streamlined.

Future Work. Very recently, OpenStereo [17] has been made public that supports newly proposed stereo matching methods, which are foundational models for stereo matching like StereoAnything [19], and LightStereo [18]. We intend to adapt our evaluator into OpenStereo to enable safety studies of SotA disparity estimation methods.

Limitations. Benchmarking disparity estimation methods is a compute and labor-intensive endeavor. Thus, best utilizing available resources, we currently use DISPBENCH to benchmark a limited number of settings, using the most popular works for now. The benchmarking tool itself offers significantly more combinations that can be benchmarked.

Reproducibility Statement

Every experiment in this work is reproducible and is part of an effort toward open-source work. DISPBENCH will be open-source and publicly available, including all evaluation logs and model checkpoint weights. This work intends to help the research community use synthetic corruptions to build more reliable and generalizable disparity estimation methods such that they are ready for deployment in the real world even under safety-critical applications.

References

[1] Shashank Agnihotri, Steffen Jung, and Margret Keuper. CosPGD: an efficient white-box adversarial attack for pixel-wise prediction tasks. In *Proc. International Conference on Machine Learning (ICML)*, 2024. 1, 2, 5, 12, 13

[2] Shashank Agnihotri, Julian Yuya Caspary, Luca Schwarz, Xinyan Gao, Jenny Schmalfuss, Andres Bruhn, and Margret Keuper. Flowbench: A robustness benchmark for optical flow estimation, 2025. 12

[3] Anonymous. Are Synthetic Corruptions A Reliable Proxy For Real-World Corruptions?, 2025. 5, 6, 10, 11

[4] Muhammad Awais, Weiming Zhuang, Lingjuan Lyu, and Sung-Ho Bae. Frod: Robust object detection for free. *CoRR*, 2023. 2

[5] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. European Conference on Computer Vision (ECCV)*, pages 611–625, 2012. 12

[6] Weiqin Chuah, Ruwan Tennakoon, Reza Hoseinnezhad, and Alireza Bab-Hadiashar. Deep learning-based incorporation of planar constraints for robust stereo depth estimation in autonomous vehicle applications. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6654–6665, 2021. 2

[7] 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, 2016. 10, 11

[8] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 5, 12

[9] Yuexiong Ding, Ming Zhang, Jia Pan, Jinxing Hu, and Xiaowei Luo. Robust object detection in extreme construction conditions. *Automation in Construction*, 165:105487, 2024. 2

[10] Ziyi Dong, Pengxu Wei, and Liang Lin. Adversarially-aware robust object detector. In *European Conference on Computer Vision*, pages 297–313. Springer, 2022. 2

[11] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 11

[12] 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. 6

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018. 1

[14] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. 1

[15] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. International Conference on Learning Representations (ICLR)*, 2015. 1, 2, 12

[16] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019. 1, 6

[17] Xianda Guo, Chenming Zhang, Juntao Lu, Yiqi Wang, Yiqun Duan, Tian Yang, Zheng Zhu, and Long Chen. Openstereo: A comprehensive benchmark for stereo matching and strong baseline. *arXiv preprint arXiv:2312.00343*, 2023. 2, 7, 10

[18] Xianda Guo, Chenming Zhang, Dujun Nie, Wenzhao Zheng, Youmin Zhang, and Long Chen. Lightstereo: Channel boost is all your need for efficient 2d cost aggregation. *arXiv preprint arXiv:2406.19833*, 2024. 7

[19] Xianda Guo, Chenming Zhang, Youmin Zhang, Dujun Nie, Ruilin Wang, Wenzhao Zheng, Matteo Poggi, and Long Chen. Stereo anything: Unifying stereo matching with large-scale mixed data. *arXiv preprint arXiv:2411.14053*, 2024. 7, 11

[20] Himanshu Gupta, Oleksandr Kotlyar, Henrik Andreasson, and Achim J Lilienthal. Robust object detection in challenging weather conditions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7523–7532, 2024. 2

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. International Conference on Learning Representations (ICLR)*, 2019. 1, 2, 10, 11, 12

[22] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. 1, 5

[23] J Hoffmann, S Agnihotri, Tonmoy Saikia, and Thomas Brox. Towards improving robustness of compressed cnns. In *ICML Workshop on Uncertainty and Robustness in Deep Learning (UDL)*, 2021. 1, 5

[24] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18963–18974, 2022. 10, 12

[25] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *Proc. International Conference on Learning Representations (ICLR)*, 2017. 1, 2, 12

[26] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, pages 99–112. Chapman and Hall/CRC, 2018. 2, 12

[27] Zhaoshuo Li, Xingtong Liu, Nathan Drenkow, Andy Ding, Francis X. Creighton, Russell H. Taylor, and Mathias Unberath. Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6197–6206, 2021. 1, 6, 11

[28] Zihua Liu, Yizhou Li, and Masatoshi Okutomi. Global occlusion-aware transformer for robust stereo matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3535–3544, 2024. 2

[29] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. 1, 2, 11, 12

[30] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015. 2, 11, 12

[31] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019. 2

[32] Jan Müller, Reuben Docea, Matthias Hardner, Katja Krug, Paul Riedel, and Ronald Tetzlaff. Fast high-resolution disparity estimation for laparoscopic surgery. In *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 573–577. IEEE, 2022. 2

[33] Adarsh Prasad. *Towards Robust and Resilient Machine Learning*. PhD thesis, Carnegie Mellon University, 2022. 1

[34] Dimitrios Psychogyios, Evangelos Mazomenos, Francisco Vasconcelos, and Danail Stoyanov. Msdesis: Multitask stereo disparity estimation and surgical instrument segmentation. *IEEE transactions on medical imaging*, 41(11):3218–3230, 2022. 2

[35] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. *arXiv preprint arXiv:2005.03778*, 2020. 6

[36] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 5, 10, 11

[37] Jenny Schmalfuss, Philipp Scholze, and Andrés Bruhn. A perturbation-constrained adversarial attack for evaluating the robustness of optical flow. In *Proc. European Conference on Computer Vision (ECCV)*, pages 183–200, 2022. 5

[38] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. 6

[39] Zhelun Shen, Yuchao Dai, and Zhibo Rao. Cfnet: Cascade and fused cost volume for robust stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13906–13915, 2021. 1, 6, 11

[40] Shiyu Tang, Ruihao Gong, Yan Wang, Aishan Liu, Jiakai Wang, Xinyun Chen, Fengwei Yu, Xianglong Liu, Dawn Song, Alan Yuille, Philip H.S. Torr, and Dacheng Tao. Robustart: Benchmarking robustness on architecture design and training techniques. *https://arxiv.org/pdf/2109.05211.pdf*, 2021. 2

[41] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. *ArXiv*, abs/2001.03994, 2020. 2, 5

[42] Bo Yang, Siyuan Xu, Lirong Yin, Chao Liu, and Wenfeng Zheng. Disparity estimation of stereo-endoscopic images using deep generative network. *ICT Express*, 2024. 2

[43] Jiawei Zhang, Jiahe Li, Lei Huang, Xiaohan Yu, Lin Gu, Jin Zheng, and Xiao Bai. Robust synthetic-to-real transfer for stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20247–20257, 2024. 2

[44] Shuangli Zhang, Weijian Xie, Guofeng Zhang, Hujun Bao, and Michael Kaess. Robust stereo matching with surface normal prediction. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2540–2547, 2017. 2

DISPBENCH: A Robustness Evaluator For Disparity Estimation

Paper #4 Supplementary Material

Table Of Content

The supplementary material covers the following information:

- Appendix A: We show that synthetic 2D Common Corruptions are indeed serve as a proxy to domain shifts in the real world.
- Appendix B: Details for the datasets used.
  - Appendix B.1: FlyingThings3D
  - Appendix B.2: KITTI2015
- Appendix C: Additional implementation details for the evaluated benchmark.
- Appendix D: In detail description of the attacks.
- Appendix E: DISPBENCH function call to get model weights.
- Appendix F: In detail explanation of the available functionalities of the DISPBENCH benchmarking tool and description of the arguments for each function.
- Appendix G: Here we provide additional results from the benchmark evaluated using DISPBENCH.
- At the end, we attach the anonymous paper of [3] for the ease of the reviewer.

Please note, due to the similarity of the objective, many aspects of this appendix are very similar to Anonymous [3].

A. Do Synthetic Corruptions Represent The Real World?

In their work Anonymous [3], they find the correlation between mean mIoU over the ACDC evaluation dataset [36] and mean mIoU over each 2D Common Corruption [21] over the Cityscapes dataset [7]. We include Figure 7 from their work here for ease of understanding. All models were trained using the training subset of the Cityscapes dataset. ACDC is the Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding captured in similar scenes are cityscapes but under four different domains: Day/Night, Rain, Snow, and Fog in the wild. ACDC is a community-used baseline for evaluating the performance of semantic segmentation methods on domain shifts observed in the wild. They find that there exists a very strong positive correlation between the two. This shows, that **yes, synthetic corruptions can serve as a proxy for the real world**. Unfortunately, a similar "in the wild" captured dataset does not exist for optical flow estimation to evaluate the effect of domain shifts on the performance of optical flow methods. However, given that for the task of semantic segmentation, we find a very high positive correlation between the performance on real-world corruptions and synthetic corruptions, it is a safe assumption that the same would hold true for optical flow estimation as well. Thus, in this work, we evaluate against synthetic 2D Common Corruptions [21] and synthetic 3D Common Corruptions [24].

B. Dataset Details

DISPBENCH currently supports two distinct disparity datasets. We intent to extend DISPBENCH to 4 other commonly used datasets by integrating DISPBENCH with OpenStereo [17]. Following, we describe these datasets in detail.

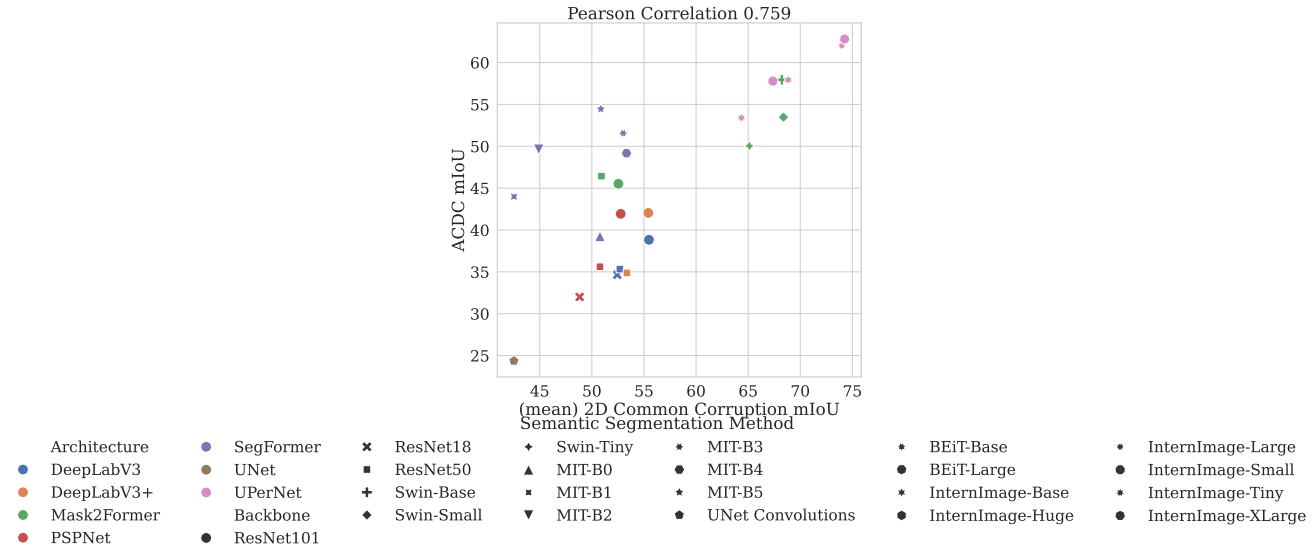

Figure 7. Results from work by Anonymous [3]. Here they find a **very strong positive correlation between mean mIoU over the ACDC evaluation dataset [36] and mean mIoU over each 2D Common Corruption [21]** over the Cityscapes dataset [7]. All models were trained using the training subset of the Cityscapes dataset. ACDC is the Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding captured in similar scenes are cityscapes but under four different domains: Day/Night, Rain, Snow, and Fog in the wild. ACDC is a community-used baseline for evaluating the performance of semantic segmentation methods on domain shifts observed in the wild.

B.1. FlyingThings3D

This is a synthetic dataset proposed by [29] largely used for training and evaluation of disparity estimation methods. This dataset consists of 25000 stereo frames, of everyday objects such as chairs, tables, cars, etc. flying around in 3D trajectories. The idea behind this dataset is to have a large volume of trajectories and random movements rather than focus on a real-world application. In their work, [11] showed models trained on FlyingThings3D can generalize to a certain extent to other datasets.

B.2. KITTI2015

Proposed by [30], this dataset is focused on the real-world driving scenario. It contains a total of 400 pairs of image frames, split equally for training and testing. The image frames were captured in the wild while driving around on the streets of various cities. The ground-truth labels were obtained by an automated process.

C. Implementation Details Of The Benchmark

Following we provide details regarding the experiments done for creating the benchmark used in the analysis.

Compute Resources. Most experiments were done on a single 40 GB NVIDIA Tesla V100 GPU each, however, MS-RAFT+, FlowFormer, and FlowFormer++ are more compute-intensive, and thus 80GB NVIDIA A100 GPUs or NVIDIA H100 were used for these models, a single GPU for each experiment.

Datasets Used. Performing adversarial attacks and OOD robustness evaluations are very expensive and compute-intensive. Thus, performing evaluation using all model-dataset pairs is not possible given the limited computing resources at our disposal. Thus, for the benchmark, we only use FlyingThings3D and KITTI2015, as these are the most commonly used datasets for evaluation [19, 27, 29, 39].

Metrics Calculation. In this work, for robustness evaluations we consider the mC-EPE, which is the mean End-Point-Error of a method, against common corruptions at a given severity, over every input image from the validation dataset. We use all 15 2D Common Corruptions: 'Gaussian Noise', Shot Noise', 'Impulse Noise', 'Defocus Blur', 'Frosted Glass

Blur', 'Motion Blur', 'Zoom Blur', 'Snow', 'Frost', 'Fog', 'Brightness', 'Contrast', 'Elastic Transform', 'Pixelate', 'JPEG Compression'. All the common corruptions are at severity=$\{1, 2, 3, 4, 5\}$. [24] offers more 3D Common Corruptions, however computing them is resource intensive. Thus, given our limited resources and an overlap in the corruptions between 2D Common Corruptions and 3D Common Corruptions, we focus on generating 3D Common Corruptions for now, however, we intend to extend DISPBENCH to also evaluate on the 3D Common Corruptions.

Calculating the EPE. EPE is the Euclidean distance between the two vectors, where one vector is the predicted flow by the disparity estimation method and the other vector is the ground truth in case of i.i.d. performance evaluations, non-targeted attacks evaluations, and OOD robustness evaluations, while it is the target flow vector, in case of targeted attacks. For each dataset, the EPE value is calculated over all the samples of the evaluation set of the respective dataset and then the mean EPE value is used as the mean-EPE of the respective method over the respective dataset.

D. Description of DISPBENCH

Following, we describe the benchmarking tool, DISPBENCH. There exists no standardized tool for evaluating the performance of disparity estimation methods. Thus, the codebase for such a tool had to be written from scratch. In the following, we describe the benchmarking tool, DISPBENCH. Currently it supports 4 unique architectures (new architectures to be added to DISPBENCH with time) and 3 distinct datasets, namely FlyingThings3D [29], KITTI2015 [30], MPI Sintel [5] (clean and final) (please refer Appendix B for additional details on the datasets). It enables training and evaluations on all aforementioned datasets including evaluations using SotA adversarial attacks such as CosPGD [1], and other commonly used adversarial attacks like BIM [26], PGD [25], FGSM [15], under various Lipshitz (l_p) norm bounds. Additionally, it enables evaluations for Out-of-Distribution (OOD) robustness by corrupting the inference samples using 2D Common Corruptions [21].

Following we show the basic commands to use DISPBENCH. We describe each attack and common corruption supported by DISPBENCH in detail in Appendix D. It enables training and evaluations on all aforementioned datasets including evaluations using SotA adversarial attacks such as CosPGD [1], and other commonly used adversarial attacks like BIM [26], PGD [25], FGSM [15], under various lipshitz (l_p) norm bounds. Additionally, it enables evaluations for Out-of-Distribution (OOD) robustness by corrupting the inference samples using 2D Common Corruptions [21].

We follow the nomenclature set by RobustBench [8] and use "threat_model" to define the kind of evaluation to be performed. When "threat_model" is defined to be "None", the evaluation is performed on unperturbed and unaltered images, if the "threat_model" is defined to be an adversarial attack, for example "PGD", "CosPGD" or "PCFA", then DISPBENCH performs an adversarial attack using the user-defined parameters. Whereas, if "threat_model" is defined to be "2DCommonCorruptions" or "3DCommonCorruptions", the DISPBENCH performs evaluations after perturbing the images with 2D Common Corruptions and 3D Common Corruptions respectively.

If the queried evaluation already exists in the benchmark provided by this work, then DISPBENCH simply retrieves the evaluations, thus saving computation.

D.1. Adversarial Attacks

Please note that due to the similarity of the objective, many aspects of this appendix are very similar to Agnihotri et al. [2]. DISPBENCH enables the use of many white-box adversarial attacks to help users better study the reliability of their disparity methods. We choose to specifically include these white-box adversarial attacks as they either serve as the common benchmark for adversarial attacks in classification literature (FGSM, BIM, PGD, APGD) or they are unique attacks proposed specifically for pixel-wise prediction tasks (CosPGD). These attacks can either be *Non-targeted* which are designed to simply fool the model into making incorrect predictions, irrespective of what the model eventually predicts, or can be *Targeted*, where the model is fooled to make a certain prediction. Most attacks can be, designed to be either Targeted or Non-targeted, these include, FGSM, BIM, PGD, APGD, CosPGD, and Adversarial Weather. In our current implementation, we are limited to Non-targeted attacks. Following, we discuss these attacks in detail and highlight their key differences.

FGSM. Assuming a non-targeted attack, given a model f_θ and an unperturbed input sample $\boldsymbol{X}^{\text{clean}}$ and ground truth label $\boldsymbol{Y}$, FGSM attack adds noise δ to $\boldsymbol{X}^{\text{clean}}$ as follows,

$$\boldsymbol{X}^{\text{adv}} = \boldsymbol{X}^{\text{clean}} + \alpha \cdot \text{sign} \nabla_{\boldsymbol{X}^{\text{clean}}} L(f_\theta(\boldsymbol{X}^{\text{clean}}), \boldsymbol{Y}), \tag{1}$$

$$\delta = \phi^{\epsilon}(\boldsymbol{X}^{\text{adv}} - \boldsymbol{X}^{\text{clean}}), \tag{2}$$

$$\boldsymbol{X}^{\text{adv}} = \phi^r(\boldsymbol{X}^{\text{clean}} + \delta). \tag{3}$$

Here, $L(\cdot)$ is the loss function (differentiable at least once) which calculates the loss between the model prediction and ground truth, $\boldsymbol{Y}$. α is a small value of ϵ that decides the size of the step to be taken in the direction of the gradient of the loss w.r.t. the input image, which leads to the input sample being perturbed such that the loss increases. $\boldsymbol{X}^{\text{adv}}$ is the adversarial sample obtained after perturbing $\boldsymbol{X}^{\text{clean}}$. To make sure that the perturbed sample is semantically indistinguishable from the unperturbed clean sample to the human eye, steps from Eq. (2) and Eq. (3) are performed. Here, function ϕ^ϵ is clipping the δ in ϵ-ball for ℓ_∞-norm bounded attacks or the ϵ-projection in other l_p-norm bounded attacks, complying with the ℓ_∞-norm or other l_p-norm constraints, respectively. While function ϕ^r clips the perturbed sample ensuring that it is still within the valid input space. FGSM, as proposed, is a single step attack. For targeted attacks, $\boldsymbol{Y}$ is the target and α is multiplied by -1 so that a step is taken to minimize the loss between the model's prediction and the target prediction.

BIM. This is the direct extension of FGSM into an iterative attack method. In FGSM, $\boldsymbol{X}^{\text{clean}}$ was perturbed just once. While in BIM, $\boldsymbol{X}^{\text{clean}}$ is perturbed iteratively for time steps $t \in [0, \boldsymbol{T}]$, such that $t \in \mathbb{Z}^+$, where $\boldsymbol{T}$ are the total number of permissible attack iterations. This changes the steps of the attack from FGSM to the following,

$$\boldsymbol{X}^{\text{adv}_{t+1}} = \boldsymbol{X}^{\text{adv}_t} + \alpha \cdot \text{sign} \nabla_{\boldsymbol{X}^{\text{adv}_t}} L(f_\theta(\boldsymbol{X}^{\text{adv}_t}), \boldsymbol{Y}), \tag{4}$$

$$\delta = \phi^\epsilon(\boldsymbol{X}^{\text{adv}_{t+1}} - \boldsymbol{X}^{\text{clean}}), \tag{5}$$

$$\boldsymbol{X}^{\text{adv}_{t+1}} = \phi^r(\boldsymbol{X}^{\text{clean}} + \delta). \tag{6}$$

Here, at t=0, $\boldsymbol{X}^{\text{adv}_t}$=$\boldsymbol{X}^{\text{clean}}$.

PGD. Since in BIM, the initial prediction always started from $\boldsymbol{X}^{\text{clean}}$, the attack required a significant amount of steps to optimize the adversarial noise and yet it was not guaranteed that in the permissible ϵ-bound, $\boldsymbol{X}^{\text{adv}_{t+1}}$ was far from $\boldsymbol{X}^{\text{clean}}$. Thus, PGD proposed introducing stochasticity to ensure random starting points for attack optimization. They achieved this by perturbing $\boldsymbol{X}^{\text{clean}}$ with $\mathcal{U}(-\epsilon, \epsilon)$, a uniform distribution in $[-\epsilon, \epsilon]$, before making the first prediction, such that, at t=0

$$\boldsymbol{X}^{adv_t} = \phi^r(\boldsymbol{X}^{clean} + \mathcal{U}(-\epsilon, \epsilon)). \tag{7}$$

APGD. Auto-PGD is an effective extension to the PGD attack that effectively scales the step size α over attack iterations considering the compute budget and the success rate of the attack.

CosPGD. All previously discussed attacks were proposed for the image classification task. Here, the input sample is a 2D image of resolution H $\times$ W, where H and W are the height and width of the spatial resolution of the sample, respectively. Pixel-wise information is inconsequential for image classification. This led to the pixel-wise loss $\mathcal{L}(\cdot)$ being aggregated to L$(\cdot)$, as follows,

$$L(f_\theta(\boldsymbol{X}^{\text{adv}_t}), \boldsymbol{Y}) = \frac{1}{\text{H} \times \text{W}} \sum_{i \in \text{H} \times \text{W}} \mathcal{L}(f_\theta(\boldsymbol{X}^{\text{adv}_t})_i, \boldsymbol{Y}_i). \tag{8}$$

This aggregation of $\mathcal{L}(\cdot)$ fails to account for pixel-wise information available in tasks other than image classification, such as pixel-wise prediction tasks like Optical Flow estimation, and disparity estimation. Thus, in their work [1] propose an effective extension of the PGD attack that takes pixel-wise information into account by scaling $\mathcal{L}(\cdot)$ by the alignment between the distribution of the predictions and the distributions of $\boldsymbol{Y}$ before aggregating leading to a better-optimized attack, modifying Eq. (4) as follows,

$$\boldsymbol{X}^{\text{adv}_{t+1}} = \boldsymbol{X}^{\text{adv}_t} + \alpha \cdot \text{sign} \nabla_{\boldsymbol{X}^{\text{adv}_t}} \sum_{i \in H \times W} \cos\left(\psi(f_\theta(\boldsymbol{X}^{\text{adv}_t})_i), \Psi(\boldsymbol{Y}_i)\right) \cdot \mathcal{L}\left(f_\theta(\boldsymbol{X}^{\text{adv}_t})_i, \boldsymbol{Y}_i\right). \tag{9}$$

Where, functions ψ and Ψ are used to obtain the distribution over the predictions and $\boldsymbol{Y}_i$, respectively, and the function cos calculates the cosine similarity between the two distributions. CosPGD is the unified SotA adversarial attack for pixel-wise prediction tasks.

In Figure 2, we show examples of adversarial attacks, on STTR using the KITTI2015 dataset. We show the samples before and after the attacks and the predictions before and after the respective adversarial attacks.

E. Model Zoo

It is challenging to find all checkpoints whereas training them is time and compute-exhaustive. Thus we gather available model checkpoints made available online by the respective authors. The trained checkpoints for all models available in DISPBENCH can be obtained using the following lines of code:

```
from dispbench.evals import load_model
model = load_model(model_name='STTR',
    dataset='KITTI2015')
```

Each model checkpoint can be retrieved with the pair of 'model_name', the name of the model, and 'dataset', the dataset for which the checkpoint was last finetuned.

F. DISPBENCH Usage Details

Following we provide a detailed description of the evaluation functions and their arguments provided in DISPBENCH.

F.1. Adversarial Attacks

To evaluate a model for a given dataset, on an attack, the following lines of code are required.

```
from dispcbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015' retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model, for example, when the threat model is a PGD attack, 'config.yml' could contain 'threat_model=*"PGD"*', 'iterations=*20*', 'alpha=*0.01*', 'epsilon=*8*', and 'lp_norm=*"Linf"*'. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False' then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the adversarial attack to be used, given as a string.
  - 'iterations' are the number of attack iterations, given as an integer.
  - 'epsilon' is the permissible perturbation budget ϵ given a floating point (float).
  - 'alpha' is the step size of the attack, α, given as a floating point (float).
  - 'lp_norm' is the Lipschitz continuity norm (l_p-norm) to be used for bounding the perturbation, possible options are 'Linf' and 'L2' given as a string.
  - 'target' is false by default, but to do targeted attacks, either the user can set 'target'=True, to use the default target of $\overrightarrow{0}$, or can pass a specific tensor to be used as the target.

F.2. 2D Common Corruptions

To evaluate a model for a given dataset, with 2D Common Corruptions, the following lines of code are required.

```
from dispbench.evals import evaluate
model, results = evaluate(
 model_name='STTR', dataset='KITTI2015', retrieve_existing=True,
 threat_config='config.yml')
```

Here, the 'config.yml' contains the configuration for the threat model; for example, when the threat model is 2D Common Corruption, 'config.yml' could contain 'threat_model=*"2DCommonCorruption"*', and 'severity=*3*'. Please note, when the 'threat_model' is a common corruption type, DISPBENCH performs evaluations on all corruptions under the respective 'threat_model' and returns the method's performance on each corruption at the requested severity. The argument description is as follows:

- 'model_name' is the name of the disparity estimation method to be used, given as a string.
- 'dataset' is the name of the dataset to be used also given as a string.
- 'retrieve_existing' is a boolean flag, which when set to 'True' will retrieve the evaluation from the benchmark if the queried evaluation exists in the benchmark provided by this work, else DISPBENCH will perform the evaluation. If the 'retrieve_existing' boolean flag is set to 'False', then DISPBENCH will perform the evaluation even if the queried evaluation exists in the provided benchmark.
- The 'config.yml' contains the following:
  - 'threat_model' is the name of the common corruption to be used, given as a string, i.e. '2DCommonCorruption'.
  - 'severity' is the severity of the corruption, given as an integer between 1 and 5 (both inclusive).

DISPBENCH supports the following 2D Common Corruption: 'gaussian_noise', shot_noise', 'impulse_noise', 'defocus_blur', 'frosted_glass_blur', 'motion_blur', 'zoom_blur', 'snow', 'frost', 'fog', 'brightness', 'contrast', 'elastic', 'pixelate', 'jpeg'. For the evaluation, DISPBENCH will evaluate the model on the validation images from the respective dataset corrupted using each of the aforementioned corruptions for the given severity, and then report the mean performance over all of them.

G. Extension To Analysis

Following, we extend the analysis from Section 4 and report additional evaluations from DISPBENCH.

G.1. KITTI2015

Following, we provide evaluations of on the KITTI2015 dataset. In Figure 8 we report the evaluations of all considered

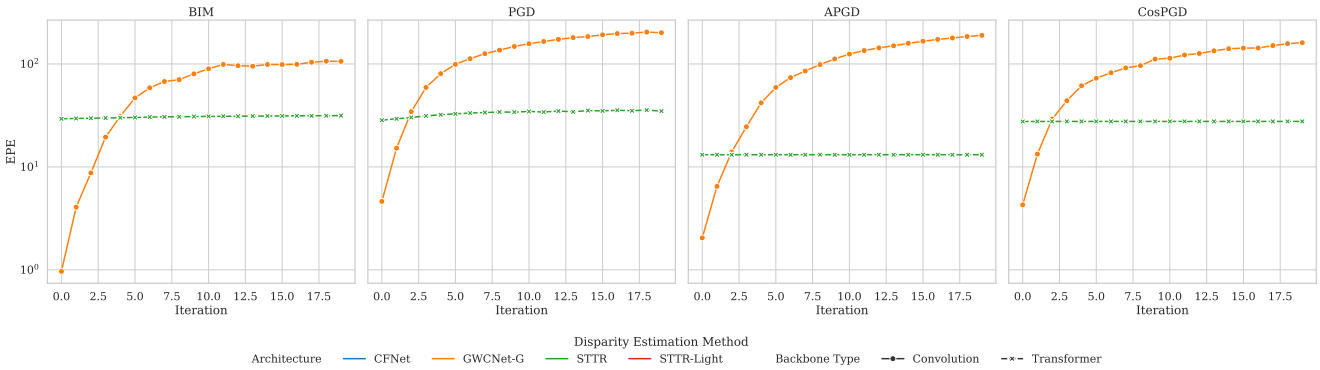

Figure 8. Evaluations of all considered adversarial attacks with $\epsilon = \frac{8}{255}$ and α=0.01 under the ℓ_∞-norm bound using the KITTI2015 validation dataset.

adversarial attacks with $\epsilon = \frac{8}{255}$ and α=0.01 under the ℓ_∞-norm bound using the KITTI2015 validation dataset.

In Figure 9, we extend the evaluations from Figure 6, showing that the observations made over the mean performance over all corruptions also hold for every individual corruption.

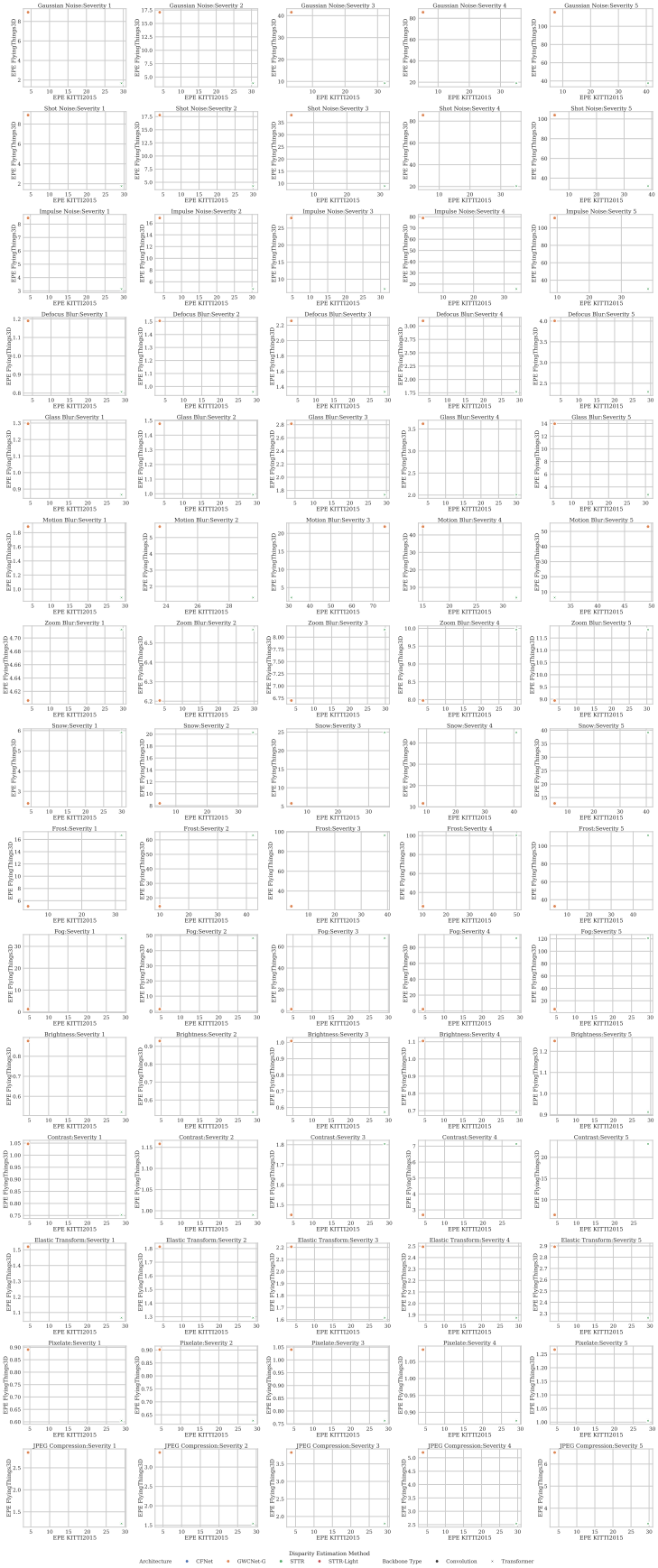

Figure 9. For the same architecture, we evaluate checkpoints pretrained on Flyingthings3D against synthetic 2D Common Corruption on Flyingthings3D and correlate its performance with the checkpoint trained on KITTI2015 against synthetic 2D Common Corruptions on KITTI2015. Here, we report the EPE across every individual corruption for a given severity level. We observe no correlation in performance, indicating that synthetic corruptions on synthetic datasets cannot be used as a proxy for real-world corruptions.

Next, we attach the Anonymous paper [3]: “Are Synthetic Corruptions A Reliable Proxy For Real-World Corruptions?”, Anonymous et. al., 2025 for the ease of the reviewer.

Are Synthetic Corruptions A Reliable Proxy For Real-World Corruptions?

Anonymous CVPR submission

Paper ID 0005

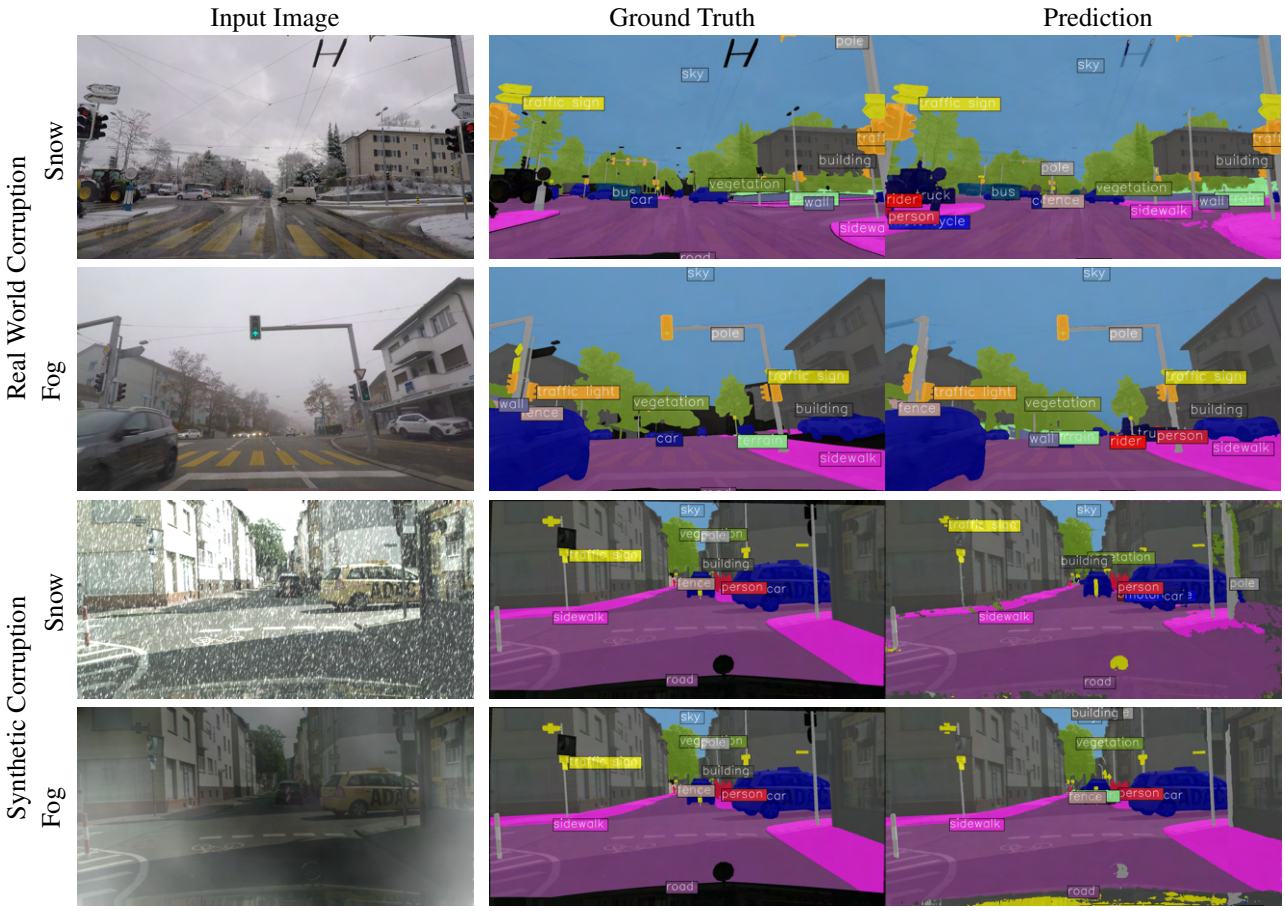

Figure 1. Comparing images with weather corruptions captured in the wild (ACDC [31]) and images corrupted using synthetic corruptions [19] and the predictions using a Mask2Former [7] with a Swin-Base [26] backbone trained on the Cityscapes [9] dataset.

Abstract

Deep learning (DL) models are widely used in real-world applications but remain vulnerable to distribution shifts, especially due to weather and lighting changes. Collecting diverse real-world data for testing the robustness of DL models is resource-intensive, making synthetic corruptions an attractive alternative for robustness testing. However, are synthetic corruptions a reliable proxy for real-world corruptions? To answer this, we conduct the largest benchmarking study on semantic segmentation models, comparing performance on real-world corruptions and synthetic corruptions datasets. Our results reveal a strong correlation in mean performance, supporting the use of synthetic corruptions for robustness evaluation. We further analyze corruption-specific correlations, providing key insights to understand when synthetic corruptions succeed in representing real-world corruptions. The code and datasets will be released upon acceptance.

1. Introduction

Although very successful in benchmark scenarios, the reliability of deep-learning (DL)-based models for semantic segmentation in real-world scenarios remains a major concern. Potentially unseen variations in the data (a.k.a. distribution shifts), for example, due to changes in weather conditions (e.g., fog, rain, snow) and lighting (e.g., nighttime, glare), can heavily degrade model performance. Ensuring robustness to such shifts is critical for safe and reliable deployment, particularly in applications like autonomous driving [9, 27] or medical imaging [11, 30]. To evaluate model robustness, researchers often rely on synthetic corruptions, such as [19]. These perturbations — designed to mimic real-world conditions — offer a scalable and controlled way to assess model performance without the cost of real-world data collection.

Several previous works [4, 23, 31] have also attempted to draw focus towards threats posed in real-world applications when facing slight domain shifts, for example, through noise or simply through changing weather. Specific evaluations involve the study of Out-Of-Distribution (OOD) samples to mimic realistic domain shifts.

Despite their widespread use, the correlation between model performance on synthetic and real-world corruptions is not well understood. Figure 1 shows one such scenario with real-world corruptions (Snow and Fog) captured in the ACDC dataset [31] and similar synthetic corruptions added on in-domain images from the cityscapes validation dataset. We observe very similar trends in the lack of robustness of the model towards both real-world and synthetic corruptions. However, a fundamental question remains:

> "Are synthetic corruptions a reliable proxy for real-world corruptions?"

If a strong correlation exists, synthetic corruptions could serve as a cost-effective alternative for robustness evaluation. Conversely, if the correlation is weak, extensive tests on real-world settings remain necessary at all stages.

Here, we conduct a large benchmarking study, analyzing the correlation between model performance on real-world and synthetic corruptions for semantic segmentation. The main contributions of this work are as follows:

- We benchmark multiple DL-based semantic segmentation models on real-world corruptions from the ACDC dataset and synthetic corruptions from Cityscapes + 2D Common Corruptions.
- We provide an in-depth analysis of corruption-specific trends, identifying cases where synthetic corruptions succeed or fail as proxies.
- We provide benchmarking of semantic segmentation methods against synthetic corruptions on ADE20k [37] and PASCAL VOC 2012 [13] datasets.

Our findings reveal a high correlation in mean performance, suggesting that synthetic corruptions can indeed serve as a reliable proxy for real-world robustness evaluation. However, we also highlight key cases where synthetic corruptions fail to fully capture real-world effects, underscoring the need for more nuanced evaluation methods.

2. Related Work

The robustness of DL-based methods to distribution shifts is often used as a measure of their generalization ability [20, 21]. Common Corruptions [19] and 3D Common Corruptions [24] are tools proposed for benchmarking the robustness of image classification models, but they can be extended to other vision tasks as for example done in [23]. However, both are synthetic corruptions, and distribution shifts occurring in the real world might be slightly different. Conversely, Sakaridis et al. [31] proposed "ACDC: The Adverse Conditions Dataset with Correspondences for Robust Semantic Driving Scene Perception". This dataset contains images captured in the wild in different conditions, such as during Night, Rain, Snow, and Fog. While ACDC does not cover many other possible conditions that can cause distribution shifts, it serves as a community-accepted tool for benchmarking real-world OOD robustness to a certain extent.

In this work, we use both Common Corruptions and ACDC to benchmark OOD robustness and thus measure the generalization ability of various semantic segmentation methods, including recently proposed SotA methods like Mask2Former [7] and InternImage [32], with the goal to investigate whether synthetic datasets that are easy to generate can serve as a proxy for a model's real world OOD robustness.

[4] provides a new benchmark for robustness against anomalies, while relevant for real-world applications, we intend to focus this work on traditional OOD robustness.

In their work, Michaelis et al. [28] proposed datasets combining 2D Common Corruptions with datasets such as MS-COCO [25], PASCAL VOC 2007 [12], and Cityscapes. However, their evaluations were limited to 2D Common Corruptions and how different severities of the corruptions on the images impact the downstream task performance. We find correlations between performance against 2D Common Corruptions and real-world corruptions. We use their proposed Cityscapes-C (Cityscapes + 2D Common Corruptions) as our synthetic corruptions dataset.

3. Metrics For Analysis At Scale

This is the first work to analyze semantic segmentation methods, especially under the lens of reliability and generalization ability on such a large scale. The most commonly used metrics for reporting evaluations on seman-

tic segmentation are mean Intersection over Union (mIoU), mean class Accuracy (mAcc), and mean Accuracy of all pixels (aAcc) [1, 2, 36]. We capture these metrics while evaluating models against both ACDC and the 15 2D Common Corruptions on the Cityscapes validation dataset. As per the commonly accepted practice of such OOD evaluations, all models are pre-trained on the Cityscapes training dataset.

Similar to [28], the 15 2D Common Corruptions [19] considered in this work are: 'gaussian noise', 'shot noise', 'impulse noise', 'defocus blur', 'frosted glass blur', 'motion blur', 'zoom blur', 'snow', 'frost', 'fog', 'brightness', 'contrast', 'elastic', 'pixelate', 'jpeg'. Similar to [19], Michaelis et al. [28] shows that synthetic corruptions with corruption severity=1 are too weak, and corruptions with corruption severity=5 are too strong for the downstream task. Thus, we use corruption severity=3 in our evaluations.

As discussed, multiple image classification works [10, 20, 21] and some semantic segmentation works [23, 28] use OOD Robustness of models for evaluating the generalization ability of the method. However, different image corruptions impact the performance of the semantic segmentation methods differently. As we are interested in the worst possible case, we define Generalization Ability Measure (GAM) as the worst mIoU across all image corruptions at a given severity level. That is, we ask the question "For a given dataset, what is the worst possible performance of a given method?". Answering this question tells us about the reliability and generalization ability of a method. We find the minimum of the mIoU of the segmentation masks predicted under image corruptions w.r.t. the ground truth masks for a given method, across all corruptions at a given severity and report this as the $\mathrm{GAM}_{severity\ level}$. For example, for severity=3, the measure would be denoted by GAM_3. The higher the GAM value, the better the generalization ability of the given semantic segmentation method. In Appendix A, we show that our observations are not limited to the mIoU metric and extend to other metrics as well.

4. Analysis And Key Findings

We analyze the correlation in mean performance to determine whether synthetic corruptions can serve as a reliable proxy for real-world corruptions. Additionally, we conduct an in-depth examination of corruption-specific trends, identifying cases where synthetic corruptions effectively mimic real-world effects and where they fall short.

4.1. Are Synthetic Corruptions Useful?

We attempt to study if synthetic corruption like that introduced by [19] does represent the distribution shifts in the real world. While this assumption has driven works such as [19, 23, 24], to the best of our knowledge, it has not yet been proven. Previous works on robustness [15] simply report performance on both, thus, to save compute in the future, we prove this assumption in Fig. 2.

For this analysis, we used methods trained on the training set of Cityscapes and evaluated them on 2D Common Corruptions [19] and the ACDC datasets. ACDC is the Adverse Conditions Dataset with Correspondences, consisting of images from similar regions and scenes as Cityscapes but captured under different conditions such as Day/Night, Fog, Rain, and Snow. These are corruptions in the real world, thus, we attempt to find correlations between performance against synthetic corruptions from 2D Common Corruptions (severity=3) and ACDC. We analyze each common corruption separately and also the mean performance across all 2D Common Corruptions.

In Fig. 2, we observe a very strong positive correlation in performance against ACDC and mean performance across all 2D Common Corruptions. This novel finding helps the community significantly. It means that we do not need to go into the wild to capture images with distribution shifts, as synthetic corruptions serve as a reliable proxy for real-world conditions. Next, we look at the correlation between the worst-case scenario measure using GAM_3 and ACDC. Here, we observe a higher correlation than the previous case, indicating that the performance against worst-case corruption serves as a reliable proxy for real-world corruptions. Lastly, as a sanity check, we find the correlation between mean performance against all corruptions and performance against worse-case corruption to observe a very high correlation. Showing that the two can be used interchangeably.

4.2. When Do Synthetic Corruptions Succeed?

Since some synthetic corruptions attempt to directly mimic the real-world scenarios in ACDC, like changes in lighting due to Day/Night changes or changes in weather due to snowfall or fog, we analyze the correlation of relevant corruptions to ACDC. As discussed in Sec. 4.1, the mean performance correlation is high. However, we observe in Fig. 3 that individual corruptions exhibit varying levels of agreement between synthetic and real-world effects. We observe that the Snow corruption shows a very strong alignment (Pearson correlation 0.867), indicating that synthetic snow corruptions effectively mimic real-world snow-related degradation, despite the corrupted images looking different to a human observer (as shown in Fig. 1).

Brightness (Pearson correlation 0.270) and Fog (Pearson correlation 0.349) exhibit weak alignment, suggesting that synthetic versions of these corruptions fail to fully capture real-world complexities. Specifically, brightness corruptions struggle to model real-world nighttime conditions, while synthetic fog does not accurately represent atmospheric distortions seen in real-world data.

These findings highlight that while synthetic corruptions

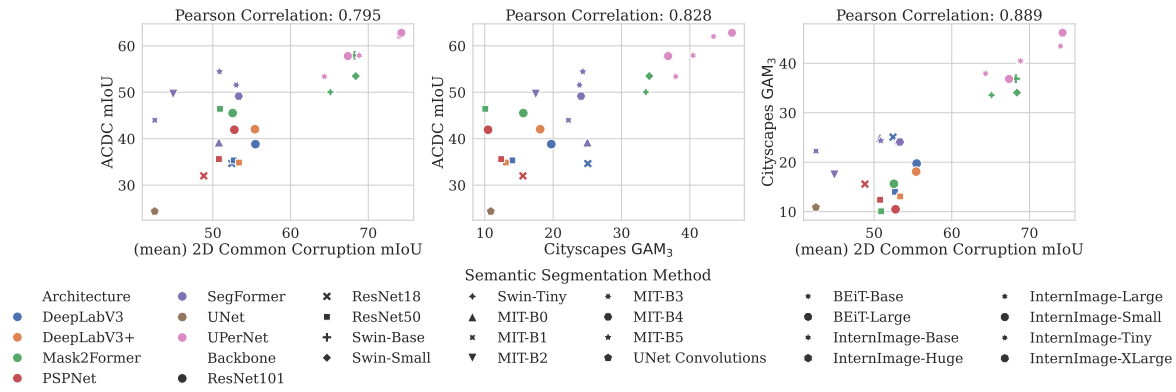

Figure 2. To empirically determine if synthetic common corruptions such as those proposed by [19] truly represent the distribution and domain shifts in the real world, we try to find correlations in evaluations on ACDC and 2D Common Corruptions. Each model is trained on the training dataset of the Cityscapes dataset. Left plot: The y-axis represents values from evaluations on the ACDC dataset, and the x-axis represents mean performance from evaluations on the Common Corruptions at severity=3. We observe a high positive correlation. Centre plot: The y-axis again represents values from evaluations on the ACDC dataset, while the x-axis represents GAM_3, which is the worst performance of the methods across all the Common Corruptions at severity=3. We observe a slightly higher positive correlation. Right plot: serves as a sanity check, where the y-axis represents GAM_3 and the x-axis represents mean performance from evaluations on the Common Corruptions at the same severity. We observe a very high correlation in performance. Thus, given the high positive correlations between performance on the ACDC and mean performance against all synthetic common corruption, we conclude for relative analysis that synthetic corruptions do serve as a reliable proxy for real-world corruptions.

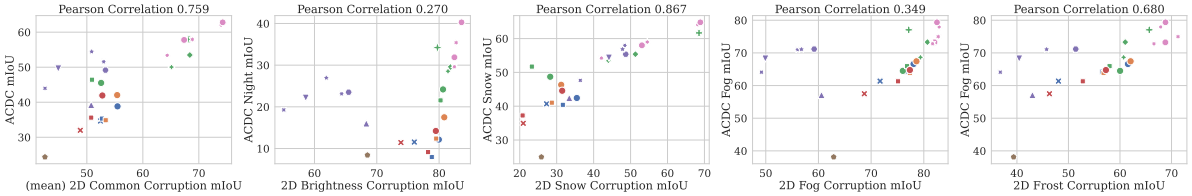

Figure 3. Correlation between model performance (legend as in Fig. 2) on ACDC (real-world corruptions) and 2D Common Corruptions (synthetic) for different corruption types. The left-most plot shows the correlation between mean mIoU across all 2D Common Corruptions and ACDC, with a strong Pearson correlation of 0.759, indicating that synthetic corruptions are generally a reasonable proxy for real-world robustness. The remaining plots analyze specific corruptions: brightness (synthetic) vs. night (real) with correlation 0.270, snow (synthetic) vs. snow (real) with correlation 0.867, fog (synthetic) vs. fog (real) with correlation 0.349, and frost (synthetic) vs. fog (real) with correlation 0.680. While some synthetic corruptions (e.g., snow) closely align with their real-world counterparts, others (e.g., brightness for night) exhibit weaker correlations, highlighting cases where synthetic corruptions may fail as accurate proxies.

can approximate real-world robustness trends, they are not universally reliable across all corruption types.

Interestingly, we observe a moderate positive correlation (Pearson correlation 0.680) in performance against ACDC Fog and 2D Common Corruption Frost. Since the Frost 2D Common Corruption involves superimposing a randomly chosen frost image on the input image with some transparency, one might hypothesize that the model finds the distribution shifts between the two to be moderately similar.

5. Conclusion

Our study provides the most comprehensive benchmarking to date on the reliability of synthetic corruptions as a proxy for real-world distribution shifts in semantic segmentation. Through extensive experiments, we observe a strong correlation in mean performance between synthetic and real-world corruptions, supporting their utility for robustness evaluation. However, a deeper analysis of individual corruption types reveals that while some synthetic corruptions (e.g., snow) closely align with real-world performance, others (e.g., brightness, fog) exhibit weak correlations, highlighting gaps in current benchmarking approaches.

These findings underscore the importance of refining synthetic corruption benchmarks to better capture real-world conditions. To promote OOD evaluations on synthetic datasets, we provide benchmarking of all 15 2D Common Corruptions on the most commonly used semantic segmentation datasets, namely, Cityscapes, ADE20k, and PASCAL VOC2012 datasets. We release our datasets and code to facilitate further research in this direction.

References

[1] Shashank Agnihotri, Steffen Jung, and Margret Keuper. CosPGD: an efficient white-box adversarial attack for pixel-wise prediction tasks. In *Proc. International Conference on Machine Learning (ICML)*, 2024. 3, 8

[2] Anurag Arnab, Ondrej Miksik, and Philip HS Torr. On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 888–897, 2018. 3

[3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 10

[4] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 2

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017. 10

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 9, 10

[7] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 1, 2, 7, 8, 9, 10, 12

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 9

[9] 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. 1, 2, 8, 12

[10] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 3

[11] Razvan-Gabriel Dumitru, Darius Peteleaza, and Catalin Craciun. Using duck-net for polyp image segmentation. *Scientific reports*, 13(1):9803, 2023. 2

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2): 303–338, 2010. 2

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 2, 9

[14] Jindong Gu, Hengshuang Zhao, Volker Tresp, and Philip HS Torr. Segpgd: An effective and efficient adversarial attack for evaluating and boosting segmentation robustness. In *ECCV*, pages 308–325. Springer, 2022. 9

[15] Yong Guo, David Stutz, and Bernt Schiele. Robustifying token attention for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17557–17568, 2023. 3

[16] Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011. 9

[17] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015. 9

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 9, 10

[19] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *Proc. International Conference on Learning Representations (ICLR)*, 2019. 1, 2, 3, 4, 9, 12

[20] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proc. of the International Conference on Learning Representations (ICLR)*, 2020. 2, 3

[21] J Hoffmann, S Agnihotri, Tonmoy Saikia, and Thomas Brox. Towards improving robustness of compressed cnns. In *ICML Workshop on Uncertainty and Robustness in Deep Learning (UDL)*, 2021. 2, 3

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 9

[23] Christoph Kamann and Carsten Rother. Benchmarking the robustness of semantic segmentation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8828–8838, 2020. 2, 3, 8, 9

[24] Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18963–18974, 2022. 2, 3, 9

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 2

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In

Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021. 1, 10, 12

[27] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015. 2

[28] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S. Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019. 2, 3

[29] Patrick Müller, Alexander Braun, and Margret Keuper. Classification robustness to common optical aberrations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3632–3643, 2023. 9

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 2, 9, 10

[31] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 12

[32] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14408–14419, 2023. 2, 9, 10

[33] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 10

[34] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34: 12077–12090, 2021. 7, 8, 10

[35] Hengshuang Zhao. semseg. `https://github.com/hszhao/semseg`, 2019. 9

[36] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017. 3, 8, 9, 10

[37] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019. 2, 8

Are Synthetic Corruptions A Reliable Proxy For Real-World Corruptions?

Paper #0005 Supplementary Material

Table Of Content

The supplementary material covers the following information:

A. Correlation In Metrics

Here, we provide a comparison of mean accuracy across synthetic (2D Common Corruptions) and real-world (ACDC) corruptions. The top plot presents mAcc (mean class accuracy) with a stronger correlation of 0.782–0.858, while the bottom plot shows results for aAcc (all pixel accuracy) with a Pearson correlation of 0.688–0.767. These results indicate that synthetic corruptions serve as a reasonable proxy for real-world robustness. Thus, the analysis made using mIoU would also hold if made using other metrics.

B. Implementation Details Of The Benchmarking

Following, we provide details regarding the experiments done for creating the benchmark used in the analysis.

B.1. Compute Resources.

Most experiments were done on a single 40 GB NVIDIA Tesla V100 GPU each, however, SegFormer [34] and Mask2Former [7] with large backbones are more compute-intensive, and thus 80GB NVIDIA A100 GPUs or NVIDIA H100

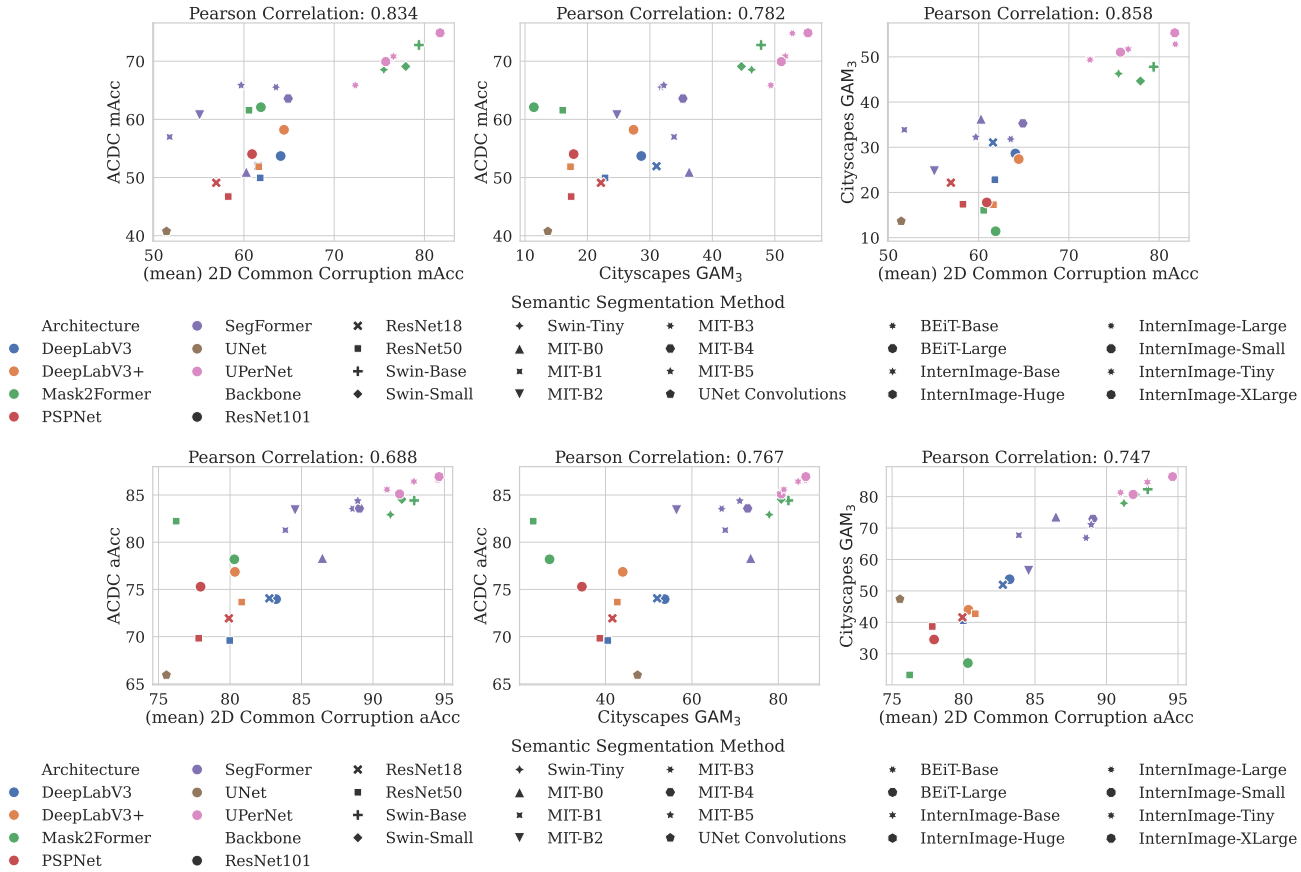

Figure 4. Comparison of mean accuracy across synthetic (2D Common Corruptions) and real-world (ACDC) corruptions. The top plot presents mAcc (mean class accuracy) with a stronger correlation of 0.782–0.858, while the bottom plot shows results for aAcc (all pixel accuracy) with a Pearson correlation of 0.688–0.767. These results indicate that synthetic corruptions serve as a reasonable proxy for real-world robustness, even when measured using metrics other than mIoU

were used for these models, a single GPU for each experiment. Training some of the architectures with large backbones required using two to four GPUs in parallel.

B.2. Dataset Details

Performing OOD robustness evaluations is very expensive and compute-intensive. Thus, for the benchmark, we only use ADE20k, Cityscapes, and PASCAL VOC2012 as these are the most commonly used datasets for evaluation [1, 7, 23, 34, 36].

B.2.1. ADE20K

ADE20K [37] dataset contains pixel-level annotations for 150 object classes, with a total of 20,210 images for training, 2000 images for validation, and 3000 images for testing. Following common practice [1, 34] we evaluate using the validation images.

B.2.2. Cityscapes

The Cityscapes dataset [9] comprises a total of 5000 images sourced from 50 different cities in Germany and neighboring countries. The images were captured at different times of the year and under typical meteorological conditions. Each image was subject to pixel-wise annotations by human experts. The dataset is split into three subsets: training (2975 images), validation (500 images), and testing (1525 images). This dataset has pixel-level annotations for 30 object classes.

B.2.3. PASCAL VOC2012

The PASCAL VOC 2012 [13], contains 20 object classes and one background class, with 1464 training images, and 1449 validation images. We follow common practice [14, 17, 35, 36], and use work by Hariharan et al. [16], augmenting the training set to 10,582 images. We evaluate using the validation set.

Calculating the mIoU. mIoU is the mean Intersection over Union of the predicted segmentation mask with the ground truth segmentation mask.

B.3. Models Used

Table 1 presents a comprehensive reference table for all semantic segmentation models used in our benchmarking. These methods include some of the first efforts in DL-based semantic segmentation methods like UNet [30], and some of the most recent SotA methods like InterImage [32]. Each model is trained on the respective training subset of its dataset and evaluated on the corresponding validation set. The evaluations on 2D Common Corruptions are conducted using the validation sets.

C. 2D Common Corruptions

[19] propose introducing a distribution shift in the input samples by perturbing images with a total of 15 synthetic corruptions that could occur in the real world. These corruptions include weather phenomena such as fog, and frost, digital corruptions such as jpeg compression, pixelation, and different kinds of blurs like motion, and zoom blur, and noise corruptions such as Gaussian and shot noise amongst others corruption types. Each of these corruptions can perturb the image at 5 different severity levels between 1 and 5. The final performance of the model is the mean of the model's performance on all the corruptions, such that every corruption is used to perturb each image in the evaluation dataset. Since these corruptions are applied to a 2D image, they are collectively termed 2D Common Corruptions.

We show examples of perturbed images over some corruptions and the changed predictions in Figure 5.

In Figure 6, we extend the visualizations from Figure 1, additionally showing Night and Rain for ACDC, and Brightness and Frost for 2D Common Corruptions.

D. Benchmarking Results

Following, we include the results from the 2D Common Corruptions evaluations of all the semantic segmentation methods over all of the common corruptions, for PASCAL VOC2012 in Figure 7, for Cityscapes in Figure 8, and for ADE20K in Figure 9.

E. Extension To The Related Work

Kamann and Rother [23] provide an OOD robustness benchmark for semantic segmentation. While they use multiple backbone architectures, such as variants of ResNet [18], MobileNet [22], and Xception [8], their evaluations are limited to the DeepLabV3+ [6] architecture. Our evaluated benchmark extends to multiple architectures and backbones, including recently proposed SotA methods like Mask2Former [7] and InternImage [32].

F. Future Work

Distribution shifts in the real world can be caused by multiple factors, one such factor is lens aberrations. [29] presents many such lens aberrations. Additionally, Kar et al. [24] recently proposed 3D Common Corruptions that take scene depth into account to make corruptions more realistic-looking. We intend to extend our analysis to include these, enabling a more comprehensive robustness study.

F.1. Limitations

Benchmarking the robustness of semantic segmentation methods is a computationally and labor-intensive endeavor. Thus, best utilizing available resources, we benchmark a limited number of settings. While more evaluations like correlation with different severity levels would be interesting, this is the most comprehensive robustness benchmark to date and instills interest to further improve our synthetic corruptions.

Table 1. An Overview of all the semantic segmentation methods used in the benchmark in this work made using SEMSEGBENCH. Each of the mentioned backbones has been evaluated using each of the architectures and datasets mentioned in the row in this table.

Backbone	**Architecture**	**Datasets**	**Time Proposed (yyyy-mm-dd)**
ResNet101 [18]	DeepLabV3 [5], DeepLabV3+ [6], Mask2Former [7], PSPNet [36]	ADE20K, Cityscapes, PASCAL VOC 2012	2017-12-05
ResNet18 [18]	DeepLabV3 [5], DeepLabV3+ [6], PSPNet [36]	Cityscapes	2017-12-05
ResNet50 [18]	DeepLabV3 [5], DeepLabV3+ [6], Mask2Former [7], PSPNet [36]	ADE20K, Cityscapes, PASCAL VOC 2012	2017-12-05
Swin-Base [26]	Mask2Former [7]	ADE20K, Cityscapes, PASCAL VOC 2012	2022-06-15
Swin-Small [26]	Mask2Former [7]	ADE20K, Cityscapes, PASCAL VOC 2012	2022-06-15
Swin-Tiny [26]	Mask2Former [7]	ADE20K, Cityscapes, PASCAL VOC 2012	2022-06-15
MIT-B0 [34]	SegFormer [34]	ADE20K, Cityscapes, PASCAL VOC 2012	2021-10-28
MIT-B1 [34]	SegFormer [34]	ADE20K, Cityscapes, PASCAL VOC 2012	2021-10-28
MIT-B2 [34]	SegFormer [34]	ADE20K, Cityscapes, PASCAL VOC 2012	2021-10-28
MIT-B3 [34]	SegFormer [34]	ADE20K, Cityscapes, PASCAL VOC 2012	2021-10-28
MIT-B4 [34]	SegFormer [34]	ADE20K, Cityscapes, PASCAL VOC 2012	2021-10-28
MIT-B5 [34]	SegFormer [34]	ADE20K, Cityscapes, PASCAL VOC 2012	2021-10-28
UNet Convolutions	UNet [30]	Cityscapes	2015-05-18
BEiT-Base [3]	UPerNet [33]	ADE20K	2022-09-03
BEiT-Large [3]	UPerNet [33]	ADE20K	2022-09-03
InternImage-Base [32]	UPerNet [33]	ADE20K, Cityscapes, PASCAL VOC 2012	2023-04-17
InternImage-Huge [32]	UPerNet [33]	ADE20K	2023-04-17
InternImage-Large [32]	UPerNet [33]	ADE20K, Cityscapes	2023-04-17
InternImage-Small [32]	UPerNet [33]	ADE20K, Cityscapes, PASCAL VOC 2012	2023-04-17
InternImage-Tiny [32]	UPerNet [33]	ADE20K, Cityscapes, PASCAL VOC 2012	2023-04-17
InternImage-XLarge [32]	UPerNet [33]	ADE20K, Cityscapes	2023-04-17

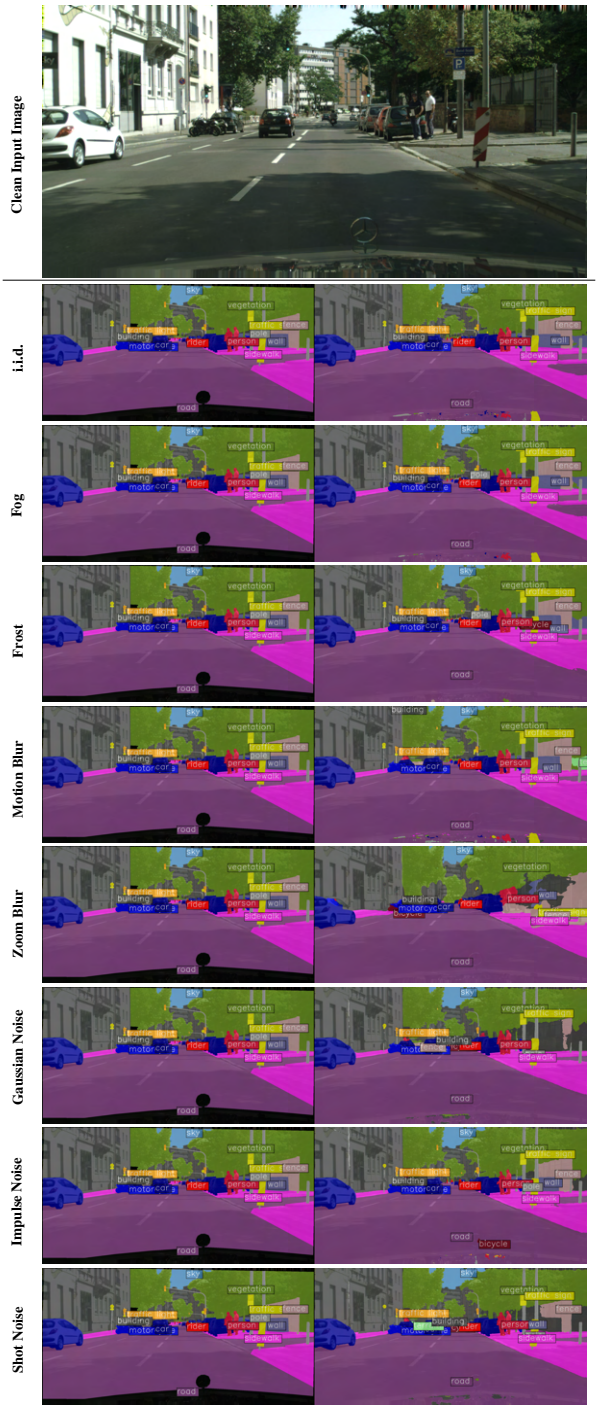

Figure 5. Illustrating changes in prediction due to different 2D Common Corruptions on a randomly chosen input image from the **Cityscapes dataset**, when attaching the semantic segmentation method **InterImage-Base**. In the subfigures with semantic segmentation mask predictions, **Left: Ground Truth Mask**, and **Right: Predicted Mask**.

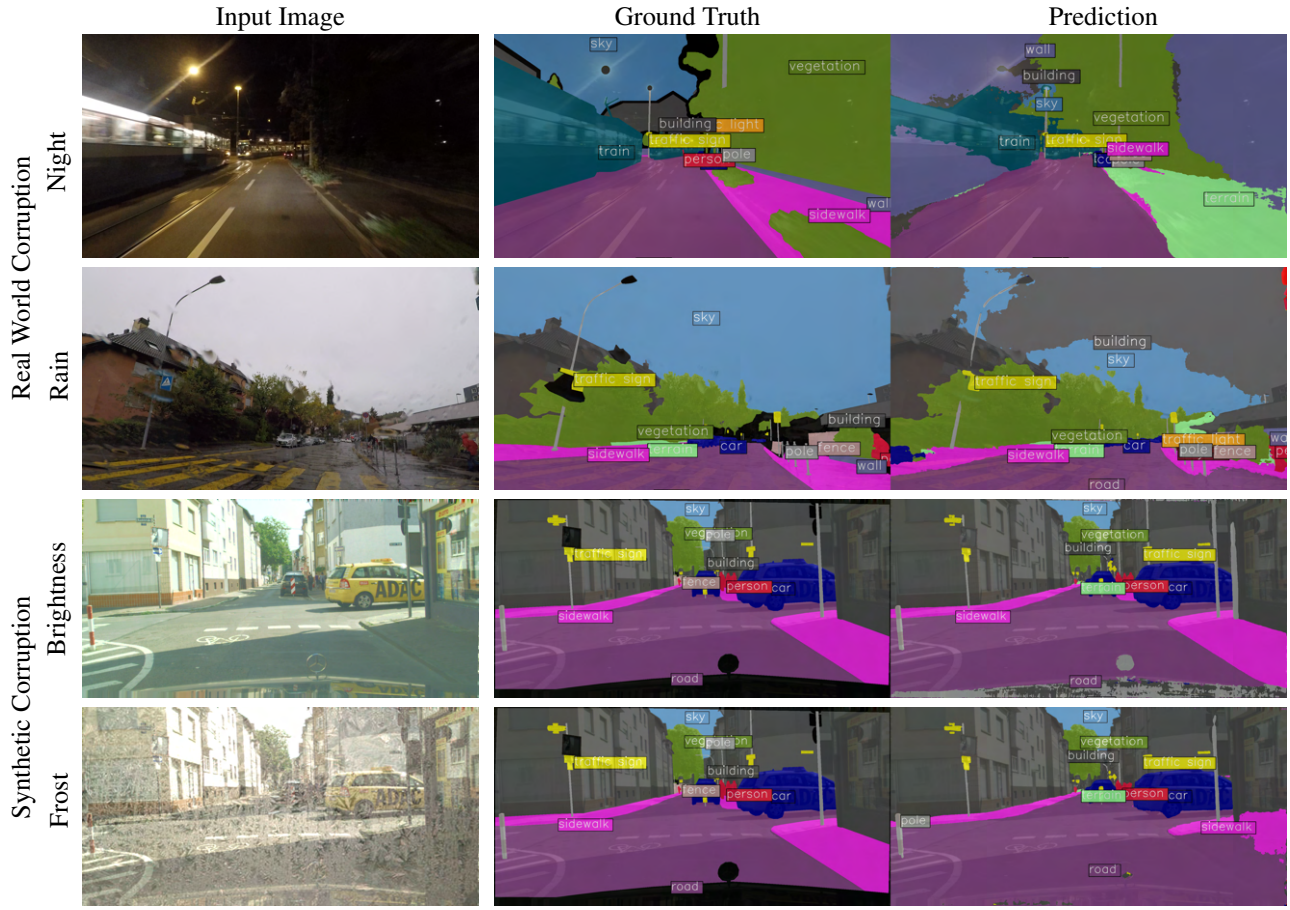

Figure 6. An extension to Figure 1, comparing images with weather corruptions captured in the wild (ACDC [31] and images corrupted using synthetic corruptions [19] and the predictions using a Mask2Former [7] with a Swin-Base [26] backbone trained on the Cityscapes [9] dataset.

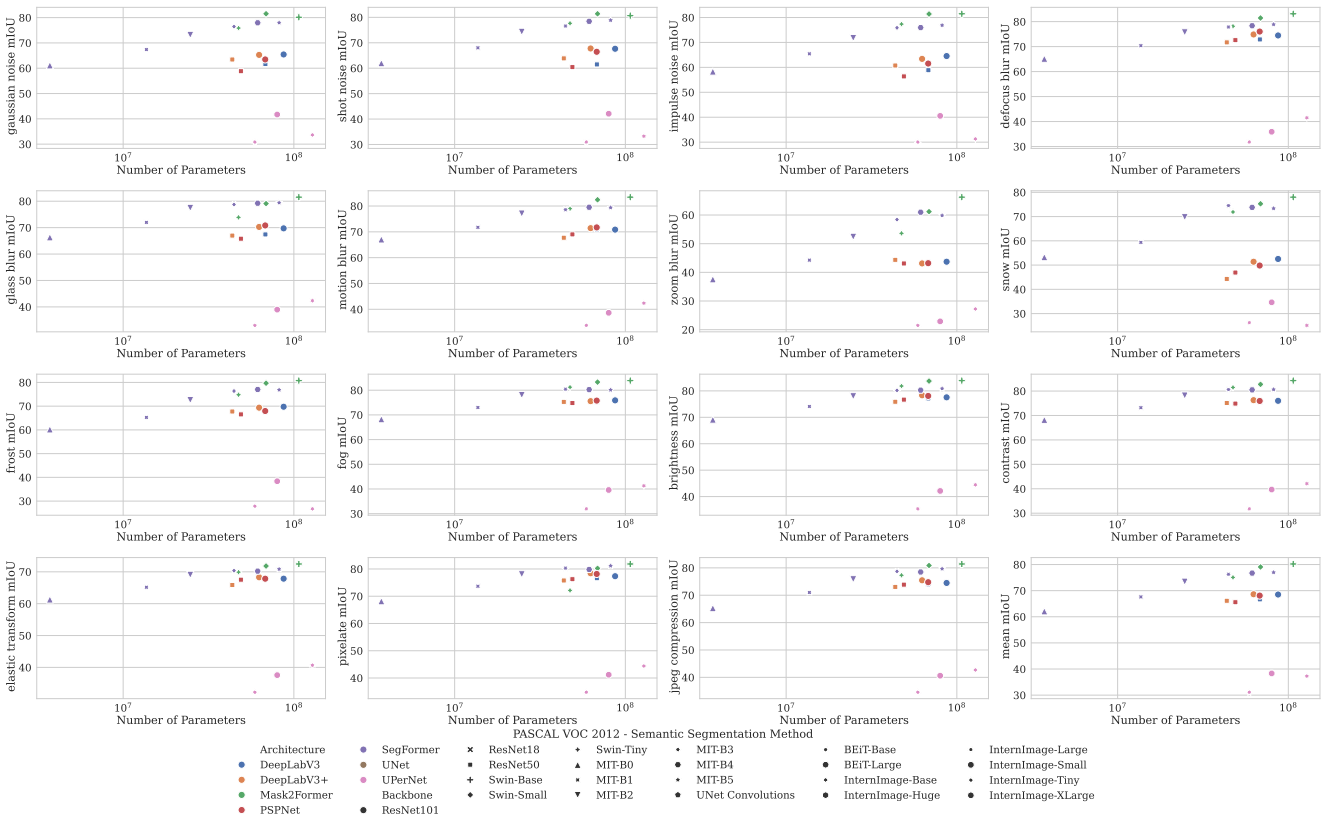

Figure 7. **Dataset used: PASCAL VOC2012**. The correlation in the performance of semantic segmentation methods against different 2D Common Corruptions. The respective axis shows the name of the common corruption used. Colors are used to show different architectures and marker styles are used to show different backbones used by the semantic segmentation methods. For the limited PASCAL VOC2012 evaluations we observe some correlation between the number of learnable parameters and the performance against common corruptions, however, more evaluations (more publicly available checkpoints) are required for a meaningful analysis.

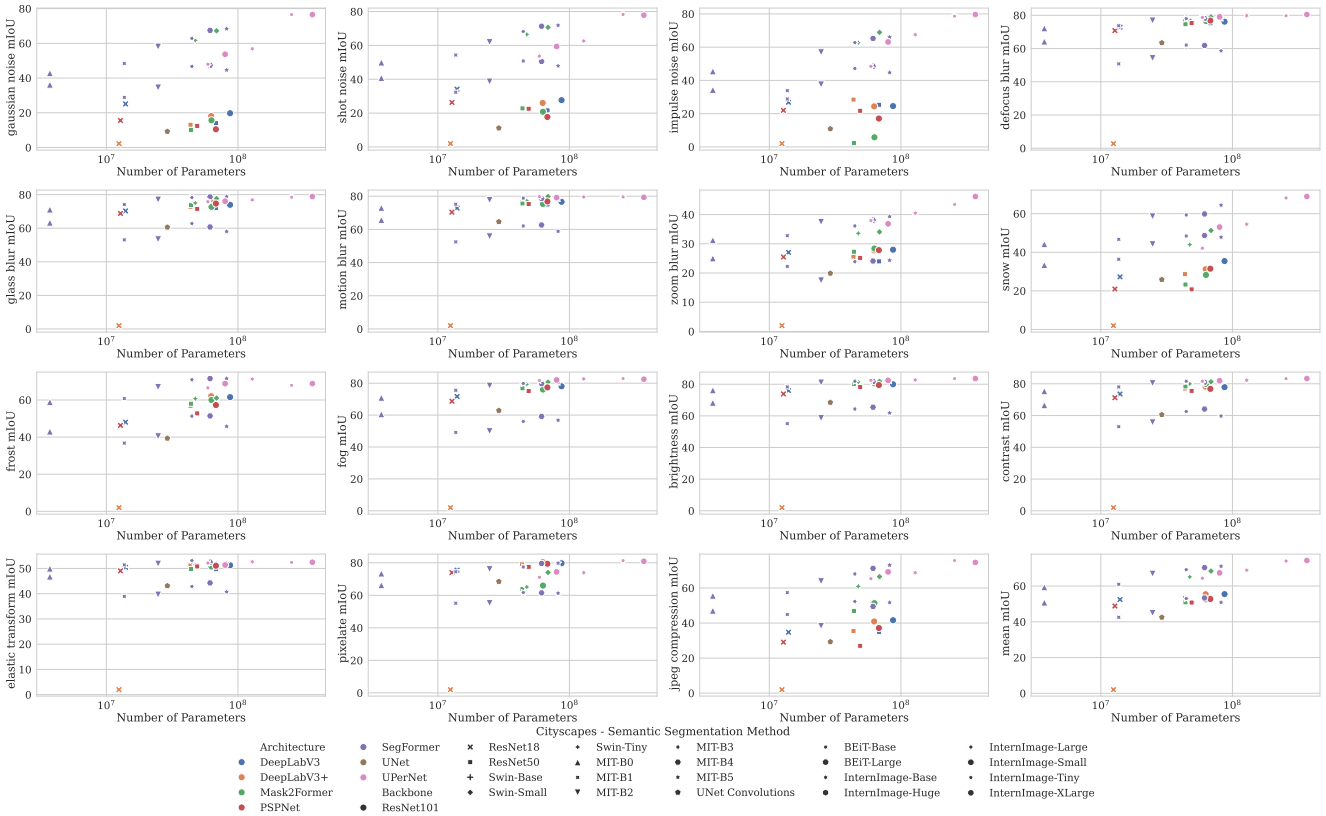

Figure 8. **Dataset used: Cityscapes**. The correlation in the performance of semantic segmentation methods against different 2D Common Corruptions. The respective axis shows the name of the common corruption used. Colors are used to show different architectures and marker styles are used to show different backbones used by the semantic segmentation methods. Except for DeepLabV3+ with a ResNet18 backbone, most other methods show a weak positive correlation between the number of learnable parameters used by a method and its performance against most of the common corruption. Multiple occurrences of an Architecture and Backbone pair are due to their evaluations being performed at two different crop sizes i.e. 512×512, and 512×1024.

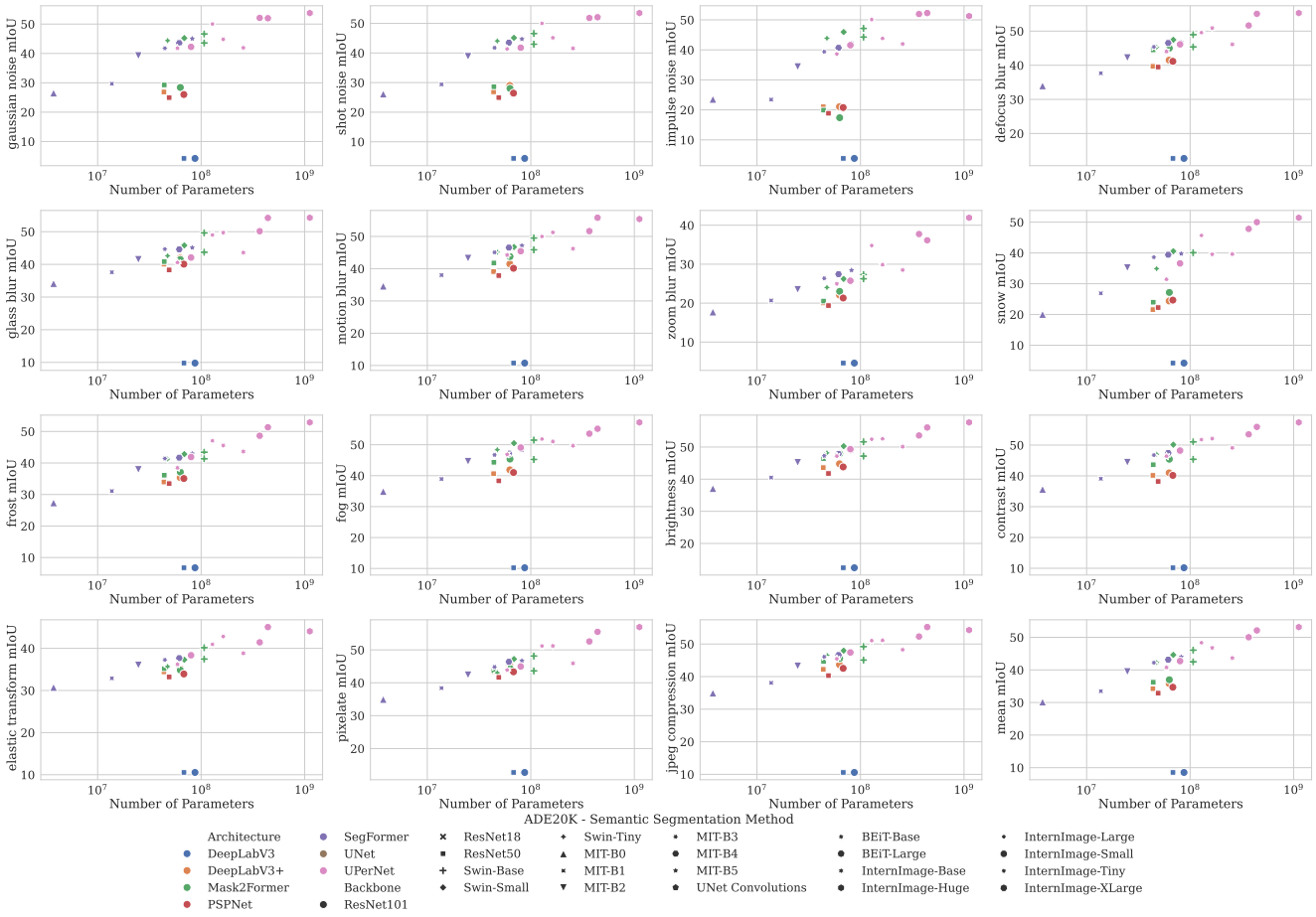

Figure 9. **Dataset used: ADE20K**. The correlation in the performance of semantic segmentation methods against different 2D Common Corruptions. The respective axis shows the name of the common corruption used. Colors are used to show different architectures and marker styles are used to show different backbones used by the semantic segmentation methods. Except for DeepLabV3, all other methods show some positive correlation between the number of learnable parameters used by a method and its performance against any common corruption.