UNIVERSITY OF CALIFORNIA
SANTA CRUZ

A FRAMEWORK FOR GENERATING DANGEROUS SCENES: TOWARDS EXPLAINING REALISTIC DRIVING TRAJECTORIES

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Shengjie Xu

June 2023

The Thesis of Shengjie Xu
is approved:

Professor Mircea Teodorescu, Chair

Professor Leilani H. Gilpin

Professor James E. Davis

Peter Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

A Framework for Generating Dangerous Scenes: Towards Explaining Realistic Driving Trajectories

by

Shengjie Xu

Deep neural networks are black box models that are hard to interpret by humans. However, organizations developing AI models must ensure transparency and accountability by providing the public with a comprehensive understanding of model functionality. We suggest integrating explainability information as feedback during the development, verification, and testing of models. Our testing framework provides the following insight during the neural network training: Does the model equally effective for minor variations in the input data? In this thesis, we showed the explainability differences by comparing original and altered autonomous driving datasets for neural network training and explainability. We propose a framework for perturbing autonomous vehicle datasets, the DANGER framework, which generates edge-case images on top of current autonomous driving datasets. The inputs to DANGER are photorealistic datasets from real driving scenarios. We present the DANGER algorithm for vehicle position manipulation and the interface towards the renderer module and present five scenario-level dangerous primitives generation applied to the virtual KITTI, virtual KITTI 2, and Waymo datasets. We contribute two innovations in our study: (a) Our experiments prove that DANGER can be used as a framework for expanding the current datasets

to cover generative while realistic and anomalous corner cases; (b) We tested the feasibility of providing interpretable information feedback in a generic deep neural network training by providing the Grad-CAM instability level.

Acknowledgments

I would like to express my heartfelt gratitude to Professor Mircea Teodorescu and Professor Leilani H. Gilpin for their invaluable guidance throughout this research study. I am deeply thankful to Prof. Teodorescu for our enlightening conversations in the bio-inspired robotics classroom and for his advice during our journey to Science Hill. I am also grateful to Professor Gilpin for her encouragement, mentorship, and invaluable advice during my time at AIEA lab. The spirit of mentorship at MIT CSAIL, the discussions about Marvin Minsky, and our Emotion Machine or AI talks will forever hold a special place in my memory.

I extend my sincere appreciation to Professor James E. Davis for serving on my thesis reading committee and for providing valuable insights and research advice during my time in the AVIS lab. Your mentorship has greatly enriched my academic interests in the field of autonomous vehicles.

I am immensely grateful to the University of California, Santa Cruz, for granting me the opportunity to study and thrive in this beautiful environment. I will fondly remember the breathtaking ocean view from the entrance of Cowell and Stevenson Dining Hall, the summer night concert in Quarry Amphitheater, the serene bike path through the Great Meadow, the enchanting redwoods surrounding Baskin School of Engineering, and the delightful sight of seals resting on the wharf as I sailed on Monterey Bay.

Lastly, I would like to express my heartfelt appreciation to my parents and

friends for their unwavering support and love, especially during the challenging times of the pandemic. Your presence and encouragement have been my pillar of strength. I am forever grateful to my fellows Yixuan Liu, Yike Wei, Jiahao Luo, Lan Mi, Li Liu, Sijia Zhong, and Bo Yang for their support and friendship on my way leading to the east coast.

Chapter 1

Introduction

Machine learning models are widely accepted in academia as black boxes that are extremely difficult to interpret [32, 80]. However, organizations and professionals are obliged to thoroughly understand the underlying Artificial Intelligence (AI) with model observing and accountability of AI and not trust them without being able to recount and defend their decisions [41, 106]. According to Gunning [36], the goal of Explainable Artificial Intelligence (XAI) is to develop a set of modified machine learning techniques that produce explainable models that enable end-users to comprehend, trust, and manage the generation of AI systems. With a transparent (or interpretable) model, the industry can use continuous model evaluation to compare predictions, quantify model risk, and optimize model performance. Moreover, XAI is valuable in improving model performance while also assisting stakeholders in understanding the behaviors of AI models. This allows researchers to visually study model through interactive charts.

Therefore, we suggest integrating explainability information as feedback during model development, verification, and testing. Our proposed testing framework provides the following insight during the neural network training: *Does the model equally effective for minor variations in the input data?* As shown in Figure 1.1, the main objective of this research is to test the feasibility of providing interpretable information feedback in a generic deep neural network training, which is divided into two sections: image generation with minor modifications or errors and providing evidence of interpretable instability under input perturbation.

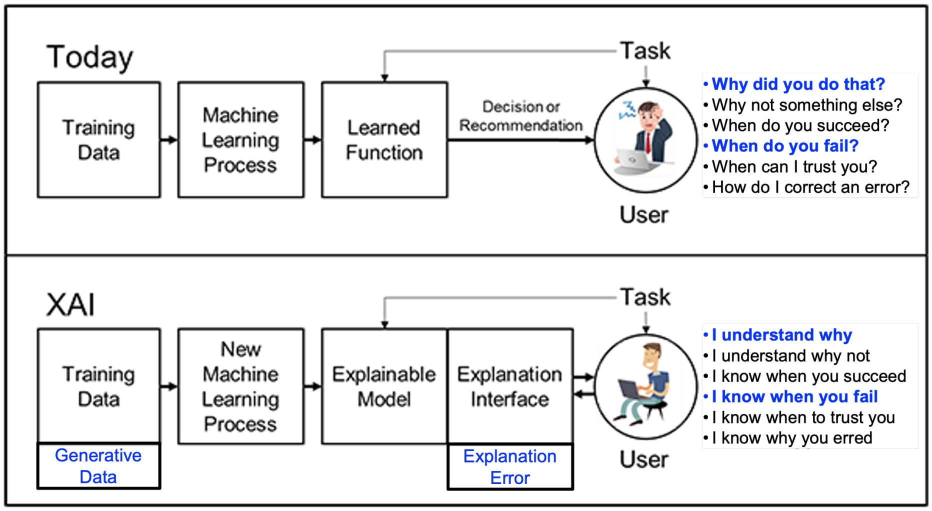


Figure 1.1: XAI concept illustrated in [36] and our proposed explanation analysis exploration in this work

The first section of of this thesis discusses the generation of photo-realistic scene-based out-of-distribution datasets. Today's deep neural networks are highly dependent on goodness-of-data and obey the "garbage in, garbage out" rule of thumb [50, 96]. In the absence of standardized metrics to characterize "the goodness" of data, conventional visual perception methods are often not able to detect dangerous scenarios. This is because corner cases have not been witnessed during training [8, 93]. Fitting metrics do not further represent the phenomenological fidelity and validity of the data. Their detection is based on well prepared data, lacking anomalous events: low possibility but realistic dangerous driving scenarios.

Chapter 2 presents an in-depth exploration of 2D and 3D image synthesis. Since the release of the KITTI [31] dataset, autonomous driving research has been data-driven. Autonomous vehicles (AVs) promise to decrease vehicle fatalities and increase safety in the modern automobile. However, the majority of datasets [11, 18, 31, 92, 102] and derived algorithms [12, 20, 30, 33, 76, 78, 101, 104, 109] are used for benchmark and standardized on perfectly curated datasets. This causes two issues: (a) these algorithms are specially designed and hard-coded into a workable scenario (b) solutions are focused on accuracy on a single dataset, rather than robustness. Instead, we want autonomous driving solutions to be able to deal with different driving scenarios. A real road scene is often dangerous and dynamic: full of unexpected events. There is an immediate need for AI systems to consider these event, especially in the context of implementing algorithms on actual on-road vehicles [87]. Instead, we propose the development of a framework to mimic the kinds of "one-off" scenarios humans may encounter in driving tests. We use this to validate that AI systems can generalize in real-world driving environments. We describe an iterative procedure for creating out-of-domain examples for autonomous driving, or corner cases, based on the existing AV datasets.

In Chapter 3, we build upon the theoretical review established in Chapter 2 and present our DANGER framework. We contribute a general framework for generating photo-level realistic driving scenes with custom trajectory inputs. We generated synthetic datasets for *Virtual KITTI* (vKITTI) [29] based on driving primitives as the input dataset. The new dataset is comprised of five categories of images, which are derived from the vKITTI dataset and built on top of the following primitives: (a) `Exit parking`, (b) `Cut-in Opposite`, (c) `Cut-in`, (d) `Slalom Lane Change`, and (e) `Braking`. We also generated new datasets from *Virtual KITTI 2* [10] and *Waymo Open Dataset* (WOD) [92] to demonstrate the versatility of our framework to real-world scenarios. Our initial perturbation on Waymo shows promise, and also room for improvement for creating more photo realistic DANGER de-rendering. Our framework, "DANGER", supports user-defined vehicle trajectories and poses to complete a sequence of frames of data generation. DANGER also supports the distortion and deletion of vehicles in an individual frame and can simulate illogical special camera failure modes.

In Chapter 4, we discuss the quantitative and user study of our DANGER framework as well as the interpretability metrics of commonly used deep learning models, specifically analyzing the stability of saliency mapping attribution under the disturbance of our generative datasets. In this thesis, we showed the explainability differences by comparing original and altered autonomous driving datasets for neural network training and explainability. Given a synthesis datasets, we analyzed the Grad-CAM explanation towards a segmentation model on the datasets pair, and we discovered that temporal instability, heatmap amplitude instability, and instability in synthesized data.

The findings of this study indicate the following contributions to the explainable AI community:

- We introduce DANGER, a framework for generating Danger-Aware datasets. DANGER can enhance robustness. It generates new scenarios with user input: a set of primitives. Each primitive is a vehicle driving trajectory and posture over time: a complete sequence of frames of data generation.
- DANGER supports the shifting and deletion of cars in individual frame and can simulate illogical special camera failure modes.
- Our DANGER implementation includes five scenario-level dangerous primitives applied on virtual KITTI and virtual KITTI 2 to generate more robust, “DANGER-vKITTI” datasets.
- We evaluate DANGER on a corner case score evaluation and via human study. Our results demonstrate that DANGER and our generated scenarios are realistic, novel, anomalous, or risky. These dataset augmentations can help increase the robustness and range of scenarios in the original datasets.
- We test the feasibility of providing interpretable information feedback in a generic deep neural network training by showing the Grad-CAM differences.

Chapter 2

Background

This section provides an overview of deep generative models, GANs, autoencoders, image synthesis techniques, 3D-aware image synthesis, relevant datasets, and explainability in AI models. We first discuss the topic of deep generative models in computer vision and focus on the use of generative adversarial networks (GANs) and autoencoders for image synthesis. Deep generative models are a rapidly evolving field that aims to generate realistic examples across various domains. Next, we explore 2D image synthesis and 3D-aware image synthesis, highlighting studies that have developed GAN-based methods for generating 3D-aware images. Thirdly, we briefly touch on datasets used in autonomous driving research. Finally, the section briefly introduces Explainable Artificial Intelligence (XAI) and discusses the promising future of explainability in bridging the gap between complex models and human understanding to improve AI robustness.

2.1 Deep Generative Models

The computer vision community has been taking various approaches to generate photorealistic photos of objects, scenes that even humans cannot tell are fake. Deep generative models are an exciting and rapidly evolving field that fulfills the promise of generative models by generating realistic examples across a wide range of problem domains.

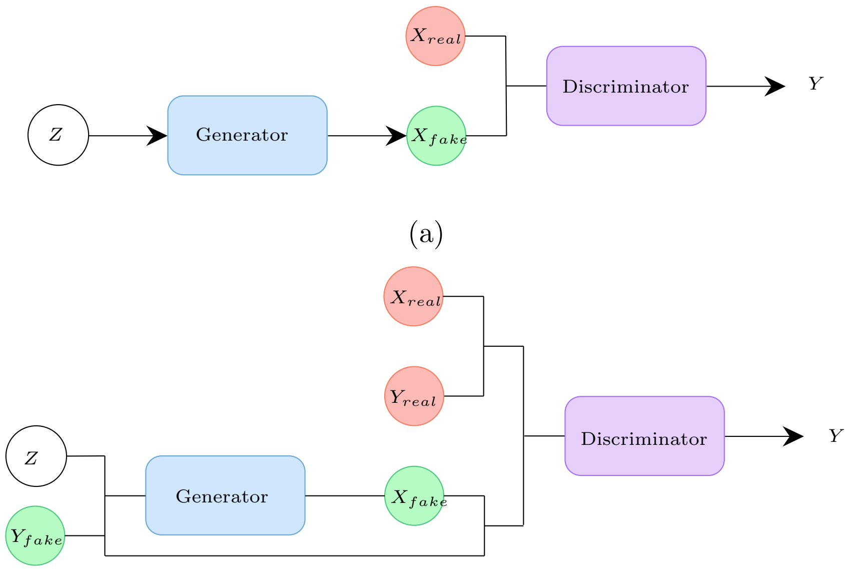
2.1.1 Generative Adversarial Networks

The first work in this line is based on *Generative Adversarial Networks* (GANs) [34, 45–47]. The training of GANs' generative models frames this unsupervised learning algorithm by using a supervised loss with two sub-models as part of the training: *the generator model* and *the discriminator model*. In an adversarial zero-sum game, the two models are trained together until the discriminator model is fooled about half of the time, indicating that the generator model generates realistic examples.

2.1.2 Autoencoder and Variational Autoencoders

Autoencoder is a type of artificial neural network that learns efficient codings of unlabeled data using unsupervised learning. The autoencoder learns a representation for a set of data by attempting to regenerate the input from the encoding via dimensionality reduction [40, 54]. Autoencoder can be compared to *principal component analysis* (PCA), the specific analysis referred to [66].

Variational Autoencoders (VAE) is an autoencoder whose training is regu-



(a)

(b)

Figure 2.1: Schematic of (a) GAN and (b) cGAN models, showing Generator and Discriminator components and associated variables

larised to avoid overfitting and ensure that the latent space has good properties that enable a generative model: they can randomly generate new data that is similar to the input training data [52, 53].

2.2 Image Synthesis

2.2.1 GANs for 2D Image Synthesis.

Generative Adversarial Networks (GANs) [34], have been used for a variety of image synthesis exercises, including image generation [3, 44, 72], image-to-image translation [42, 110], text-to-image synthesis [77, 107], and inpainting [70]. StyleGAN [46] generates high-quality, high-resolution face images. StyleGan can also generate car-like images; however, flaws remain in the generated dataset. Firstly, the generated images were frequently displayed at a 45-degree exhibition angle rather than the perspective view of a car in motion. Second, the shape of some of the cars was distorted and produced a peculiar effect of indistinguishable front and rear of the car. StyleGAN2 [47] fixed the artifacts problem remained in StyleGAN. However, both StyleGAN nor StyleGAN2 cannot generate images by controlling car object-dependent appearance, pose, and size in 3D since the latent space is unclear.

Recent studies [73, 74, 81] showed the SOTA GPT-based methods of generating photorealistic images from text. Nevertheless, the above methods generate images that have low fidelity and cannot generate continuous frame images or videos.

Several studies [22, 75] learned transformation and color adjustment features, such as rotate, shear, shear, contrast, or, hue. These strategies, however, are limited to 2D images or color space and are challenging to generate new viewpoint data. Contrary to 2D, we need a disentangled 3D-aware image synthesis model that allows a user to edit the viewpoint, object shape, or texture independently [111].

StyleGAN uses the progressive growth idea to stabilize the training for high-resolution images, and it generates some of the most indistinguishable face images available today [46]. The *Style* refers to the attributes related to a face in the dataset, such as the pose of the person, expression, orientation, hairstyle, and also includes the texture details in terms of skin color, lighting, etc.

Although Karras et al. [46] demonstrated that StyleGAN could be used to generate car-like images, we discovered flaws in the generated dataset: first, the generated images were frequently displayed at a 45-degree exhibition angle rather than the perspective view of a car in motion; second, the shape of some of the cars was distorted and produced a peculiar effect of indistinguishable front and rear of the car.

Despite the fact that Karras et al. [47] demonstrated that StyleGAN2 fixes the artifacts problem remained in StyleGAN and improves the quality of the generated images by proposing a new alternative to progressive growing, we discovered that generation defects in car images still exist.

Pix2pix [42] is a conditional generative adversarial network (cGAN) framework, which is first proposed in [64]. It takes pairs of images as input, where one image belongs to the source domain and the other image belongs to the target domain. The model learns to map images from the source domain to the target domain by training on aligned image pairs and optimizing a combination of adversarial and pixel-wise loss functions. This allows for generating realistic and high-quality images in the target domain. pix2pixHD [99] builds upon the pix2pix framework and extends it to generate high-resolution images. It incorporates a multi-scale generator network that progressively refines the output image at different resolutions. By considering multiple scales and incorporating global and local context, pix2pixHD achieves more detailed and visually appealing results compared to its predecessor. Both pix2pix and pix2pixHD have been widely used in various applications, including image-to-image translation, image synthesis, and image editing. These models have demonstrated the ability to learn meaningful mappings between different visual domains, enabling tasks such as converting sketches to realistic images, converting day-to-night images, or generating high-resolution images from low-resolution inputs.

2.2.2 3D-aware Image Synthesis

Several recent studies [57, 85, 105] developed multiple GAN-based 3D-aware image synthesis methods including 3D-friendly features: interpretable, disentangled scene representation, viewpoint manipulation, and 3D Controllable. [57, 103], and [35] inspired by rendering a 2D image by 3D game engine, which treats images as a projection of the 3D world. The learned 3D space can be potentially useful for various tasks such as image reasoning tasks in dangerous scene understanding.

Inspired by [63], GRAF [85] introduces GANs to implement Neural Radiance Fields [63], and uses conditional GAN [64] to achieve controllability of the rendered object. GIRAFFE [68] uses one Neural Radiance Field per object in order to combine objects from different scenes. This enables the movement and rotation of objects in the generative new image. GRAF and [57] can generate images of cars in different poses, but the resulting backgrounds are often monochromatic or pure white. 3D-SDN [105] and PNF [55], are the algorithm that can modify both the 3D pose and position of the spawning vehicles while remaining realistic city and road scenes.

2.3 Datasets

2.3.1 Autonomous Driving Datasets

KITTI [31] is the pioneering benchmark dataset for use in autonomous driving by providing LiDAR sensors, stereo cameras, and GPS/IMU data. Compared with KITTI, Waymo [92] provides a large-scale, high-quality dataset with high-intensity annotations and higher annotation frequency using five cameras and five LiDARs from different angles and locations. nuScenes [11] provides 360overage from the LiDAR, radar, and camera sensors. Argoverse [18] contributes detailed geometric and semantic maps of the environment, and its sibling Argoverse2 [102] has the most extensive self-driving taxonomy with HD maps that include real-world changes. Cityscapes [21] provides semantic, instance-wise annotations for semantic understanding of urban street scenes. The current state-of-the-art (SOTA) datasets are inherently designed for independent model training based on various sensors rather than handling real-world challenges. As [88] estimated that AVs need to drive 30 billion miles to get enough statistical evidence to prove that AVs are three times safer than human drivers, yet leading Waymo claims their test fleet has run 20+ million miles on public roads [84]. Therefore, corner case or out-of-distribution data in existing autonomous driving datasets are insufficient. These defects in the previous works inspired us to fill the gap.

2.3.2 Synthetic Datasets

Advances in computer graphics have made it possible to easily annotate and generate virtual datasets, such as SYNTHIA, Virtual KITTI (vKITTI), and Virtual KITTI 2 (vKITTI2), include various scene types under different weather, environment, and lighting conditions [10, 29, 79]. [43] demonstrate that SOTA neural networks trained using only synthetic data perform better than the same architectures trained on real-world dataset. CARLA [24] supports custom waypoints input for vehicle trajectory generation, but it is projected in a virtual city built in a game engine without any modification compatibility to an existing real-world dataset.

2.4 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) are techniques that produce interpretable and transparent models, enabling end-users to comprehend and trust the decision-making process of AI systems. XAI aims to bridge the gap between complex black-box models and human understanding, providing insights into model behaviors, reasoning, and decision factors. The goal is to enhance transparency, accountability, and trustworthiness in AI systems, enabling users to assess model performance, detect biases, ensure fairness, and facilitate decision-making. XAI techniques encompass various approaches, including rule-based systems, feature importance analysis, visual explanations, and interactive interfaces.

Various metrics to evaluate explainability can be found in the field of Explainable AI (XAI). Gilpin [32] developed a classification for XAI techniques and concluded that there are three categories of explanations: Explanations of Deep Network Processing, Explanations of Deep Network Representations, and Explanation-Producing Systems. In general terms of *processing*, *Linear Proxy Models*, *Automatic-Rule Extraction*, *Decision Trees*, and *Saliency Mapping* [90] are four kinds of tools widely accepted by researchers. A discussion of *Grad-CAM* [86], a subbranch of Saliency Mapping, is investigated in this paper. In the context of *representations*, there are three kinds of techniques: *SHAP* [60], *Network Dissection* [4], *TCAV* [49]. The discussion of *explanation-producing* is outside the scope of this paper.

2.4.1 Saliency Mapping

The *Saliency Maps*, or *Vanilla Gradient*, was first introduced by Simonyan et al. with a simple idea by calculating the derivatives of score S_c given by a convolutional neural network to the input image I [65, 90].

$$w = \frac{\delta S_C}{\delta I}|_{I_0}$$

Saliency maps use the same backpropagation formula as the training stage so that we can evaluate the saliency maps of all commonly used AI frameworks with minimal effort. Nevertheless, saliency maps has a saturation problem according to Shrikumar et al. [89]. When using a ReLU in activation layers, such derivatives in saliency maps can miss important information that flows through a network.

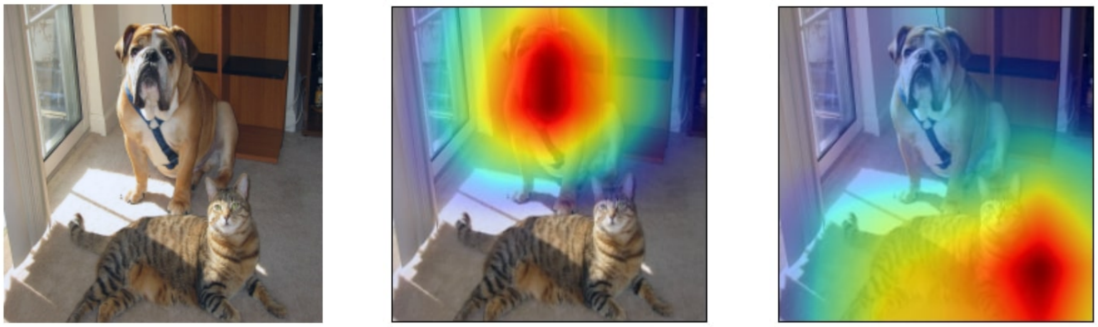
2.4.2 Grad-CAM

Grad-CAM stands for *Gradient-weighted Class Activation Map.* As the name suggests, it's based on the gradient of neural networks. Grad-CAM was introduced in 2017 by Selvaraju et al. [86] for weakly-supervised localization, weakly-supervised segmentation, and providing insight into model failure modes. The goal of Grad-CAM is to visualize where of an image a convolutional layer is "looking" for a particular classification. Grad-CAM is a *Class Activation Mapping* (CAM) [108] style technique for creating a class-specific heatmap based on a particular input image, a trained Convolutional Neural Network (CNN) [56], and a chosen class of interest. As long as the layers of the CNN are differentiable, Grad-CAM can be calculated on any CNN architecture.

$$L^c_{Grad-CAM} \in \mathbb{R}^{u \times v} = ReLU\left(\sum_k \alpha^c_k A^k\right)$$
$$\alpha^c_k = \frac{1}{Z}\sum_i \sum_j \frac{\delta y^c}{\delta A^k_{ij}}$$

Grad-CAM and gradient base saliency maps have long been considered the most reliable techniques. However, recent research identifies a critical problem with Grad-CAM: Grad-CAM sometimes highlights regions that the model did not actually use, indicating that it is an untrustworthy model explanation method. As a result, HiResCAM is recommended instead of Grad-CAM for a model explanation [25].

Another concern about Grad-CAM is its sensitivity to the model or data. Insensitivity to model or data is highly unwanted because it implies that the "interpretation" has not interacted with the model or the data. Adebayo et al. [1] proved that



(a) Original Image (b) Grad-CAM 'Dog' (c) Grad-CAM 'Cat'

Figure 2.2: Explanations with Grad-CAM for ResNet presented in [39]

partial gradient-based saliency methods are sensitive to model and data. The insensitivity test described in this paper was passed by Vanilla Gradient and Grad-CAM, but Guided Backpropagation and Guided Grad-CAM failed. Regardless, this article was also criticized by Tomsett et al. [94] for lack of consistency in test metrics.

2.5 Robustness

Our framework is a layer that improves robustness by introducing out-of-domain semantically-significant samples. Our work is similar to a knowledge-driven cognitive model approach to improve AI system robustness [61], with a focus on enhancing autonomous vehicle data sets with predefined primitives. We argue that using a primitive representation, similar to the abstract script-like representation in language [7, 83], can make the underlying opaque system more understandable. [2] demonstrates that the VISTA simulator can generate novel camera images to improve out-of-domain datasets.

However, the generated photo-realistic pictures can only be obtained via changing the driving angle of the ego vehicle but not the pose of other cars on the road.

Our work is inspired by prior work on a stress testing framework for autonomous system verification and validation [26]. The key idea for DANGER is that the robustness should be tested dyanmically with a series of automatically generated stress tests. Our goal is to use these generative examples to generate counterfactuals. This is similar to previous work using adversarial examples for explaining counterfactuals [71]. Our contribution is a set of semantic primitives, similar to the semantic layer in DNNs discussed in [9].

According to the systematization of corner case detection complexity proposed by [8], our framework is designed to obtain the most dangerous anomalous scenario for a detector, and we also answer the research question, '*How to generate or record corner case from descriptions?*', proposed by [5]. [8] defined a corner case as a non-predictable relevant object/class in a relevant location, [8] extended the definition of corner case level by giving three most dangerous scenes: the anomalous scenario is a potentially dangerous unknown object, the novel scenario is a harmless unknown object, the risky scenario is a potentially dangerous but known object. In the result section, we present the design our primitives to address these problems.

Chapter 3

Methodology

The primary research methods employed in this study involve dataset generation and the evaluation of a specific neural network model using XAI metrics. To address this, we present a novel approach for generating a photo-realistic dataset of the autonomous vehicle scenario. In the initial step, we utilize a 3D-aware image synthesis algorithm to generate the dataset. Subsequently, we analyze and compare the explanations provided by interpretable models for both the original and generated datasets. In this chapter, we introduce the DANGER Framework along with three datasets used in our study: the vKITTI dataset, vKITTI 2 dataset, and the Waymo Open Datasets [10, 29, 92].

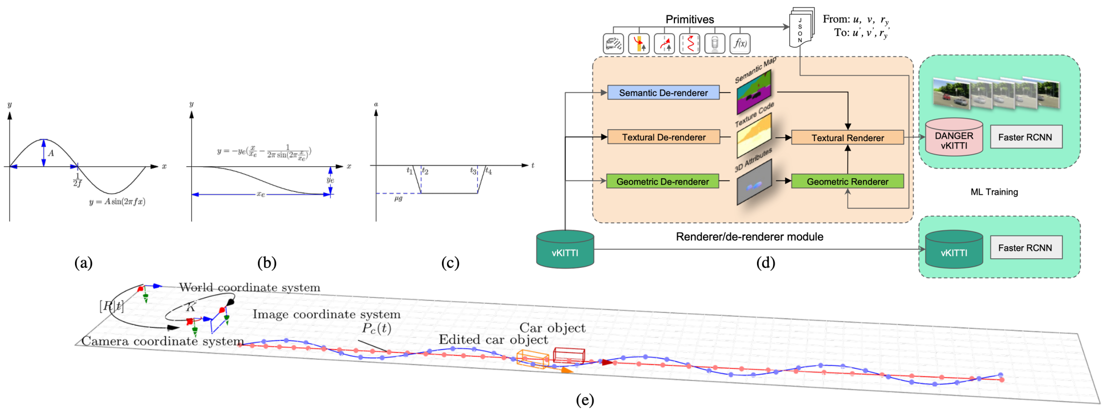
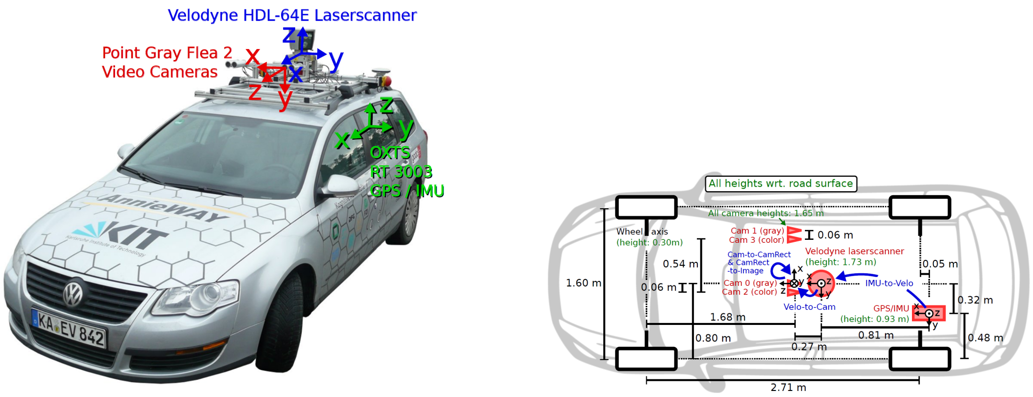


Figure 3.1: DANGER framework, primitives, and visualization of scene editing, in which we modify the original datasets as input and we build dangerous driving datasets on top of it

3.1 Framework and Datasets

3.1.1 DANGER Framework

The DANGER Framework, illustrated in Figure 3.1, encompasses a renderer/de-renderer module, a primitive function module, and a generated descriptive file. This framework empowers users to tackle a wide range of corner cases efficiently. The scene's fundamental editing is facilitated by pre-refined primitives, exemplified by the lane change, cut-in, and braking functions depicted in Figure 3.1(a)(b)(c). As demonstrated in the slalom-lane-change scene editing of scene `0006` in world coordinate, the orange-red curve represents the original trajectory of car object `tid` 4, while the light blue curve denotes the corresponding edited trajectory in world coordinate.



(a) Instrumentation

(b) Sensor location

Figure 3.2: Setup of the KITTI datasets

3.1.2 Virtual KITTI Dataset

Virtual KITTI is a computer-generated simulation of The KITTI Dataset, a benchmark for autonomous driving research that was initially introduced in 2013. As shown in Figure 3.2, the original dataset consists of data captured by two grayscale cameras, two color cameras, a Velodyne LiDAR scanner, and a GPS/IMU unit. To obtain explicit annotations, we use the raw data of KITTI, and the corresponding association of KITTI, vKITTI/vKITTI2, and Waymo are shown in Table 3.1.

In 2015, Virtual KITTI was released with 50 photo-realistic monocular videos (21,260 frames) synthesized from virtual worlds under various lighting and weather conditions, respectively. A real-to-virtual cloning method was used to convert five driving video sequences from the original KITTI dataset [31] into the Virtual KITTI, built automatically using the Unity game engine. As shown in Figure 3.3, vKITTI are photo-realistic rendered images with fully annotated information: bounding box, optical flow, depth labels, category, and instance-level segmentation.

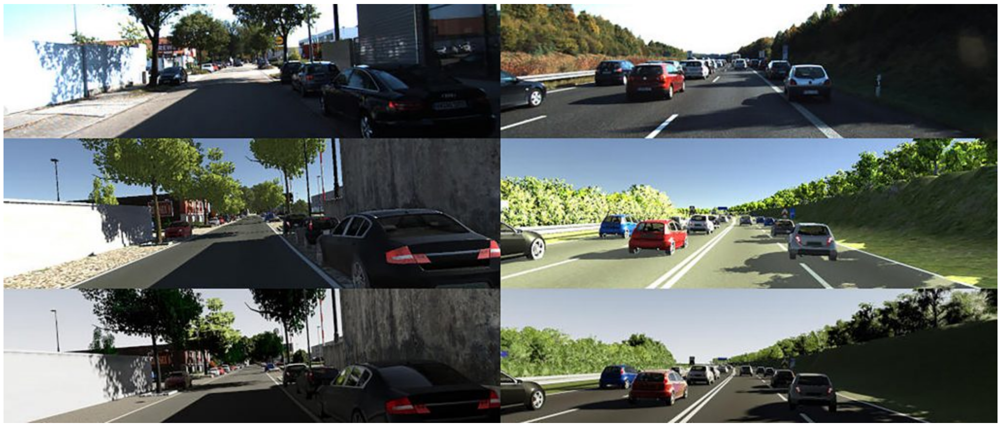


Figure 3.3: Example images from the original KITTI (top), Virtual KITTI (middle), and Virtual KITTI 2 (bottom) datasets, where we can notice the photorealistic rendering and the rendering style difference.

vKITTI 2 is a recently released augmented version of the original Virtual KITTI dataset. It was made available in 2020 and features various enhancements compared to its predecessor, vKITTI, including more photo-realistic images. It utilizes the updated Unity game engine enhancements and offers new data such as stereo pictures and scene flow. vKITTI 2 consists of the same five sequence clones as Virtual KITTI, but has the following new features.

- **Increased photorealism**: The advances in the Unity game engine 2018.4 mean that the basic Virtual KITTI image sequences are closer to the image sequences of the original real KITTI dataset. Moreover, vKITTI 2 exploits recent improvements in lighting and post-processing of the game engine such that the changes in the virtual sequences are even closer to real changes in conditions.

- **Stereo cameras**: A new camera has been added to vKITTI 2 to offer stereo pictures for compensating the Virtual KITTI's short-come of lacking one camera position compared to the original KITTI dataset.

- **Additional ground truth**: Each Virtual KITTI camera renders an RGB image. It also renders several types of ground-truth: class segmentation, instance segmentation, depth, and forward optical flow. For each sequence, camera parameters as well as vehicle colour, pose, and bounding boxes are provided. In addition, in vKITTI 2, backward optical flow and forward and backward scene flow images are newly provided.

We used the 3D-SDN model of vKITTI 1, as provided by [105]. We then trained our own specific semantic, geometric, and texture models for 3D-SDN on vKITTI 2. The details of model training are discussed in Section 3.2.5.

3.1.3 Waymo Open Dataset

The Perception Dataset of Waymo Open Dataset [92] contains 1,150 scenes, each with 20 seconds of data captured at 10Hz (200 frames per scene). The overall dataset was split into three categories: training, testing, and validation, with corresponding counts of 798, 150, and 202 segments, respectively. Each data frame in the dataset includes 3D point clouds and images collected from five LiDAR and five cameras (position shown in Figure 3.4), and ground truth 3D and 2D bounding boxes annotated by humans in the LiDAR point clouds and camera images, respectively. Each bounding box contains an ID that is unique to that object across the entirety of each scene. For the LiDAR data, this allows for tracking in the whole scene. For the camera data, these IDs are consistent within each camera's images only.

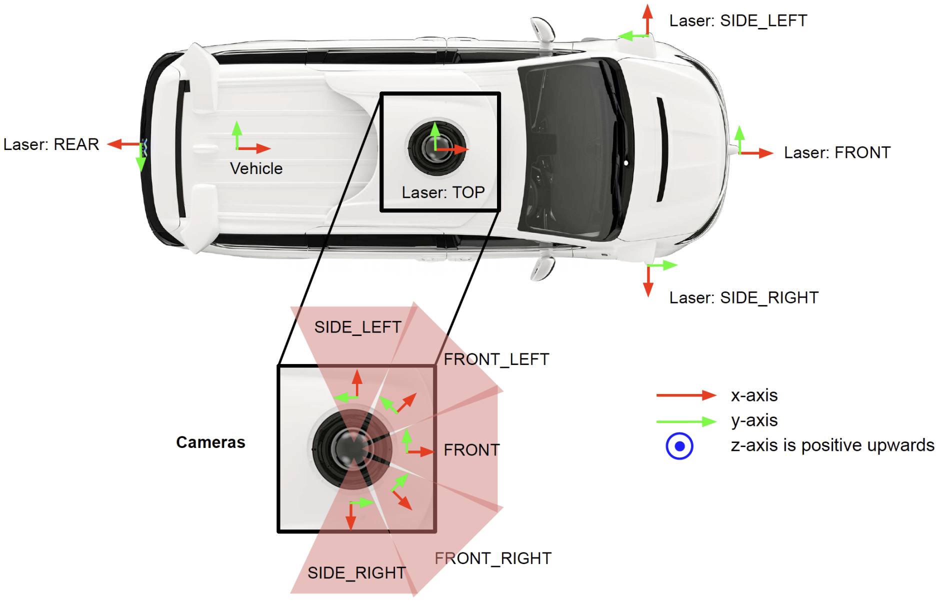


Figure 3.4: Sensor layout and coordinate systems for Waymo data collection.

The Waymo Open Dataset (WOD) Panoptic Viewpoint Synthesis (PVPS) dataset v1.4.2 was released on January 2023, consists of 100,000 images with panoptic segmentation labels. The images are split into a training set, validation set, and test set, using a prescribed split. The dataset is subsampled from the existing 1.15 million images. The availability of panoptic segmentation labels makes it possible to realize 3D-SDN.

The WOD PVPS provides panoptic labels for each frame, which are composed of two sub-labels: semantic label and instance label. The semantic label is the class number for each pixel, and the instance label is the object ID assigned to each pixel for the current frame. However, we developed a unique ID over consecutive frames to localize our editing target. Therefore, we need to convert the instance label into a unique global ID using a conversion dictionary {*instance_id:global_id*} provided by the WOD PVPS. The details of data conversion are described in Section 3.2.1.

KITTI	vKITTI/vKITTI2	Waymo Training Sets
city/2011_09_26_drive_0009	0001	17437352085580560526
city/2011_09_26_drive_0011	0002	10072231702153043603
city/2011_09_26_drive_0018	0006	4468278022208380281
road/2011_09_29_drive_0004	0018	17874036087982478403
road/2011_10_03_drive_0047	0020	16191439239940794174

Table 3.1: The association of KITTI, vKITTI/vKITTI2, and Waymo

3.2 Dataset Generation

In this section, we discuss the input dataset of our framework and how to formally normalize WOD and vKITTI. Then we show the principles of the renderer/de-renderer module and primitive, and discuss how to design new driving trajectory in the global world coordinate system. Finally, we show the training process of the 3D-SDN model by taking WOD as an example, and show the training and 3D-aware image editing results.

3.2.1 Dataset Preparation

Our overall strategy for preparing the datasets for training 3D-SDN is to follow the vKITTI annotation definition. Therefore, we choose the front camera subset of WOD as the peusdo vKITTI for training. The challenge that remains for us is to build a Waymo to vKITTI converter to meet the correct annotation format. In our *Waymo to vKITTI* converter[1], we solved n problems that not being considered in the original *Waymo to KITTI* converter:

- Associating instance IDs across cameras and frames [62].
- Append the unique ID for each car object to the label descriptor.
- Save the front camera's instance-level segmentation images in png format.

3.2.2 Renderer/de-renderer module

The 3D scene de-rendering networks (3D-SDN) [105] is an optimal algorithm that generates photo-level realistic synthetic images. It employs an encoder-decoder architecture with three branches: scene semantics, object geometry, 3D pose, and textual appearance of objects and the background. As shown in Figure 3.1 (d), three branches intend to learn a scene's semantic segmentation, infer the object shape and 3D pose, and encode the appearance of each object and background segment. Disentangling 3D

[1] https://github.com/jayhsu0627/DANGER/tree/main/waymo_kitti_converter

geometry and pose from the given scene enables 3D-aware scene manipulation in image coordinate with a target object location (u, v), orientation pose r_y, and `(delete, modify)` operations described in a JSON file.

The geometric and textural renderer were used to recover the input scene using the generated semantic, geometric, and textural information. Disentangling 3D geometry and pose from texture enables 3D-aware scene manipulation. The geometric branch consists of two components: Mask-RCNN and Derender3D [38]. The textural renderer, or the scene manipulation, is done by pix2pix [42]. To move a car closer, we can edit its position and 3D pose, but leave its texture representation untouched.

In the result section, we use 3D-SDN as the renderer/de-renderer module to demonstrate the feasibility of our framework. In practice, modules such as PNF can also be selected [55].

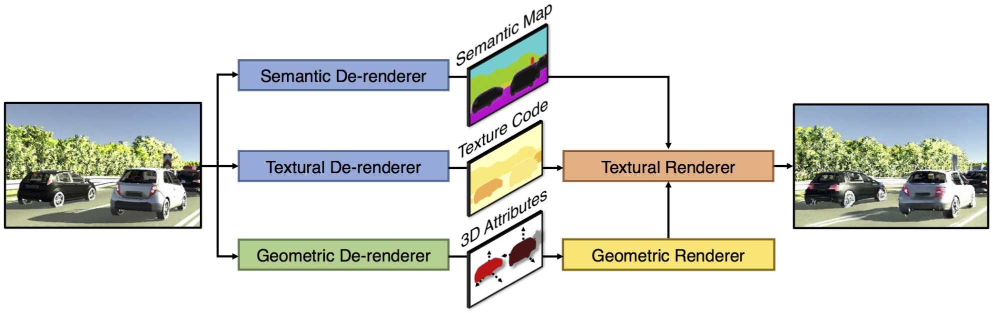


Figure 3.5: 3D Scene De-rendering Networks presented in [105]

3.2.3 Primitives

To augment the input dataset, we define five danger-aware primitives: `Exit parking`, `Cut-in Opposite`, `Cut-in`, `Slalom Lane Change`, and `Braking`. Detail of these primitives will be introduced in Section 4.

3.2.4 Scene Editing Computation

Our approach is adaptive. The object's position can be edited in the real-world 2D plane according to any arbitrary function. At the same time, the implementation of editing in the world coordinate system requires reading the position information of the object and transforming it into the world coordinate system. The vKITTI dataset was designed to match the multi-object tracking (MOT) evaluation benchmark of KITTI. Therefore, the MOT ground truth and exact position of each car object are provided in the folder `motgt`. The following object annotation and terminology we inherited from vKITTI is detailed in [28, 29] and appendix.

We assume that the original driving trajectory of the target vehicle object is a straight line. Given a image $\mathcal{I} \in \mathbb{R}^{W \times H \times 3}$ defined by `scene`, `topic`, `tid`[2], and `frame` with known camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times 4}$ and camera extrinsic matrix $[\mathbf{R}|\mathbf{t}]$, we can convert the position of an object in world coordinate, camera coordinate, or pixel coordinate by Equation (3.1) and Equation (3.3), where $\mathbf{R} \in \mathbb{R}^{3\times 3}$ and $\mathbf{t} \in \mathbb{R}^{3\times 1}$ indicate the rotation and translation matrices [37]. $P_c \in \mathbb{R}^{4\times 1}$ is the 3D point position in camera coordinates and $P_w \in \mathbb{R}^{4\times 1}$ is the 3D point position in world coordinates. In both cases,

[2] A unique track identification number for each object instance in the scene.

they are represented in homogeneous coordinates. A camera extrinsic matrix $\mathbf{M} \in \mathbb{R}^{4\times4}$ is used to denote a projective mapping from world coordinates to pixel coordinates.

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.1}$$

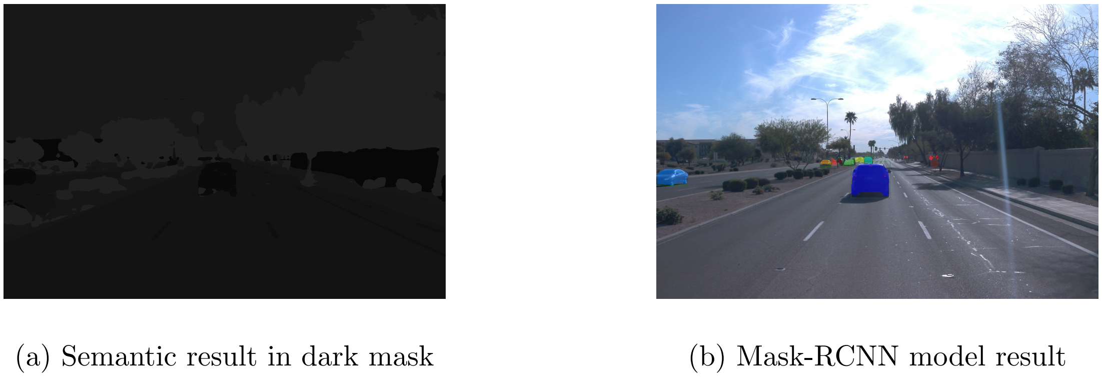
$$\mathbf{P} = \mathbf{KM} \tag{3.2}$$

$$P_c = \mathbf{M} P_w \tag{3.3}$$

The elements of object's center position vector P_c can be acquired from the MOT ground truth data `x3d`, `y3d`, `z3d`, and `h3d`, and the corresponding P_w will be easily obtained by apply the inverse of frame-dependent matrix $\mathbf{M}$. In the x-z plane, arbitrary vehicle poses can be generated according to the primitive function, where $\mathbf{r}'_y$ is a unit tangent vector to the curve at (x'_w, z'_w) representing the target orientation of the car object. The trajectory curve and heading vectors are generated at the origin and then translated to the desired starting point. The curve was further rotated by θ using the rotation matrix $\mathcal{R}_\theta$ to align with the original trajectory's slope a, where a, θ are the slope and angle of original path, $\mathcal{R}$ is a corresponding rotation matrix [100]. A detailed algorithm of the implementation of the primitive operation in 3D world coordinate is shown in the appendix.

3.2.5 Training

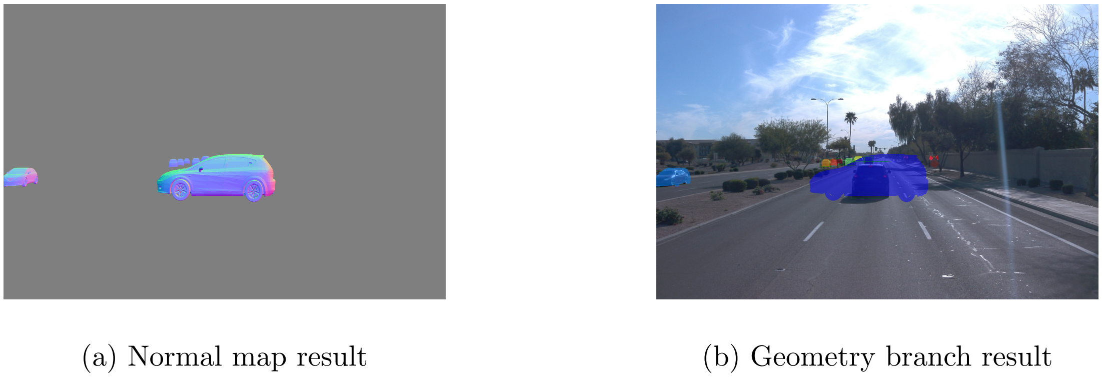
In our experiments, we downscale vKITTI images to 624 × 192 and WOD images to 624 × 416. Our inference model of vKITTI was inherited from [105], and we also trained models followed the setup of [105] for vKITTI2 and WOD on four NVIDIA GeForce 1080 Ti (11G) GPU. The intermediate results for the 3D-SDN, semantic, geometry, and textural models, when applied to frame 24 of the WOD segment 3 in Table 3.1, are depicted in the Figure 3.6-3.8.



(a) Semantic result in dark mask

(b) Mask-RCNN model result

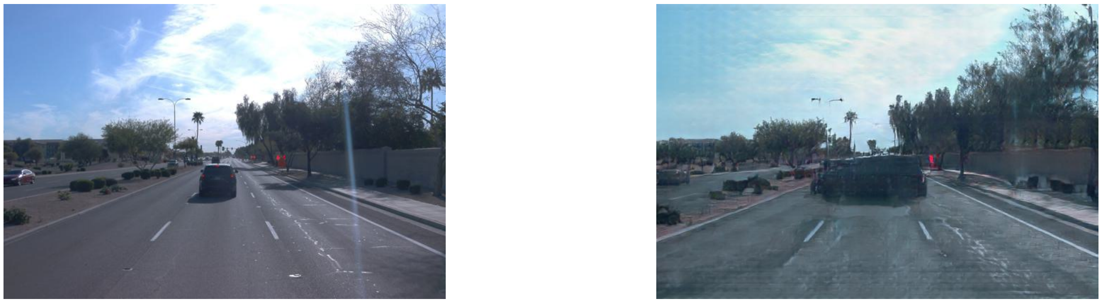
Figure 3.6: Semantic and geometry branch



(a) Normal map result

(b) Geometry branch result

Figure 3.7: Geometry branch



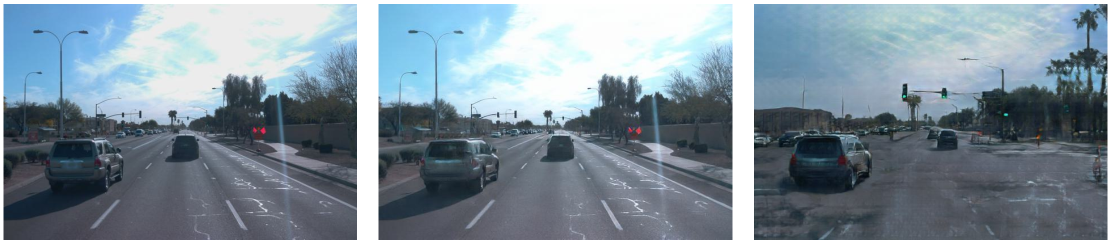
(a) Original image

(b) Synthesis image

Figure 3.8: Textural branch apply a rotation editing on original WOD image

In our experiments on the WOD dataset, we follow a training setup where 14 out of the 20 images in each segment are assigned to the training group, while the remaining 6 images form the test group. For the semantic model, we conduct training with 200 iterations per epoch, totaling 20 epochs. In the MaskRCNN model, we complete 220 epochs, and in the Derender3D model, we train for 256 epochs. Lastly, the textural branch undergoes training for a total of 1058 epochs, which took about 20 hours.

As illustrated in Figure 3.9, by epoch 1050, the textural model trained on the training group demonstrates a high level of fidelity. However, when presented with images from the unseen test group, we observe that the kernel pix2pixHD model of textural branch still exhibits limited drawing capability.



(a) Original

(b) Training

(c) Testing

Figure 3.9: Textural training result on epoch 1050

Chapter 4

Results

In this section, we present the DANGER dataset and primitives we used in our generation framework. We evaluate our results in two aspects that focus on the corner cases generation capability and the realistic level of our dangerous maneuver via a human study.

We hypothesize that adding risky maneuvers to the dataset will result in a higher corner case score for the ego vehicle. We performed a numerical analysis of our generated data frames by applying a corner case detector that considers object-level and predictability and we also validated our results with a user study. Given such a synthesis dataset, we analyzed the Grad-CAM explanation towards a segmentation model on the dataset pair.

4.1 Dangerous Corner Case Generation

4.1.1 Primitives

We designed five scenario-level dangerous corner cases in the world space based on the vKITTI dataset. These primitives are deliberately selected for each scene presented in the vKITTI according to the vehicle object's position and motion. Though these are hand-curated, each primitive follows the definition of a scenario-level corner case that is an anomalous or risky scenario derived from the real-world. We also refer to the safety assist test procedure and autonomous vehicle collision report as templates for our dangerous corner cases [13, 95].

Cut-in Many accidents are caused by neighboring vehicles suddenly driving in front of a moving car dangerously, either due to the driver impatiently overtaking or an unintentional aggressively traverse due to forgetting the highway exits [27, 67]. To achieve a realistic cut-in lane change, we define the single-lane change curvature according to the sinusoidal ramp function [91] as following:

$$y = y_e\left(\frac{x}{x_e} - \frac{1}{2\pi \sin(2\pi \frac{x}{x_e})}\right), \tag{4.1}$$

where x_e is the longitudinal offset of the target position, y_e is the lateral lane-change offset of the target position as shown in Figure 3.1 (b). The forwarding trajectory will be rotated and aligned with the camera space's longitudinal axis z_c. We chose two vehicles with `tid` 1 and 2 in scene `0018` to simulate two scenarios of cut-in ego vehicle and overtaking with nine sets of scene editing parameters, respectively.

Exit Parking A careless driver may suddenly place its front end out of a line of parked cars on either side of a narrow street. We assign a trajectory generated by the cut-in function presented above to two distinguished car objects with `tid` 63 and 70 in scene `0001`. Among the nine sets of parameters for each vehicle, x_e was chosen as a distance of 4 m for one and a half vehicle lengths.

Cut-in on Opposite We incorporated the rotated cut-in function to a car (`tid` 0) driving in the opposite direction. The rampage driver's driving leads to an upcoming accident that potentially causes severe injury in scene `0002`. By combining six x_e and three y_e parameters, we obtained eighteen sets of different driving conditions distributed in a two-dimensional space of ninety square meters.

Slalom Lane Change According to [98]'s study, lane change crashes caused over 244,000 accidents in 1991, accounting for 4.0% of all accidents in the US. Therefore, we aim to design a novel scenario that a driver can barely decide whether other car objects choose to make a lane change or not. As illustrated in Figure 3.1 (a), we borrow the idea from the slalom test in automotive engineering and assign a parameter-dependent sine-wave to the `tid` 2 and 7 in scene `0006` as follows:

$$y = Asin(2\pi fx) \tag{4.2}$$

where A is the lateral offset amplitude in meters, and f is the steering input frequency in Hz.

Braking We simulate a constant deceleration maneuver in scene `0020` when `tid` 16 applies a braking pedal by exploiting the inverted trapezoid piecewise function:

$$a_t = \begin{cases} 0, & t < t_1 \\ -\frac{(t-t_1)\mu g}{t_2-t_1}, & t_1 \leq t < t_2 \\ -\mu g, & t_2 \leq t < t_3 \\ -\frac{(t_4-t)\mu g}{t_4-t_3}, & t_3 \leq t < t_4 \\ 0, & t_4 \leq t \end{cases} \tag{4.3}$$

where a_t is the time dependent acceleration of the editing vehicle, μ and g constitute a uniform deceleration value, t_i is the timestamp when the braking caliper initiated and stopped. As illustrated in Figure 3.1 (c), we replace the object speed in the original video frame with the target braking speed curve obtained by filtering and integrating the generated acceleration curve. Then a new waypoint can be generated in world coordinates to replace the original.

For each primitive, we also calculate the rotation Δr_y, zoomed-in factor ρ, and the transformed (u, v) pixels in camera space. All operation parameters are packaged into a JSON file for 3D-SDN processing. For the detailed JSON and parameter settings, see appendix.

4.1.2 Datasets Generation on vKITTI and vKITTI2

Our DANGER-vKITTI dataset contains 4527 pictures with consecutive frames for each of the five scenes with 18 different sets of parameters. Compared to the original five scenes from vKITTI, DANGER has expanded the original dataset into 90 additional

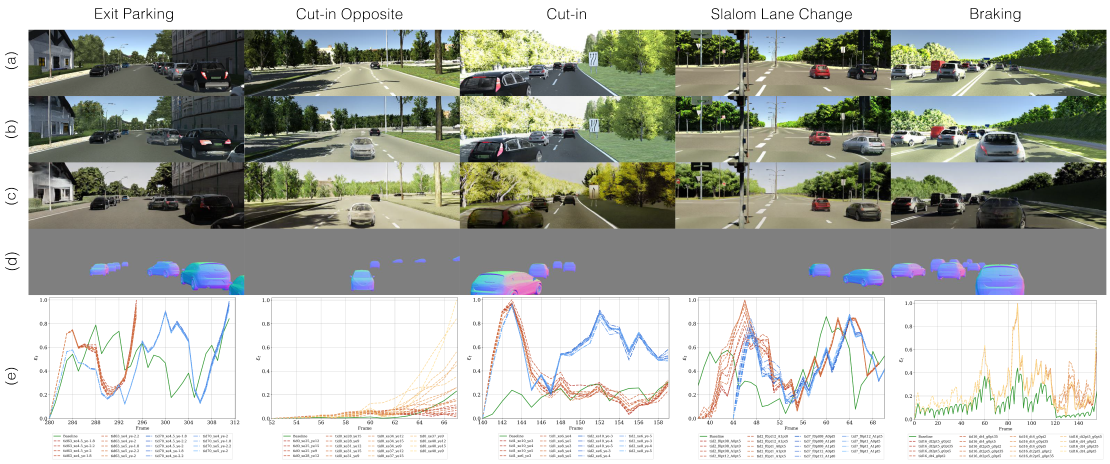


Figure 4.1: Primitive dangerous scenarios and evaluation results

scenes using the experimental parameters.

Generation results for each primitive in our study are shown in Figure 4.1 from left to right: `Exit parking` (frame301), `Cut-in Opposite` (frame67), `Cut-in` (frame144), `Lane Change` (frame52), and `Braking` (frame129). The (a) vKITTI, (b) DANGER-vKITTI, (c) DANGER-vKITTI2, (d) normal map, and (e) a normalized dangerous corner scores evaluation of the generated DANGER datasets (two `tids` in red and blue color cluster, baseline vKITTI in green) are shown from top to bottom.

4.2 Evaluation

4.2.1 Quantitative

We use the conceptual definition of the corner case [6] proposed that a *non-predictable relevant object/class* in *relevant location*, and we hypothesize a dangerous movement is a subset of corner cases. Our corner case detector considers semantic

No.	Metrics	Datasets	Exit Parking	Cut-in Opposite	Cut-in	Slalom Lane Change	Braking
(1)	$\bar{\epsilon}$	vKITTI	0.48	0.04	0.19	0.41	0.13
(2)	ϵ^*	vKITTI	0.84	0.23	0.31	0.86	0.44
(3)	avg. of $\bar{\epsilon}_{\mu\in\mathcal{D}}$	ours	0.46	0.07	0.44	0.45	0.26
(3)/(1)	Ratio		0.97	**1.97**	**2.27**	**1.09**	**1.99**
(4)	avg. of $\epsilon^*_{\mu\in\mathcal{D}}$	ours	0.95	0.29	0.96	0.89	0.98
(4)/(2)	Ratio		**1.13**	**1.26**	**3.15**	**1.03**	**2.23**
(5)	$\epsilon^*_{\mu\in\mathcal{D}}$	ours	1.00	1.00	1.00	1.00	1.00
(5)/(2)	Ratio		**1.19**	**4.33**	**3.27**	**1.16**	**1.21**

Table 4.1: Quantitative evaluation of DANGER-vKITTI with the baseline dataset vKITTI

segmentation and image non-predictability.

First, we adopt a segmentation network based on the `MobileNet-V2` [82] that allows us to classify and localize the *objects* in the scene for which moving objects are considered as *relevant*. Specifically, we use the model pretrained on ImageNet, MS-COCO, and Cityscapes `train_fine` set provided in `DeepLabv3` [19, 21, 23, 58]. Second, we used the advantage of `PredNet` [59] that can sensing the moving objects to compare the deviation between predicted frame $\hat{\mathbf{x}}_t$ and the real next frame $\mathbf{x}_t$ for time t. As a metric for the corner case, we calculate an error e_t to represent the *non-predictable* component of the detection system.

$$\mathbf{e}_t = \hat{\mathbf{x}}_t - \mathbf{x}_t, \tag{4.4}$$

where elements $e_t(i), i \in \mathcal{I}$. Here, each pixel $i \in \mathcal{I}$, where $\mathcal{I}$ denotes the set of pixel indices in the given image, and $\|\mathcal{I}\| = H \cdot W$ represents the number of pixels. The input image $\mathbf{x}_t \in \mathbb{G}^{H\times W\times C}$ with image pixel $x_t(i) \in \mathbb{G}$, where $\mathbb{G} = \{0, 1, \ldots, 255\}$ is the set of gray values, H and W are the image height and width in pixels and $\mathbb{C} = \{1, 2, 3\}$ is the number of color channels. The segmentation network maps the input to output scores $\mathbf{P}_t \in \mathbb{I}^{H \times W \times \|\mathcal{S}\|}$, where $\mathcal{S}$ denotes the set of classes with cardinality $\|\mathcal{S}\| = 19$ and $\mathbb{I} = [0, 1]$. Taking the argmax over the output scores we obtain the $(H \times W)$-dimensional mask $\mathbf{m}_t = \arg\max_{s \in \mathcal{S}} \mathbf{P}_t$, which gives us a pixel-wise classification $m_t(i) \in \mathcal{S}$. Within this work, we limit the target objects $s \in \mathcal{S}_{rel} = \{12, 13, \ldots, 19\}$, which represents the *relevant classes* such as `Person`, `Car`, `Truck`, etc [21].

The information from the two preceding processing steps is combined as a detection system as metric ϵ to evaluate our DANGER datasets. Since we only interested in the target classes defined in $\mathcal{S}_{rel}$, we filter the error map Equation (4.4) with the following formula:

$$e_{t,rel}(i) = \begin{cases} e_t(i), & m_t(i) \in \mathcal{S}_{rel} \\ 0, & m_t(i) \notin \mathcal{S}_{rel} \end{cases} \tag{4.5}$$

To assign higher dangerous weights for the objects near the ego vehicle, we use the weighted squared errors of the relevant classes introduced by [6] as follows:

$$\epsilon' = \sum_{i \in \mathcal{I}} e^2_{t,rel}(i) \cdot (1 - \frac{h_i}{H - 1}), \tag{4.6}$$

with $h_i \in \{0, 1, \ldots, H - 1\}$ being the row index from bottom-up. In contrast to [6], we normalize the error scores ϵ'_t of m^{th} dangerous parameter in set $\mathcal{D} = \{1, 2, \ldots, 18\}$ for the same scene to ensure the peak dangerous frame $\mathbf{x}_t$ of the system generates a corner case score $\epsilon_t = 1$. The corner case score is obtained by normalizing the error score ϵ'_t to a value range from 0 to 1 using:

$$\epsilon_t = \frac{\epsilon'_t - \min_{\tau\in\mathcal{T},\mu\in\mathcal{D}} \epsilon'_{\tau,\mu}}{\max_{\tau\in\mathcal{T},\mu\in\mathcal{D}} \epsilon'_{\tau,\mu} - \min_{\tau\in\mathcal{T},\mu\in\mathcal{D}} \epsilon'_{\tau,\mu}} \tag{4.7}$$

where $\mathcal{T}$ denotes a set of timestamps, and μ denotes the parameter set in $\mathcal{D}$.

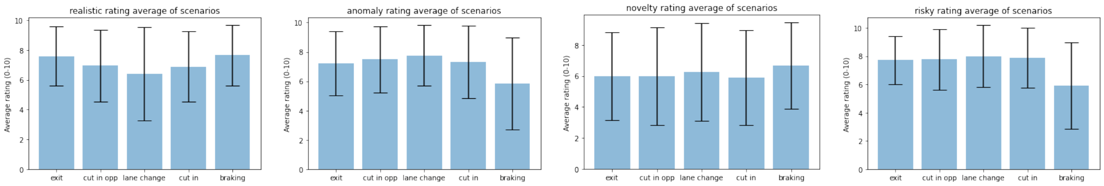
Compared to the green curves for the original vKITTI datasets, Figure 4.1 (e) shows that the corner score increases regardless of which `tid` object is modified. The highest score for `Exit parking` is in the last frame (296 and 312) when the car is completely exiting the side parking; the highest score for `Cut-in Opposite` is in frame 67 when the vehicle is about to crash into the ego; the dangerous score of the `Cut-in` is highest in frame 143 when the black car starts to leave the lane and in frame 152 when the gray car is heading to the opposite lane; the dangerous score is higher in the `Slalom Lane Change` between frames 43 and 52 when the twisting vehicle is relatively close; in the `Braking`, the dangerous score is relatively high in the last frame, 158, and the peak in the middle may be caused by the sudden disappearance of the vehicle object in the frame due to the instability of the 3D-SDN model. Table A.1 shows that DANGER's corner case level is between 1 and 3.15 times that of the original datasets considering the average performance across parameters. Considering only the individual parameters, DANGER's hazard factor can be up to 4.33 times the original data.

4.2.2 User Study

We designed a user study to validate our synthetically generated data samples, i.e., scenarios. Our hypothesis was that users would find the scenarios realistic and the level of “dangerousness” would correlate with Table 4.2 and Figure 4.1 (e). We recruited 100 subjects, who were described the domain and problem setup: that we have augmented a data set to create new scenarios that they are to rank. Users were also presented with the definitions of novel, anomalous, risky and unknown/known events, which are consistent with [8]. Subjects were recruited via Amazon Mechanical Turk and compensated for completing the survey.

Users were presented with a driving scenario and answered four questions for each scenario: to rank whether the scenario was realistic, novel, anomalous, and risky. Users were presented with a Likert scale from 1-10, with 10 indicating very realistic/novel/anomalous/risk, and 1 being not realistic/novel/anomalous/risk at all. We presented a random sample of scenarios, one per primitive. The questions were presented in a random order. The results are in Figure 4.2.

We found that users generally agreed with our hypothesis. Users found our scenarios to be realistic, rating them with average scores in the 7-8 range. Users found the cut-in opposite and lane change scenarios to be the most anomalous. Users found the lane change and braking scenarios to be the most novel. Users found most of the scenarios to be risky. Notably, they did not find the braking maneuvers to be very risky. This might be because the braking scenarios are long: almost 1 minute in length. We hypothesize that users did not watch the full scenario.



(a) Realistic rating (b) Anomaly rating (c) Novelty rating (d) Risky rating

Figure 4.2: Aggregated user study results for rating our scenarios as realistic, anomalous, novel and risky. Averages are plotted as the bars, and error bars show the standard deviation

Primitive	Level	Testimony
Exit Parking	Novel	*'. . . a parked pickup . . . pulled out . . . and made contact with . . . the Waymo AV. . . '* [15]
Cut-in Opposite	Anomalous	*'. . . a sedan was traveling the wrong way on a one-way section . . . made contact with the Cruise AV. . . '* [14]
Lane Change	Novel	N/A
Cut-in	Risky	*'. . . a passenger vehicle in the right adjacent lane abruptly cut in front of the Zoox vehicle. . . '* [17]
Braking	Risky	*'. . . The passenger vehicle ahead . . . transitioned into reverse and began to accelerate, making contact with the front sensor of the Waymo AV. . . '* [16]

Table 4.2: We classify our primitives under the taxonomy of corner cases defined by [8]. We also provide descriptive prompts derived from Autonomous Vehicle Collision Reports in 2022 for each primitive [13].

4.2.3 Grad-CAM Explainablility

This experiment applies Grad-CAM to a semantic segmentation model using the `deeplabv3_resnet50` [19] model from `torchvision` [69]. While primarily designed for classification tasks, the `deeplabv3_resnet50` model can also predict scores for individual pixels in the context of semantic segmentation. In this case, the objective is to compute Grad-CAM with respect to a specific target class, specifically the "car" category, with the goal of maximizing the score associated with that category.

Semantic segmentation offers more possibilities for target selection due to a spatial dimension in the model's output. Based on a study proposed by [97], two alternatives are considered: (a) examining the activation of a single pixel and (b) assessing the collective activation of all pixels associated with the target category. For this tutorial, the second option is chosen as an illustrative example.

We accomplish the desired outcome with the following steps. First, a model wrapper is created to obtain the output tensor, as the `deeplabv3_resnet50` model produces a custom dictionary. Then, a target class is defined and tailored explicitly for semantic segmentation. Implementing the class activation method requires certain decisions, including selecting the layer to be used and the target to be maximized. In this tutorial, the `backbone.layer4` is chosen as the layer, but it can be adjusted according to specific requirements.

All pixels assigned to the "car" category are considered for the target, and their predictions are summed. Following these steps, the Grad-CAM technique is applied to perform semantic segmentation using the `deeplabv3_resnet50` model. This enables identifying and highlighting relevant regions within images belonging to the "car" category, providing valuable insights, such as layer or neuron attribution, for model understanding.

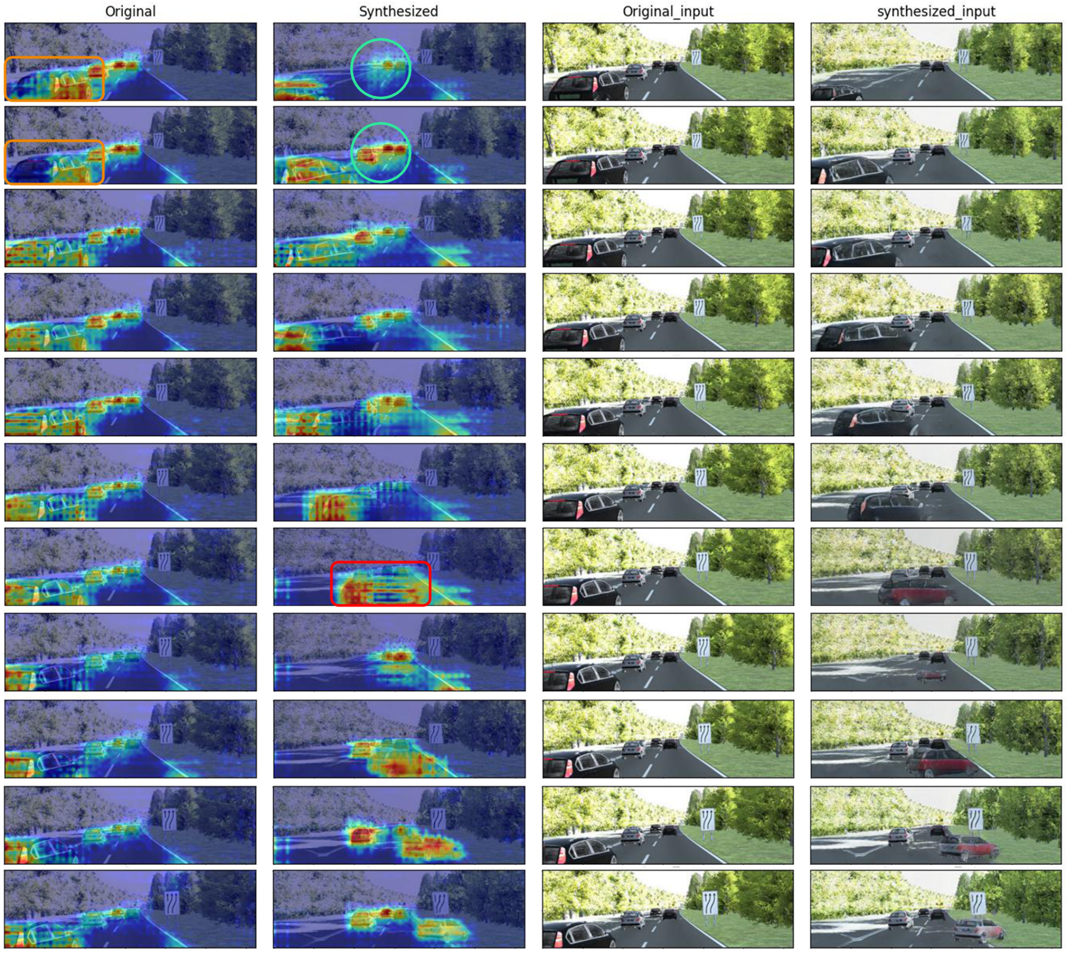


Figure 4.3: Grad-CAM results on vKITTI and Danger vKITTI set

From the explanations of `deeplabv3_resnet50` using Grad-CAM shown in Figure 4.3, for example `0018_clone_tid1_ci_xe6_ye5`, we observe the following: (a)

Temporal instability (orange): The heatmaps of the first two frames of the original data do not completely encompass the vehicle shape, and the shape of these pixel attribution algorithms is changing. (b) Heatmap amplitude instability (sky blue): Although the heatmap indicates the presence of vehicles in the first two frames, it fails to form a stable red-highlighted heatmap. (c) Instability in synthesized data (red): When the vehicle showed its side door in the photo, the explanation becomes fragmented.

Unfortunately, since Grad-CAM is an intuitively interpretable method, we cannot temporarily quantify this failures. In Section 6, we will discuss the feasibility of quantifying this difference by comparing Grad-CAM with ground truth segmentation overlap ratios. As indicated by [1, 51], these pixel attribution methods may be highly unreliable, and even saliency methods may be insensitive to models and data. Therefore, the interpretation of visual models still yields unsatisfactory results, and further research is needed for evaluation.

Chapter 5

Conclusion

In this thesis, we present DANGER: a framework for generating anomalous driving scenarios from existing self-driving vehicle datasets. We defined a set of primitives which align with corner cases and user feedback. We promoted the use of DANGER to robustify existing autonomous vehicle datasets that may not contain error cases. In fact, it might be implausible to collect such corner cases, either because of the intractability of the scenario, or because of the ethical consequences. We presented a framework and technique to define, extract, and classify these scenarios *from existing data*.

Our work has limitations. The 3D-SDN model may generate a jitter frame due to the geometry estimation failure, this might be improved by choosing a more robust renderer/de-renderer module. Similarly, we sometimes get a missing car in a single frame, which may lead to errors in the calculation of the dangerous score. Currently, we can only complete modifications with 3 degrees of freedom. These limitations highlight the challenges of generating corner cases by data perturbation.

We define “dangerousness” as a superclass consisting of novel, anomalous and risky behaviors and actions. Our initial set of primitives are intentionally distributed over anomalous, novel, and risky scenarios. Users generally agree with labels and categorizations. While we demonstrated DANGER on vKITTI, it can be used with any autonomous vehicle dataset that supports semantic and 3D annotations with a given renderer.

We explored the explanations of `deeplabv3_resnet50` using Grad-CAM on our DANGER-vKITTI, and we observed the temporal instability, heatmap amplitude instability, and instability in synthesized data. We show the vulnerability of models `deeplabv3_resnet50` and XAI metric Grad-CAM. A quantitative study of this error and instability may be useful for further root cause studies.

In summary, the DANGER framework is a robustness generator for self-driving car datasets. It is adaptable to multiple types of primitives, and it can cover a wide range of dangerous levels: novel, risky, and anomalous. Our work has opened a new area of robustness data generation, where users, stakeholders, and system designers can identify and easily generate corner cases to augment datasets in order to make them more robust.

Chapter 6

Future Work

Future research can be explored in three primary areas. Firstly, visualization of the 3D model can be achieved using Grad-CAM, facilitating the differentiation between safe and reckless drivers through the generation of 3D Grad-CAM heat maps in conjunction with a language model. By obtaining a 3D model for each frame of the scene using NeRF, it becomes possible to inject more detailed numerical labels into the data, enabling the generation of finer linguistic descriptions that offer enhanced interpretability in combination with Grad-CAM. Inspired by [48], we can combine the large language model to localize the dangerous driver or event in the 3D content.

Secondly, improving the accuracy of picture descriptions can be accomplished by providing more precise labeling information such as car speed, location, color, model, and other relevant attributes. Current state-of-the-art caption labels only provide approximate textual descriptions of the scene, e.g., ‘The car pulls over to the right side of the road.’ Augmenting Grad-CAM’s heat map with more accurate labeling information, such as 'This red Toyota Corolla is slowly pulling over to the right side of the road at 40mph, indicating the driver's safe behavior,' can lead to a more causality-based artificial intelligence approach.

Lastly, the feasibility of quantifying the difference can be discussed by comparing Grad-CAM with ground truth segmentation overlap ratios. By localizing neurons corresponding to different driving objects on the road, it is possible to dissect the deep learning model at a more detailed level, thereby enabling a finer-grained analysis.

Appendix A

Details of Experiments

A.1 Primitive Generation

In this section, we provide more detailed pictures of primitive generation, mainly focusing on the visualization of trajectories in the world coordinate system and the camera coordinate system.

A.1.1 Exit Parallel Parking

In the world coordinate system, we generate the primitive function (blue curve) at the origin, push it to the target point, and convert to the camera coordinate system. The pink color in the figure is the original position of the vehicle parked on the side of the road in the camera coordinate system, while the dark blue point and the light blue arrow are the converted coordinate points and the vehicle head direction converted to the camera coordinate system according to the primitives, respectively.

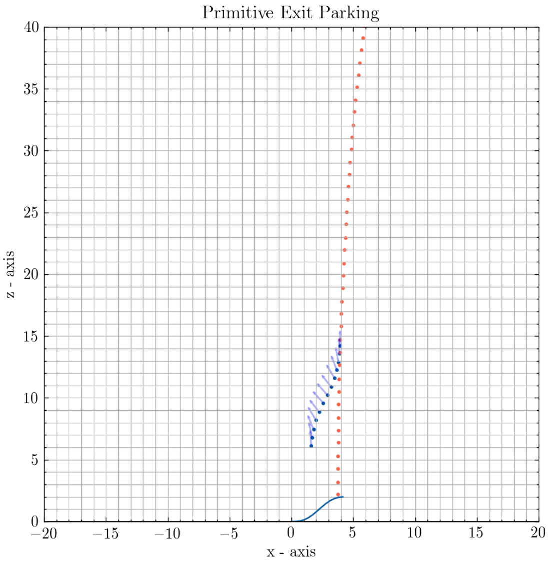


Figure A.1: Visualization of primitive operation in camera space.

A.1.2 Cut-in Opposite

In the world coordinate system, we generate the primitive function (blue curve) at the origin and push it to the target starting point shown as the green curve. The pink color in the figure is the original path curve, while the dark blue point and light blue arrow are the rotated coordinate point and the direction of the converted car head, respectively. The following sections color markings are the same.

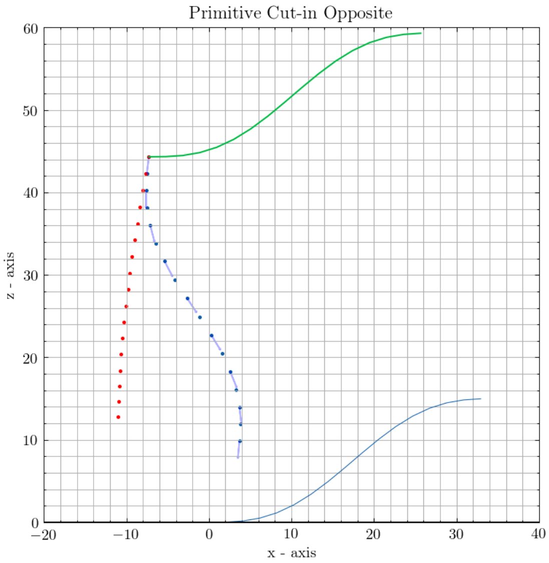


Figure A.2: Visualization of primitive operation in world space.

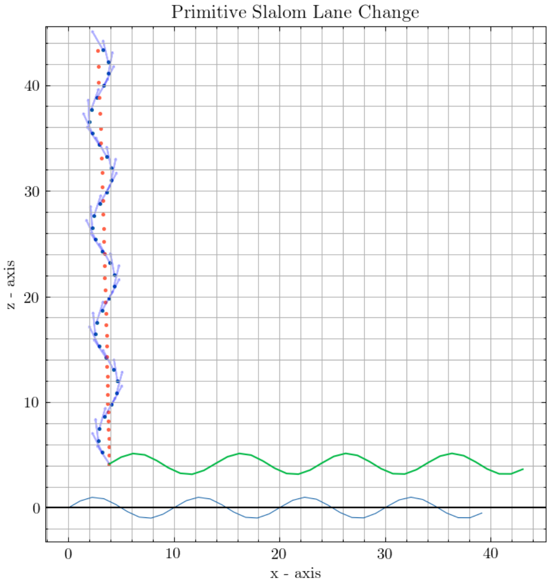


Figure A.3: Visualization of slalom lane change operation in world space.

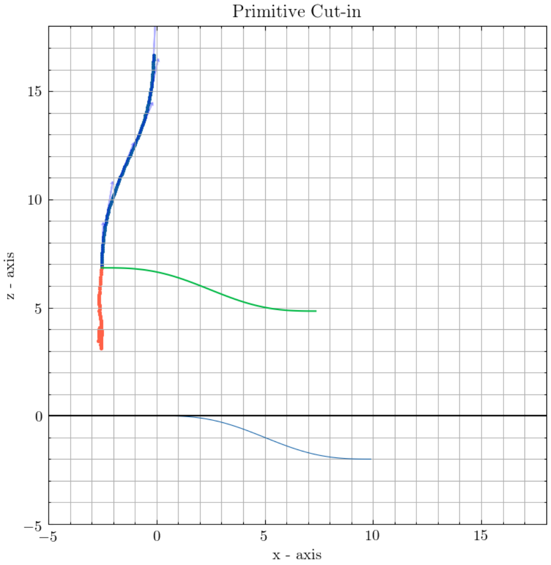


Figure A.4: Visualization of cut-in operation in world space.

A.1.3 Slalom

A.1.4 Cut-in

A.1.5 Brake

Unlike the above method, we simulate the real braking deceleration in the brake primitive and transform it into the velocity and displacement profile of the vehicle object by integration. By re-projecting them onto the original driving trajectory in the world coordinate system, they are then transformed into relative position points in the camera coordinate system. Our approach is not just to bring the vehicle closer by a distance, but to pull the original vehicle to the real trajectory points based on the real physical travel.

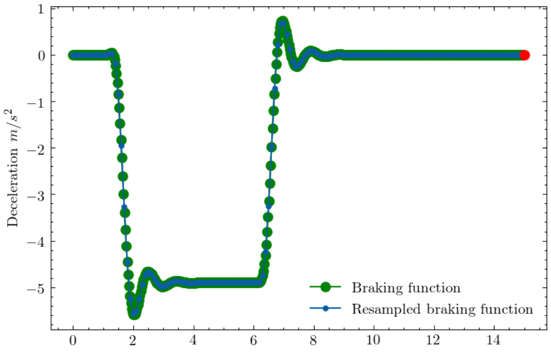


Figure A.5: Braking acceleration based on the parameters input.

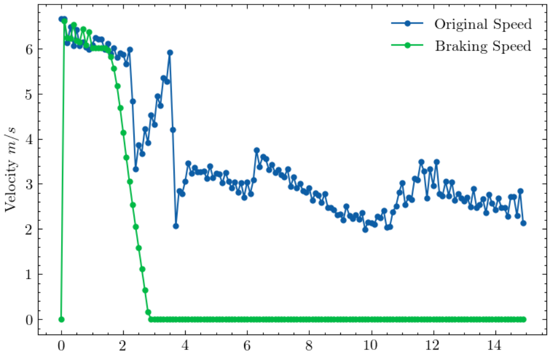


Figure A.6: Original and modified speed profile.

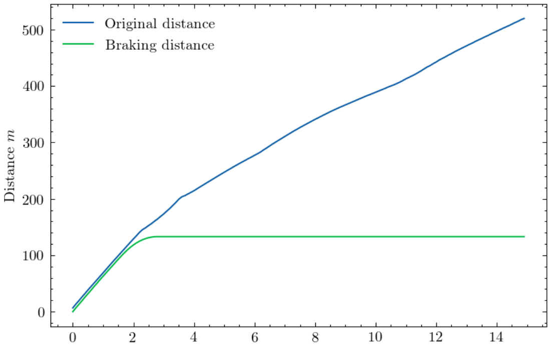


Figure A.7: Original and modified travel distance profile

A.2 JSON file naming

We saved the generated JSON file by the following rule,

`<world>_<topic>_<tid>_<prim>_variable1#1_variable2#2.json`

where

- `<world>` is the name of a virtual world, which is the sequence number of the corresponding original “seed” real-world KITTI sequence (`0001`, `0002`, `0006`, `0018`, `0020)` in the 3D Object Detection Evaluation challenge.

- `<topic>` denotes one of the 10 different rendering variations in terms of imaging or weather conditions. `clone`: rendering as close as possible to original real-world KITTI conditions; `morning`: typical lighting conditions after dawn on a sunny day; `sunset`: lighting typical of slightly before sunset; `overcast`: typical overcast weather (diffuse shadows, strong ambient lighting).

- `tid` The operation will apply to which car object with number `'<tid>'`. Please look up the number under column tid in Table A.1.

- `prim` Primitive parameters defined in Table A.1, and corresponding value `variable#`.

	"0001"	"0002"	"0006"	"0018"	"0020"
	Exit Parking	Cut-in Opposite	Slalom	Cut-in	Braking
	(tid, x_e, y_e)	(tid, f, A)	(tid, f, A)	(tid, x_e, y_e)	(tid, t_1, dt, μ)
Case 1	(63, 4, 1.8)	(0, 20, 9)	(7, 0.1, 0.5)	(1, 6, 4)	(6, 5, 2.5, 0.35)
Case 2	(63, 4, 2)	(0, 23, 9)	(7, 0.1, 1.0)	(1, 8, 4)	(6, 5, 2.5, 0.2)
Case 3	(63, 4, 2.2)	(0, 26, 9)	(7, 0.1, 1.5)	(1, 10, 4)	(6, 5, 2.5, 0.5)
Case 4	(63, 4.5, 1.8)	(0, 29, 9)	(7, 0.08, 0.5)	(1, 6, 3)	(6, 5, 4, 0.35)
Case 5	(63, 4.5, 2)	(0, 32, 9)	(7, 0.08, 1.0)	(1, 8, 3)	(6, 5, 4, 0.5)
Case 6	(63, 4.5, 2.2)	(0, 35, 9)	(7, 0.08, 1.5)	(1, 10, 3)	(6, 5, 4, 0.2)
Case 7	(63, 5, 1.8)	(0, 20, 12)	(7, 0.12, 0.5)	(1, 6, 5)	(6, 10, 2.5, 0.35)
Case 8	(63, 5, 2)	(0, 23, 12)	(7, 0.12, 1.0)	(1, 8, 5)	(6, 10, 2.5, 0.2)
Case 9	(63, 5, 2.2)	(0, 26, 12)	(7, 0.12, 1.5)	(1, 10, 5)	(6, 10, 2.5, 0.5)
Case 10	(70, 4, 1.8)	(0, 29, 12)	(2, 0.1, 0.5)	(2, 6, -4)	(6, 10, 4, 0.35)
Case 11	(70, 4, 2)	(0, 32, 12)	(2, 0.1, 1.0)	(2, 8, -4)	(6, 10, 4, 0.5)
Case 12	(70, 4, 2.2)	(0, 35, 12)	(2, 0.1, 1.5)	(2, 10, -4)	(6, 10, 4, 0.2)
Case 13	(70, 4.5, 1.8)	(0, 20, 15)	(2, 0.08, 0.5)	(2, 6, -3)	(6, 2, 2.5, 0.35)
Case 14	(70, 4.5, 2)	(0, 23, 15)	(2, 0.08, 1.0)	(2, 8, -3)	(6, 2, 2.5, 0.2)
Case 15	(70, 4.5, 2.2)	(0, 26, 15)	(2, 0.08, 1.5)	(2, 10, -3)	(6, 2, 2.5, 0.5)
Case 16	(70, 5, 1.8)	(0, 29, 15)	(2, 0.12, 0.5)	(2, 6, -5)	(6, 2, 4, 0.35)
Case 17	(70, 5, 2)	(0, 32, 15)	(2, 0.12, 1.0)	(2, 8, -5)	(6, 2, 4, 0.5)
Case 18	(70, 5, 2.2)	(0, 35, 15)	(2, 0.12, 1.5)	(2, 10, -5)	(6, 2, 4, 0.2)

Table A.1: Parameters in primitives.

A sample JSON file that contains the (u, v), `zoom`, and r_y information is listed as follows:

```
[
    {
        "world": "0018",
        "topic": "clone",
        "source": "00140",
        "target": "00140",
        "operations": [
            {
                "type": "modify",
                "from": {
                    "u": "350.3",
                    "v": "287.0"
                },
                "to": {
                    "u": "247.9",
                    "v": "319.6",
                    "roi": [ 0, 0, 0, 0]
                },
                "zoom": "1.3252036238820235",
                "ry": "0.056247797876731065"
            }
        ]
    },
]
```

A.3 Class used in `MobileNet-V2`

No.	Class	No.	Class	No.	Class	No.	Class
1	Road	6	Pole	11	Sky	**16**	**Bus**
2	Sidewalk	7	Traffic ligh	**12**	**Person**	**17**	**Train**
3	Building	8	Traffic sign	**13**	**Rider**	**18**	**Motorcycle**
4	Wall	9	Vegetation	**14**	**Car**	**19**	**Bicycle**
5	Fence	10	Terrain	**15**	**Truck**		

Table A.2: Segmentation classes

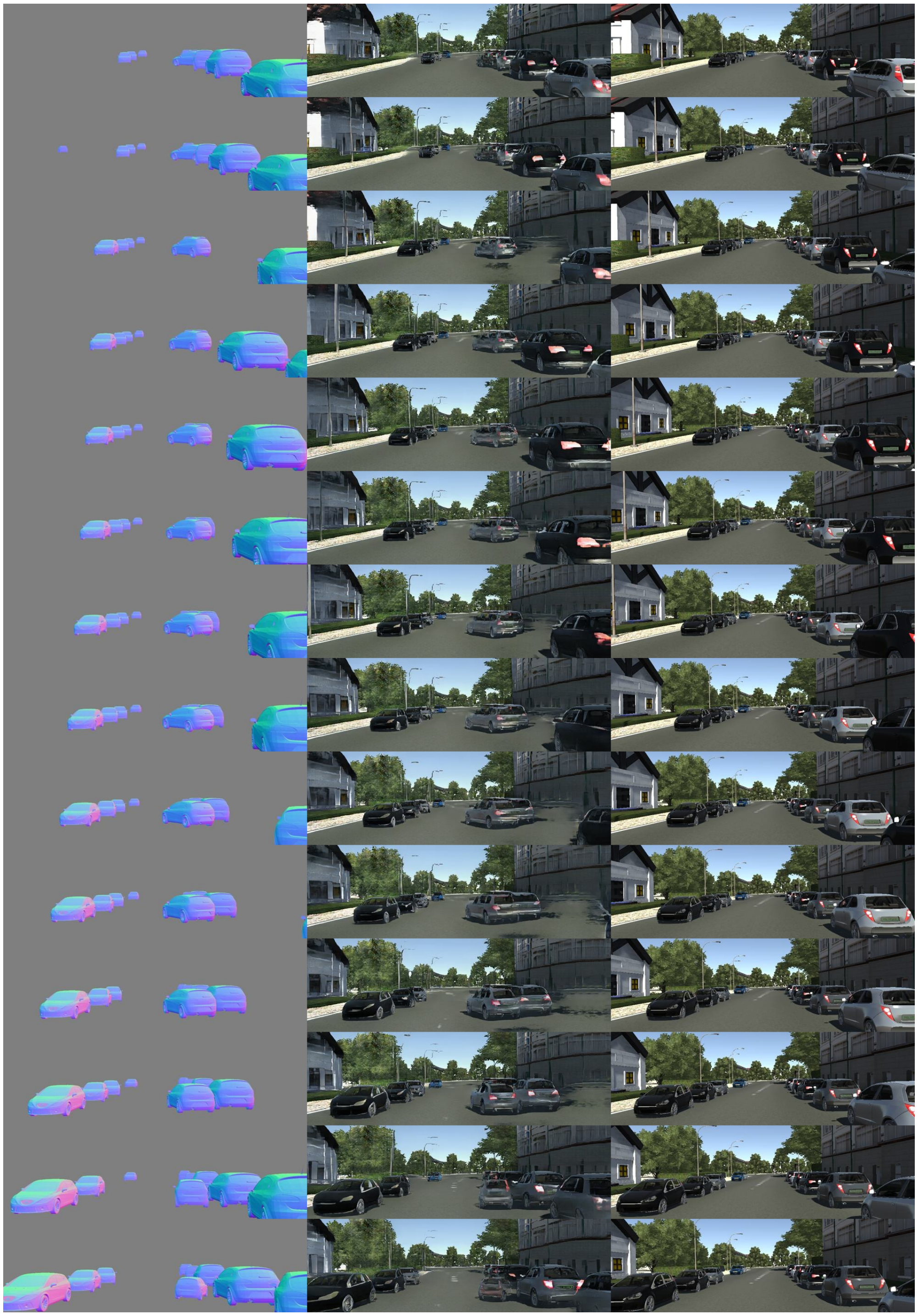


Figure A.8: 0001_clone_tid70_epp_xe5_ye2.2

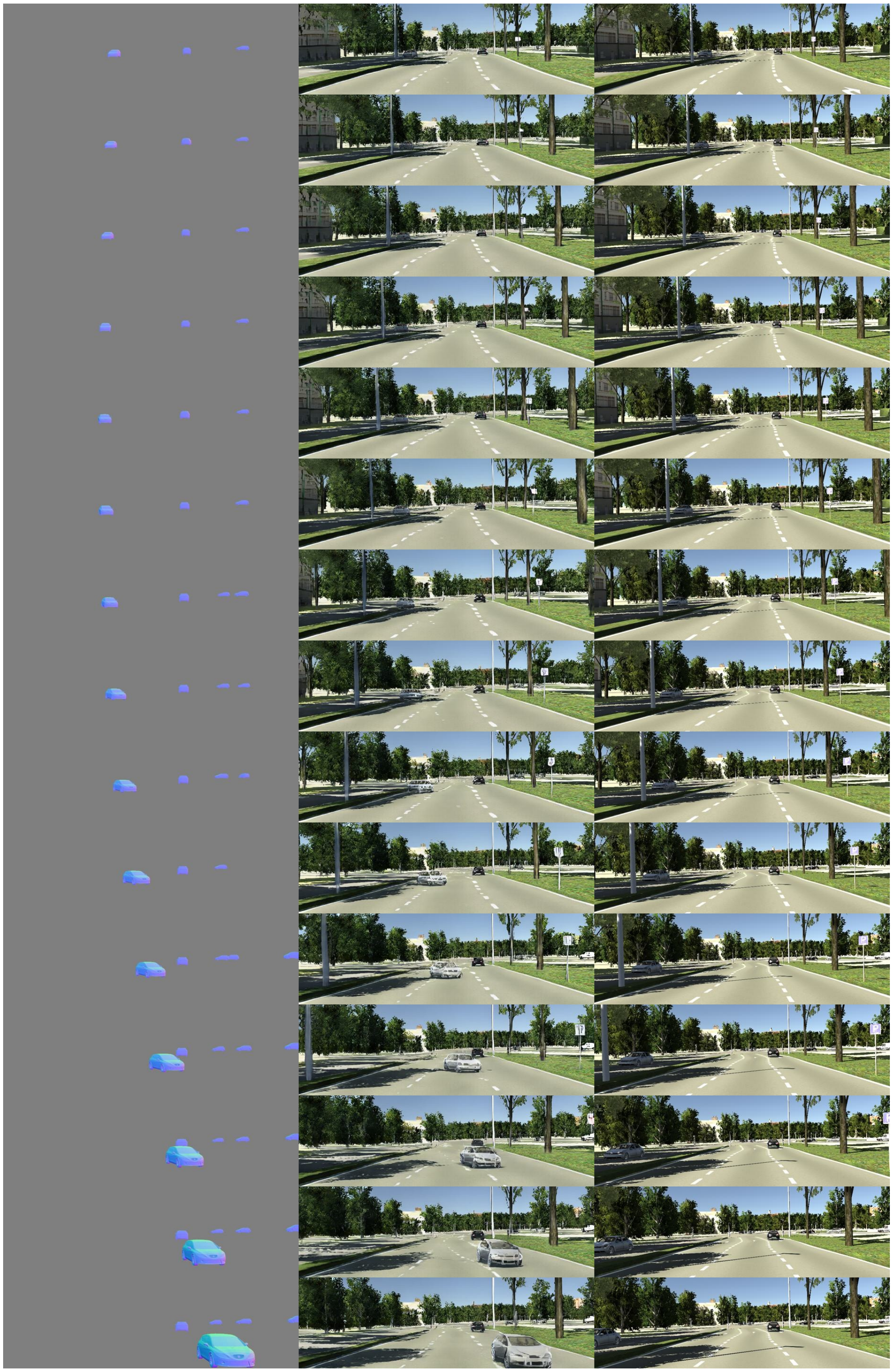


Figure A.9: `0002_clone_tid0_cio_xe40_ye15`

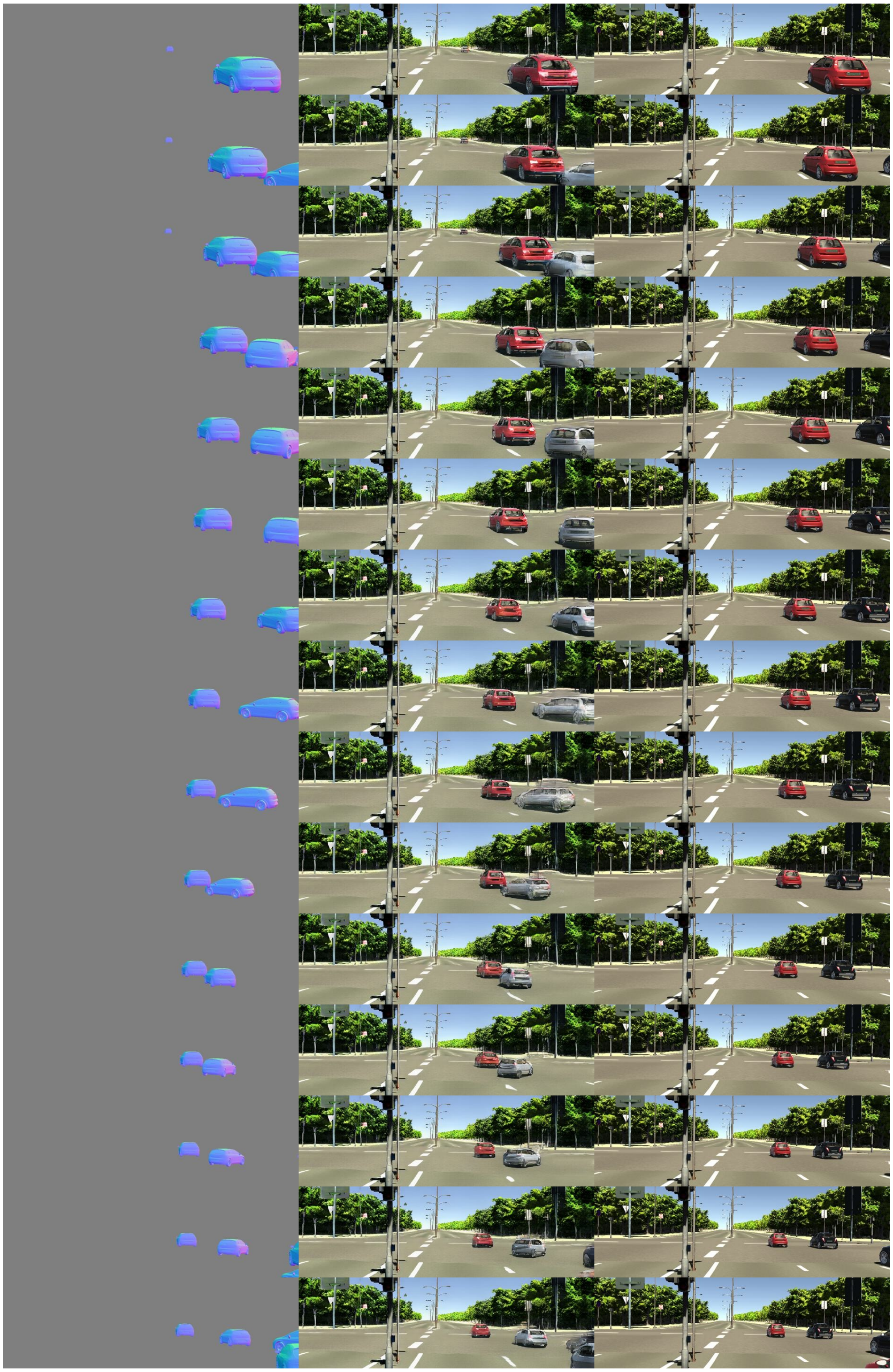


Figure A.10: `0006_clone_tid7_lc_f0pt12_A1pt5`

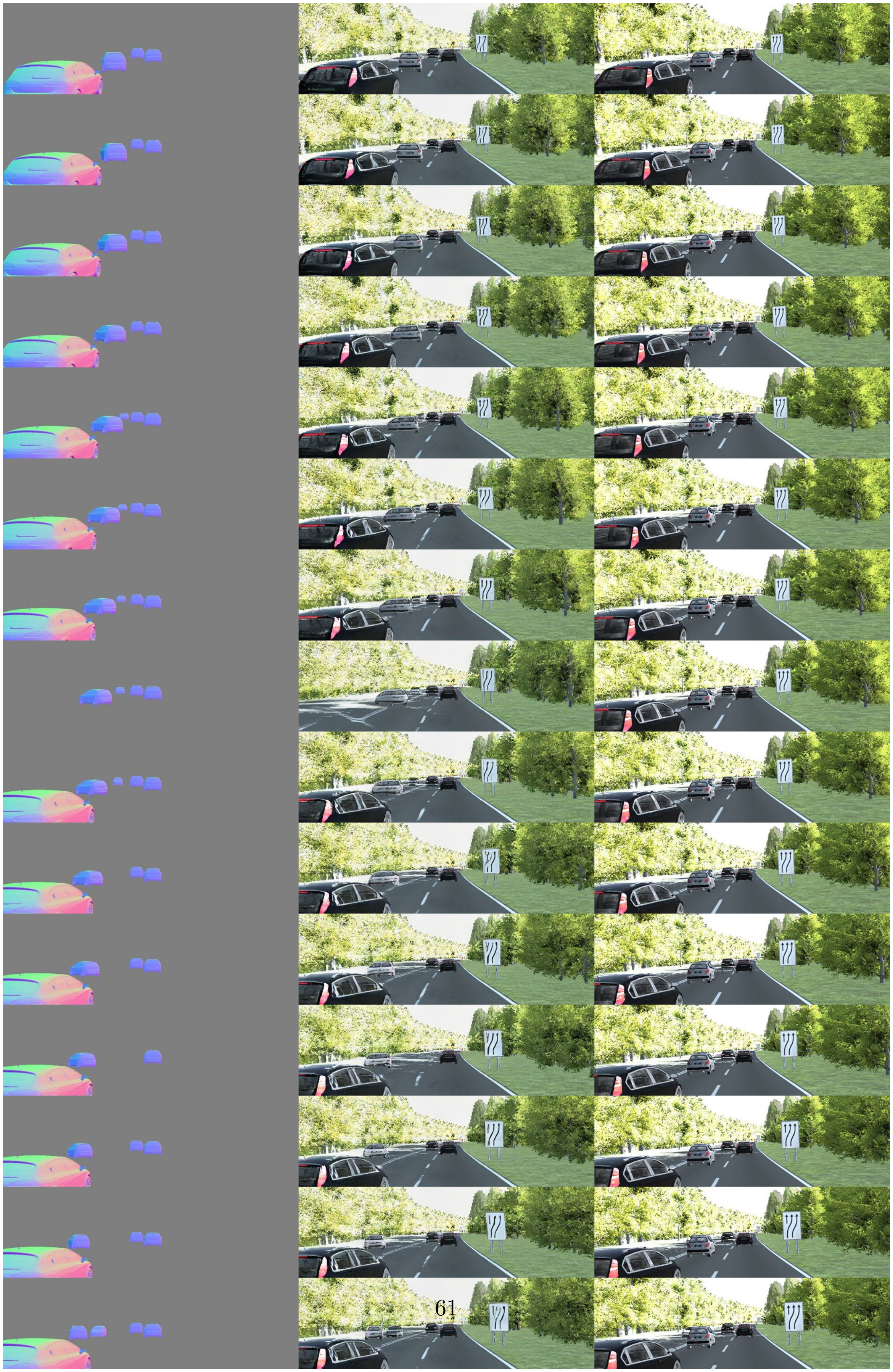


Figure A.11: 0018_clone_tid2_ci_xe10_ye5

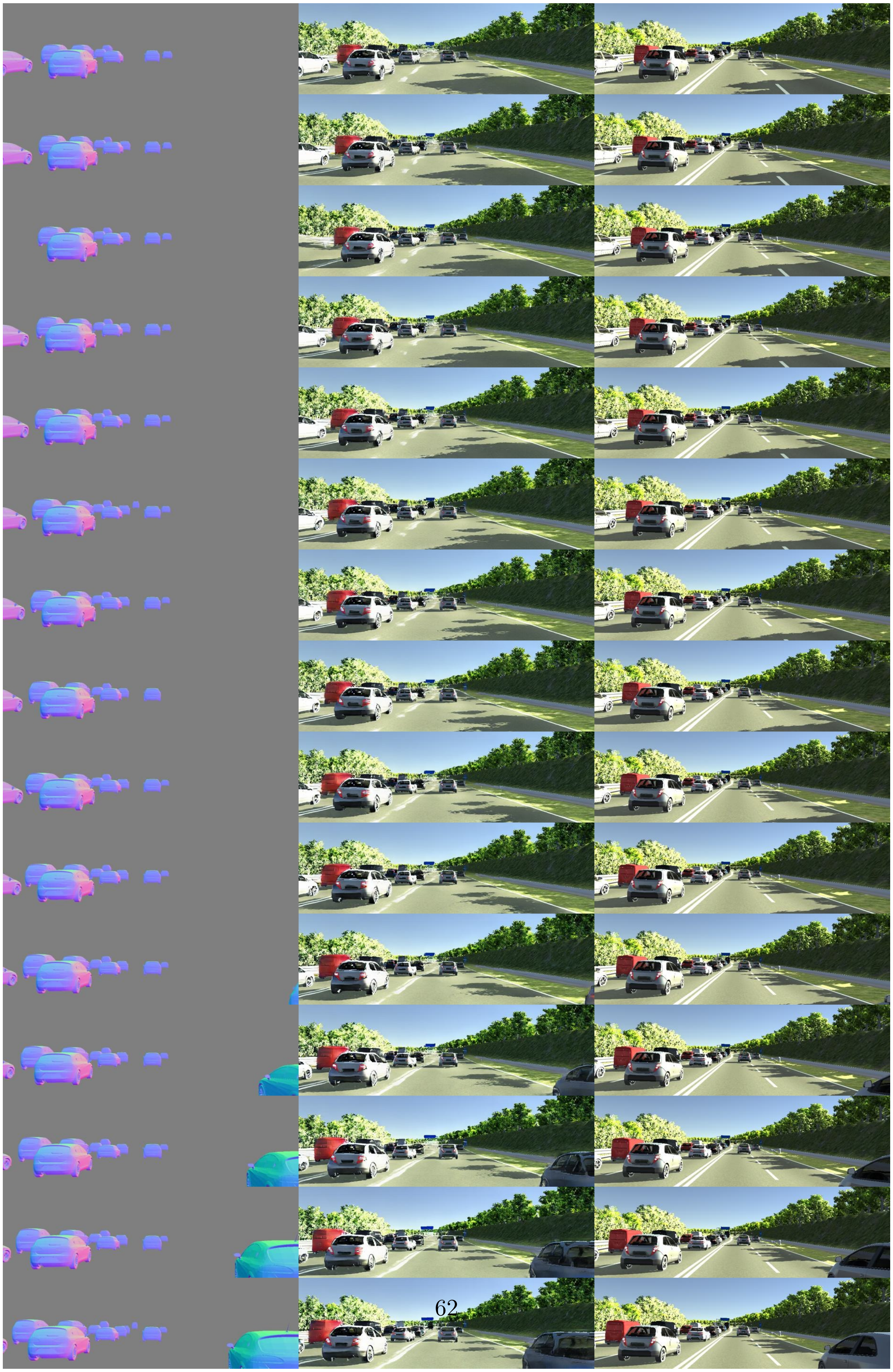


Figure A.12: 0020_clone_tid16_br_t010_dt4_g0pt35

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[2] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2419–2426. IEEE, 2022.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[5] Daniel Bogdoll, Jasmin Breitenstein, Florian Heidecker, Maarten Bieshaar, Bernhard Sick, Tim Fingscheidt, and Marius Zöllner. Description of corner cases in automated driving: Goals and challenges. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1023–1028, 2021.

[6] Jan-Aike Bolte, Andreas Bar, Daniel Lipinski, and Tim Fingscheidt. Towards corner case detection for autonomous driving. In *2019 IEEE Intelligent vehicles symposium (IV)*, pages 438–445. IEEE, 2019.

[7] Gary C Borchardt. Understanding causal descriptions of physical systems. In *AAAI*, pages 2–8, 1992.

[8] Jasmin Breitenstein, Jan-Aike Termöhlen, Daniel Lipinski, and Tim Fingscheidt. Corner cases for visual perception in automated driving: Some guidance on detection approaches. *arXiv preprint arXiv:2102.05897*, 2021.

[9] Kieran Browne and Ben Swift. Semantics and explanation: why counterfactual explanations produce adversarial examples in deep neural networks. *arXiv preprint arXiv:2012.10076*, 2020.

[10] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2. *arXiv preprint arXiv:2001.10773*, 2020.

[11] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[12] Yingfeng Cai, Lei Dai, Hai Wang, and Zhixiong Li. Multi-target pan-class intrinsic relevance driven model for improving semantic segmentation in autonomous driving. *IEEE Transactions on Image Processing*, 30:9069–9084, 2021.

[13] California DMV. Autonomous vehicle collision reports, Aug 2022.

[14] California DMV. *Cruise April 25, 2022*. STATE OF CALIFORNIA, DEPARTMENT OF MOTOR VEHICLES, 2022.

[15] California DMV. *Waymo April 29, 2022*. STATE OF CALIFORNIA, DEPARTMENT OF MOTOR VEHICLES, 2022.

[16] California DMV. *Waymo June 5, 2022*. STATE OF CALIFORNIA, DEPARTMENT OF MOTOR VEHICLES, 2022.

[17] California DMV. *Zoox July 8, 2022*. STATE OF CALIFORNIA, DEPARTMENT OF MOTOR VEHICLES, 2022.

[18] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[19] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[20] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *Advances in Neural Information Processing Systems*, 33:22158–22169, 2020.

[21] 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.

[22] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[24] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[25] Rachel Lea Draelos and Lawrence Carin. Use hirescam instead of grad-cam for faithful explanations of convolutional neural networks, 2020.

[26] Gregory Falco and Leilani H Gilpin. A stress testing framework for autonomous system verification and validation (v&v). In *2021 IEEE International Conference on Autonomous Systems (ICAS)*, pages 1–5. IEEE, 2021.

[27] Scott Ferguson, Yu Guo, and Mohamed Abdel-Aziz. Analysis of driver behavior leading to lane-changing accidents. *Accident Analysis Prevention*, 127:104–112, 2019.

[28] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual kitti dataset. `https://europe.naverlabs.com/research/computer-vision/proxy-virtual-worlds-vkitti-1/`, 2016. (Accessed on 08/13/2022).

[29] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtualworlds as proxy for multi-object tracking analysis. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2016.

[30] Aditya Ganeshan, Alexis Vallet, Yasunori Kudo, Shin-ichi Maeda, Tommi Kerola, Rares Ambrus, Dennis Park, and Adrien Gaidon. Warp-refine propagation: Semi-supervised auto-labeling via cycle-consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15499–15509, 2021.

[31] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[32] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[33] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[35] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022.

[36] D Gunning. Explainable artificial intelligence (xai) darpa-baa-16-53. *Defense Advanced Research Projects Agency*, 2016.

[37] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[38] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

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

[40] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[41] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.

[42] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[43] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*, 2016.

[44] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[45] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-Free Generative Adversarial Networks.

[46] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[47] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and Improving the Image Quality of StyleGAN.

[48] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. *arXiv preprint arXiv:2303.09553*, 2023.

[49] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[50] Yoonsang Kim, Jidong Huang, Sherry Emery, et al. Garbage in, garbage out: data collection, quality assessment and reporting standards for social media data use in health research, infodemiology and digital disease detection. *Journal of medical Internet research*, 18(2):e4738, 2016.

[51] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. *Explainable AI: Interpreting, explaining and visualizing deep learning*, pages 267–280, 2019.

[52] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[53] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[54] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, volume 1, page 2. Citeseer, 2011.

[55] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *CVPR*, 2022.

[56] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[57] Yiyi Liao, Katja Schwarz, Lars Mescheder, and Andreas Geiger. Towards unsupervised learning of generative models for 3d controllable image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5871–5880, 2020.

[58] 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 *European conference on computer vision*, pages 740–755. Springer, 2014.

[59] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[60] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[61] Gary Marcus. The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*, 2020.

[62] Jieru Mei, Alex Zihao Zhu, Xinchen Yan, Hang Yan, Siyuan Qiao, Liang-Chieh Chen, and Henrik Kretzschmar. Waymo open dataset: Panoramic video panoptic segmentation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIX*, pages 53–72. Springer, 2022.

[63] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[64] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[65] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[66] Urwa Muaz. Autoencoders vs pca: When to use whichnbsp;?, Jul 2019.

[67] National Highway Traffic Safety Administration (NHTSA). Traffic safety facts: Overview of motor vehicle crashes. Technical Report DOT HS 812 069, National Highway Traffic Safety Administration, 2019.

[68] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11453–11464, 2021.

[69] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[70] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[71] Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 4574–4594. PMLR, 2022.

[72] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[73] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[74] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

[75] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 30, 2017.

[76] Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8555–8564, 2021.

[77] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR, 2016.

[78] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[79] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[80] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[81] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[82] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[83] Roger C Schank. The primitive acts of conceptual dependency. In *Theoretical issues in natural language processing*, 1975.

[84] Matthew Schwall, Tom Daniel, Trent Victor, Francesca Favaro, and Henning Hohnhold. Waymo public road safety performance data, 2020.

[85] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33:20154–20166, 2020.

[86] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[87] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[88] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017.

[89] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[90] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[91] Nathaniel H Sledge Jr and Kurt M Marshek. Comparison of ideal vehicle lane-change trajectories. *SAE transactions*, pages 2004–2027, 1997.

[92] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[93] Rachel Thomas and David Uminsky. The problem with metrics is a fundamental problem for ai. *arXiv preprint arXiv:2002.08512*, 2020.

[94] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics, 2019.

[95] Michiel R Van Ratingen. The euro ncap safety rating. In *Karosseriebautage Hamburg 2017*, pages 11–20. Springer, 2017.

[96] Bertie Vidgen and Leon Derczynski. Directions in abusive language training data, a systematic review: Garbage in, garbage out. *Plos one*, 15(12):e0243300, 2020.

[97] Kira Vinogradova, Alexandr Dibrov, and Gene Myers. Towards interpretable semantic segmentation via gradient-weighted class activation mapping (student abstract). In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13943–13944, 2020.

[98] J-S Wang. *Lane change/merge crashes: problem size assessment and statistical description.* US Department of Transportation, National Highway Traffic Safety Administration, 1994.

[99] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[100] Eric W Weisstein. Rotation matrix. *https://mathworld. wolfram. com/*, 2003.

[101] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. Ab3dmot: A baseline for 3d multi-object tracking and new evaluation metrics. *arXiv preprint arXiv:2008.08063*, 2020.

[102] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[103] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 699–707, 2017.

[104] Gangwei Xu, Junda Cheng, Peng Guo, and Xin Yang. Acvnet: Attention concatenation volume for accurate and efficient stereo matching. *arXiv preprint arXiv:2203.02146*, 2022.

[105] Shunyu Yao, Tzu Ming Harry Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, William T. Freeman, and Joshua B. Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *Advances in Neural Information Processing Systems*, 2018.

[106] Éloi Zablocki, Hédi Ben-Younes, Patrick Pérez, and Matthieu Cord. Explainability of vision-based autonomous driving systems: Review and challenges. *arXiv preprint arXiv:2101.05307*, 2021.

[107] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[108] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization, 2015.

[109] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[110] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[111] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: Image generation with disentangled 3d representations. *Advances in neural information processing systems*, 31, 2018.